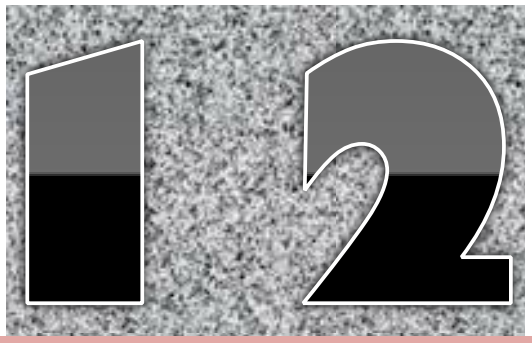


Uncertainty in Artificial Intelligence

Proceedings of the Twenty-Eighth Conference

Edited by

Nando de Freitas

Kevin Murphy

Uncertainty in Artificial Intelligence

Proceedings of the Twenty-Eighth Conference (2012)

August 15–17, 2012 Catalina Island, United States

Edited by
Kevin Murphy, Google Research, United States
Nando de Freitas, University of British Columbia, Canada

Conference Chair
Fabio Gagliardi Cozman, Universidade de São Paulo, Brazil

Local Arrangements Chair
David Heckerman, Microsoft Research, United States

Sponsored by
Microsoft Research, Google, Artificial Intelligence Journal, Pascal2 Network, Charles River Analytics, IBM Research

AUAI Press Corvallis, Oregon

The cover design is used with permission from Elsevier.

Published by AUAI Press for
Association for Uncertainty in Artificial Intelligence
http://auai.org

Editorial Office:
P.O. Box 866
Corvallis, Oregon 97339
USA

ISBN 978-0-9749039-8-9

Contents

Preface

This volume contains all papers presented at the 28'th Conference on Uncertainty in Artificial Intelligence (UAI), held at Catalina Island in California, August 15-17, 2012. Papers appearing in this volume were subjected to a rigorous review process: 304 papers were submitted, but only 95 were accepted (24 for oral presentation, 18 for poster spotlight presentation, and 53 for poster presentation). We are confident that the proceedings, like past UAI Conference Proceedings, will become an important archival reference for the field.

We are pleased to announce that the best paper award was given to T. Claassen and T. Heskes for their paper entitled "A Bayesian Approach to Constraint Based Causal Inference". Also, the best student paper award was given to R. Grosse (co-authored with R. Salakhutdinov, W. Freeman, and J. Tenenbaum) for their paper entitled "Exploiting compositionality to explore a large space of model structures".

In addition to the presentation of technical papers, we were very pleased to have the following distinguished invited speakers: Tom Mitchell (Carnegie Mellon University), Bill Macready (D-Wave systems), Pedro Felzenszwalb (Brown University), Alex Gray (Georgia Tech), and Judea Pearl (UCLA). Also, we were happy to have, as Banquet Speaker, Alison Gopnik (UC Berkeley).

UAI 2012 also hosted a tutorial day (organized by Irina Rish), covering the following topics: "Copulas in Machine learning" by Gal Elidan, "Determinantal point processes" by Alex Kulesza and Ben Taskar, "Graphical models for causal inference" by Karthika Mohan and Judea Pearl, and "Continuous-time Markov models" by Christian Shelton and Gianfranco Ciardo.

This year UAI also hosted a new series of one-day workshops (organized by Ruslan Salakhutdinov) covering the following topics: "Causal Structure Learning", "StarAI – Statistical Relational AI", "Uncertainty in Natural Intelligence", and "Bayesian Modeling Applications".

Nando de Freitas and Kevin Murphy (Program Co-Chairs)
Fabio Cozman (Conference Chair)
David Heckerman (Local Arrangements Chair)

Acknowledgements

The success of a rigorously reviewed conference hinges upon the diligent efforts of many parties. We assembled a Senior Program Committee and a Program Committee with internationally renowned experts in the range of specialties covered by UAI. At least three Program Committee members (or in some cases, auxiliary reviewers) reviewed each paper. Despite the time pressure imposed by rigid conference deadlines, these individuals did their best to supply balanced, fair, and helpful reviews to all authors of submitted papers. Our Program Committee did an outstanding job for which the program chairs and authors alike are quite grateful. The Senior Program Committee acted in a supervisory role, ensuring that paper reviews lived up to UAI standards, moderating discussion among Program Committee members and, ultimately, making recommendations to the program chairs. In some cases Senior Program Committee members wrote supplementary reviews or consulted outside experts to ensure that authors received fair and detailed feedback. We are extremely grateful for their efforts.

Senior Program Committee

Kathryn Laskey	George Mason University
Kevin Leyton-Brown	University of British Columbia
Vikash Mansinghka	Navia Systems
Benjamin Marlin	U. of Massachusetts
Chris Meek	Microsoft Research
Marina Meila	University of Washington
Claire Monteleoni	George Washington University
Joris Mooij	Radboud University Nijmegen
Remi Munos	INRIA Lille
Iain Murray	University of Edinburgh
Petri Myllymaki	Helsinki Institute for Information Technology
Ann Nicholson	Monash University
Thomas Nielsen	Aalborg University
Nuria Oliver	Telefonica
Ronald Parr	Duke University
Avi Pfeffer	Charles River Analytics
David Poole	University of British Columbia
Marc'Aurelio Ranzato	Google
Thomas Richardson	University of Washington
Teemu Roos	Helsinki Institute for Information Technology
Regis Sabbadin	INRA
Christian Shelton	University of California, Riverside
Prakash Shenoy	University of Kansas
Ricardo Silva	University College London
Alex Smola	Yahoo
Padhraic Smyth	UC Irvine
David Sontag	New York University
Peter Spirtes	Carnegie Mellon University
Csaba Szepesvari	University of Alberta
Ben Taskar	University of Pennsylvania
Jin Tian	Iowa State University
Marc Toussaint	FU Berlin
Raquel Urtasun	Toyota Technological Institute at Chicago
S. V. N. Vishwanathan	Purdue University
Michael Wellman	University of Michigan
David Wingate	MIT
Jenn Vaughan	UCLA
Nevin Zhang	HKUST
Alice Zheng	Microsoft Research

Program Committee

Ryan Adams	Pranjal Awasthi	Guillaume Bouchard
Apoorv Agarwal	Francis Bach	Alexandre Bouchard-Cote
Noa Agmon	Maria Florina Balcan	Ronen Brafman
John-Mark Agosta	Elias Bareinboim	Darius Braziunas
Edoardo Airoldi	Peter Bartlett	Gavin Brown
Ayesha Ali	Dhruv Batra	Marcus Brubaker
Russell Almond	Shai Ben-David	Emma Brunskill
Eyal Amir	Salem Benferhat	Wray Buntine
Dragomir Anguelov	Alina Beygelzimer	Tiberio Caetano
Alessandro Antonucci	Bozhena Bidyuk	Cassio de Campos
Dimitrios Antos	Concha Bielza	Peter Carbonetto
Cedric Archambeau	Charles Blundell	Lawrence Carin
Nimar Arora	Antoine Bordes	Francois Caron

Robert Castelo
Hei Chan
Laurent Charlin
Gal Chechik
Minmin Chen
Tao Chen
Max Chickering
Arthur Choi
Anna Choromanska
Tianjiao Chu
Clement Chung
Tom Claassen
Adam Coates
Barry Cobb
Ira Cohen
Paulo Costa
Aaron Courville
James Cussens
Adnan Darwiche
Denver Dash
Richard Dearden
Rina Dechter
Vasil Denchev
Sebastien Destercke
Vanessa Didelez
Francisco Diez
Pedro Domingos
Frederick Eberhardt
Lloyd Elliott
Evan Ettinger
Helene Fargier
Alan Fern
M. Julia Flores
Emily Fox
Roman Garnett
Jan Gasthaus
Mohammad Ghavamzadeh
Angelo Gilio
Mark Girolami
Bob Givan
Amir Globerson
Clark Glymour
Lluis Godo
Vibhav Gogate
Moises Goldszmidt
Manuel Gomez-Olmedo
Christophe Gonzales
Ian Goodfellow
Noah Goodman
Stephen Gould
Amit Gruber
Carlos Guestrin
Michele Guindani
Yuhong Guo
Maya Gupta
Hannaneh Hajishirzi
David Heckerman
Katherine Heller
Tom Heskes
Jesse Hoey
Matt Hoffman
Patrik Hoyer
William Hsu
Bert Huang
Jianhua Huang
Jim Huang
Ferenc Huszar
Marcus Hutter
Alexander Ihler
Tommi Jaakkola
Dominik Janzing
Stefanie Jegelka
David Jensen
Albert Jiang
Nebojsa Jojic
Carl Kadie
Ece Kamar
Ashish Kapoor
Kalev Kask
Yarden Katz
Koray Kavukcuoglu
Kristian Kersting
Sergey Kirshner
Jens Kober
Mikko Koivisto
Nikos Komodakis
Petri Kontkanen
Kevin Korb
Timo Koski
Wojciech Kotlowski
Andreas Krause
Alex Kulesza
Akshat Kumar
Branislav Kveton
Simon Lacoste-Julien
Gert Lanckriet
Jerome Lang
Tobias Lang
Helge Langseth
Hugo Larochelle
Pedro Larranaga
Kathryn Laskey
Quoc Le
Su-In Lee
Jan Lemeire
Philippe Leray
Josh Letchford
Kevin Leyton-Brown
Lihong Li
Yingyu Liang
Jennifer Listgarten
Weiru Liu
Dan Lizotte
Phil Long
Daniel Lowd
Peter Lucas
Laurens van der Maaten
Marloes Maathuis
Omid Madani
Subramani Mani
Vikash Mansinghka
Dimitris Margaritis
Radu Marinescu
Benjamin Marlin
Bhaskara Marthi
Maria Vanina Martinez
Andre Martins
Robert Mateescu
Andrew McCallum
Chris Meek
Marina Meila
Talya Meltzer
Ole Mengshoel
Taneli Mielikainen
Thomas Minka
Andriy Mnih
Volodymyr Mnih
Shakir Mohamed
Claire Monteleoni
Joris Mooij
Sach Mukherjee
Remi Munos
Iain Murray
Petri Myllymaki
Ann Nicholson
Alex Niculescu-Mizil
Thomas Nielsen
Guillaume Obozinski
Nuria Oliver
Luis Ortiz
Michael Osborne
David Page
John Paisley
Ronald Parr
David Pennock
Patrice Perny
Jonas Peters
Nathalie Peyrard
Avi Pfeffer
Kim-Leng Poh
David Poole
Hoifung Poon
Pascal Poupart
Doina Precup
Bob Price

David Pynadath
Yuan Qi
Zengchang Qin
Erik Quaeghebeur
Emmanuel Rachelson
Ali Rahimi
Deva Ramanan
Roland Ramsahai
Marc'Aurelio Ranzato
Vinayak Rao
Silja Renooij
Thomas Richardson
Teemu Roos
Mike Ruberry
Rafael Rumi
Alexander Rush
Daniil Ryabko
Regis Sabbadin
Ardavan Saeedi
Antonio Salmeron
Scott Sanner
Cristina Savin
Mark Schmidt
Jeff Schneider
Dale Schuurmans
Alexander Schwing
Bart Selman
Ross Shachter
Christian Shelton
Prakash Shenoy
Ilya Shpitser
Ricardo Silva
Gerardo Simari
Aarti Singh
Ajit Singh
Tomas Singliar
Arunesh Sinha
Alex Smola
Padhraic Smyth
Richard Socher
Le Song
David Sontag
Peter Spirtes
Ilya Sutskever
Joe Suzuki
Kevin Swersky
Csaba Szepesvari
Daniel Tarlow
Ben Taskar
Graham Taylor
Bo Thiesson
Jin Tian
Robert Tillman
Michalis Titsias
Marc Toussaint
Volker Tresp
Matthias Troffaes
Ryan Turner
Charles Twardy
Raquel Urtasun
Marco Valtorta
Jennifer Wortman Vaughan
Pascal Vincent
S.V.N. Vishwanathan
Yevgeniy Vorobeychik
Chong Wang
Yi Wang
Ziyu Wang
Renata Wassermann
Max Welling
Michael Wellman
Wim Wiegerinck
Sinead Williamson
David Wingate
David Wipf
Dongrui Wu
Changhe Yuan
Xiaotong Yuan
Hyokun Yun
Vincent Zhai
Chongjie Zhang
Kun Zhang
Nevin Zhang
Yu Zhang
Alice Zheng
Onno Zoeter
Masrour Zoghi
Aviv Zohar
Michal Rosen Zvi

Student Volunteers

Bryant Chen
Marwan Mattar
Saeed Amizadeh
Sanmi Koyejo
Sreangsu Acharyya
Udi Apsel
Bo Liu
Andrew Gelfand
Natasha Flerova

Other people who helped

A number of people have made significant contributions towards making UAI 2012 possible. We acknowledge and thank:

- David Heckerman for being local arrangements chair.
- Irina Rish for being tutorial chair.
- Ruslan Salakhutdinov for being workshop chair.
- Prakash Shenoy, for his help with the financial aspects of the conference.
- Teemu Ross, for crucial help in getting support from the Pascal2 Network.
- David Matheson, for his help in various matters, including creating the conference web site, producing the conference booklet, etc.
- Matt Hoffman, for his help in various matters, including the CMT review system, producing the conference proceedings, etc.

- Joelle Pineau, John Langford, and Mahdi Milani Fard for their advice on how to use the CMT paper matching system.
- Suming Chen and Elias Bareinboim, for their help with various matters.
- Laurent Charlin and Richard Zemel for their help with the paper matching process.

Sponsors

Finally, we gratefully acknowledge the generous support provided by our sponsors: Microsoft Research, Google Inc., Artificial Intelligence Journal, Pascal II Network of Excellence, Charles River Analytics, IBM Research.

Invited talks

Babies, brains and Bayes

Alison Gopnik

How do young children learn so much about the world so quickly and accurately? Many researchers have proposed that children implicitly formulate structured hypotheses about the world and then use evidence to test and revise those hypotheses. Ill describe extensive research has shown over the past ten years that even two-year-olds formulate causal hypotheses and test and evaluate them against the data in a normatively accurate way. But this work raises several problems. Where do those hypotheses come from? What algorithms could children implicitly use to approximate ideal Bayesian inference? How can we reconcile childrens striking inferential success with their apparent variability and irrationality? And are there developmental differences in the ways that children and adults learn?

I will suggest that all these questions can be illuminated by thinking about childrens hypothesis generation as a sampling process an idea with a long and successful history in computer science. Preschoolers may use sampling to generate hypotheses, and this may explain both their successful inferences and the variability in their behavior. In fact, in recent research we have discovered that childrens causal learning has some of the signatures of sampling. In particular, the variability in childrens responses reflects the probability of their hypotheses. Moreover, we have shown that children may use a particular type of sampling algorithm. We also found that preschoolers were actually more open-minded learners than older children and adults in some tasks. They seemed more likely to accept a wide range of novel hypotheses. This suggests that they may search at a higher temperature than adults do. From an evolutionary perspective the transition from childhood to adulthood may be natures way of performing simulated annealing.

Never ending learning

Tom Mitchell

We will never really understand machine learning until we can build machines that learn many different things, over years, and become better learners over time.

This talk describes our research to build a Never-Ending Language Learner (NELL) that runs 24 hours per day, forever, learning to read the web. Each day NELL extracts (reads) more facts from the web, and integrates these into its growing knowledge base of beliefs. Each day NELL also learns to read better than yesterday, enabling it to go back to the text it read yesterday, and extract more facts, more accurately. NELL has been running 24 hours/day for over two years now. The result so far is a collection of 15 million interconnected beliefs (e.g., servedWtih(coffee, applePie), isA(applePie, bakedGood)), that NELL is considering at different levels of confidence, along with hundreds of thousands of learned phrasings, morphoogical features, and web page structures that NELL uses to extract beliefs from the web. Track NELL's progress at http://rtw.ml.cmu.edu.

Quantum computing meets machine learning

Bill Macready

Quantum Computing offers the theoretical promise of dramatically faster computation through direct utilization of the underlying quantum aspects of reality. This idea, first proposed in the early 1980s, exploded in interest in 1994 with Peter Shor's discovery of a polynomial time integer factoring algorithm. Today the first experimental platforms realizing small-scale quantum algorithms are becoming commonplace. Interestingly, machine learning may be the killer app for quantum computing. We will introduce quantum algorithms, with focus on a recent quantum computational model that will be familiar to researchers with a background in graphical models. We will show how a particular quantum algorithm – quantum annealing – running on current quantum hardware can be applied to certain optimization problems arising in machine learning. In turn, we will describe a number of challenges to further progress in quantum computing, and suggest that machine learning researchers may be well-positioned to drive the first real-world applications of quantum annealing.

Graphical models for computer vision

Pedro Felzenszwalb

Graphical models provide a powerful framework for expressing and solving a variety of inference problems. The approach has had an enormous impact in computer vision. In this talk I will review some of the developments that have enabled this impact, focusing on efficient algorithms that exploit the structure of vision problems. I will discuss several applications including the low-level vision problem of image restoration, the mid-level problem of segmentation and the high-level problem of model-based recognition. I will also discuss some of the current challenges in the area.

Machine Learning on (Astronomically) Large Datasets

Alexander Gray

There is much talk of big data these days, surrounding a seismic shift currently underway in industry. Much of what is being discussed today is reminiscent of, and perhaps can be informed by, efforts beginning at least two decades ago in astronomy – which was forced into that pioneering position by some unique scientific constraints and motivating problems that we'll discuss. So what are the best ideas that have been developed over the years for doing machine learning on massive datasets? Toward scaling up the most popular textbook machine learning methods across seven basic types of statistical tasks, we'll first identify seven main types of computational bottlenecks. Then we'll consider seven cross-cutting classes of computational techniques which characterize the current fastest algorithms for these bottlenecks, discussing their strengths and weaknesses. We'll end with some exhortations regarding where more foundational research is needed.

Invited and Not-for-publication papers

The *Do*-Calculus Revisited
Judea Pearl
Keynote Lecture, August 17, 2012
UAI-2012 Conference, Catalina, CA

Abstract

The *do*-calculus was developed in 1995 to facilitate the identification of causal effects in non-parametric models. The completeness proofs of [Huang and Valtorta, 2006] and [Shpitser and Pearl, 2006] and the graphical criteria of [Tian and Shpitser, 2010] have laid this identification problem to rest. Recent explorations unveil the usefulness of the *do*-calculus in three additional areas: mediation analysis [Pearl, 2012], transportability [Pearl and Bareinboim, 2011] and meta-synthesis. Meta-synthesis (freshly coined) is the task of fusing empirical results from several diverse studies, conducted on heterogeneous populations and under different conditions, so as to synthesize an estimate of a causal relation in some target environment, potentially different from those under study. The talk surveys these results with emphasis on the challenges posed by meta-synthesis. For background material, see ⟨http://bayes.cs.ucla.edu/csl_papers.html⟩.

1 Introduction

Assuming readers are familiar with the basics of graphical models, I will start by reviewing the problem of nonparametric identification and how it was solved by the *do*-calculus and its derivatives. I will then show how the *do*-calculus benefits mediation analysis (Section 2) , transportability problems (Section 3) and meta-synthesis (Section 4).

1.1 Causal Models, interventions, and Identification

Definition 1. *(Structural Equation Model) [Pearl, 2000, p. 203].*
A structural equation model (SEM) M is defined as follows:

1. *A set U of background or exogenous variables, representing factors outside the model, which nevertheless affect relationship within the model.*
2. *A set $V = \{V_1, \ldots, V_n\}$ of observed endogenous variables, where each V_i is functionally dependent on a subset PA_i of $U \cup V \backslash \{V_i\}$.*
3. *A set F of functions $\{f_1, \ldots, f_n\}$ such that each f_i determines the value of $V_i \in V, v_i = f_i(pa_i, u)$.*
4. *A joint probability distribution $P(u)$ over U.*

When an instantiation $U = u$ is given, together with F, the model is said to be "completely specified" (at the unit level) when the pair $\langle P(u), F\rangle$ is given, the model is "fully specified" (at the population level). Each fully specified model defines a *causal diagram* G in which an arrow is drawn towards V_i from each member of its parent set PA_i.

Interventions and counterfactuals are defined through a mathematical operator called $do(x)$, which simulates physical interventions by deleting certain functions from the model, replacing them with a constant $X = x$, while keeping the rest of the model unchanged. The resulting model is denoted M_x.

The postintervention distribution resulting from the action $do(X = x)$ is given by the equation

$$P_M(y|do(x)) = P_{M_x}(y) \tag{1}$$

In words, in the framework of model M, the postintervention distribution of outcome Y is defined as the probability that model M_x assigns to each outcome level $Y = y$. From this distribution, which is readily computed from any fully specified model M, we are able to assess treatment efficacy by comparing aspects of this distribution at different levels of x. Counterfactuals are defined similarly through the equation $Y_x(u) = Y_{M_x}(u)$ (see [Pearl, 2009, Ch. 7]).

The following definition captures the requirement that a causal query Q be estimable from the data:

Definition 2. *(Identifiability) [Pearl, 2000, p. 77]. A causal query $Q(M)$ is identifiable, given a set of assumptions A, if for any two (fully specified) models M_1 and M_2 that satisfy A, we have*

$$P(M_1) = P(M_2) \Rightarrow Q(M_1) = Q(M_2) \qquad (2)$$

In words, the functional details of M_1 and M_2 do not matter; what matters is that the assumptions in A (e.g., those encoded in the diagram) would constrain the variability of those details in such a way that equality of Ps would entail equality of Qs. When this happens, Q depends on P only, and should therefore be expressible in terms of the parameters of P.

1.2 The Rules of *do*-calculus

When a query Q is given in the form of a *do*-expression, for example $Q = P(y|do(x), z)$, its identifiability can be decided systematically using an algebraic procedure known as the *do*-calculus [Pearl, 1995]. It consists of three inference rules that permit us to map interventional and observational distributions whenever certain conditions hold in the causal diagram G.

Let X, Y, Z, and W be arbitrary disjoint sets of nodes in a causal DAG G. We denote by $G_{\overline{X}}$ the graph obtained by deleting from G all arrows pointing to nodes in X. Likewise, we denote by $G_{\underline{X}}$ the graph obtained by deleting from G all arrows emerging from nodes in X. To represent the deletion of both incoming and outgoing arrows, we use the notation $G_{\overline{X}\underline{Z}}$.

The following three rules are valid for every interventional distribution compatible with G.

Rule 1 (Insertion/deletion of observations):

$$P(y|do(x), z, w) = P(y|do(x), w) \quad \text{if } (Y \perp\!\!\!\perp Z|X, W)_{G_{\overline{X}}} \qquad (3)$$

Rule 2 (Action/observation exchange):

$$P(y|do(x), do(z), w) = P(y|do(x), z, w) \quad \text{if } (Y \perp\!\!\!\perp Z|X, W)_{G_{\overline{X}\underline{Z}}} \qquad (4)$$

Rule 3 (Insertion/deletion of actions):

$$P(y|do(x), do(z), w) = P(y|do(x), w) \quad \text{if } (Y \perp\!\!\!\perp Z|X, W)_{\overline{XZ(W)}}, \qquad (5)$$

where $Z(W)$ is the set of Z-nodes that are not ancestors of any W-node in $G_{\overline{X}}$.

To establish identifiability of a query Q, one needs to repeatedly apply the rules of *do*-calculus to Q, until the final expression no longer contains a *do*-operator[1]; this renders it estimable from non-experimental data, and the final *do*-free expression can serve as an *estimator* of Q. The *do*-calculus was proven to be complete to the identifiability of causal effects [Shpitser and Pearl, 2006; Huang and Valtorta, 2006], which means that if the *do*-operations cannot be removed by repeated application of these three rules, Q is not identifiable.

Parallel works by [Tian and Pearl, 2002] and [Shpitser and Pearl, 2006] have led to graphical criteria for verifying the identifiability of Q as well as polynomial time algorithms for constructing an estimator of Q. This, from a mathematical viewpoint, closes the chapter of nonparametric identification of causal effects.

2 Using *do*-Calculus for Identifying Direct and Indirect Effects

Consider the mediation model in Fig. 1(a). (In this section, M stands for the mediating variable.) The *Controlled Direct Effect* (CDE) of X on Y is defined as

$$CDE(m) = E(Y|do(X = 1, M = m)) - E(Y|do(X = 0, M = m))$$

and the *Natural Direct Effect* (NDE) is defined by the counterfactual expression

$$NDE = E(Y_{X=1, M_{X=0}}) - E(Y_{X=0})$$

The natural direct effect represents the effect transmitted from X and Y while keeping some intermediate variable M at whatever level it attained prior to the transition [Robins and Greenland, 1992; Pearl, 2001].

Since CDE is a *do*-expression, its identification is fully characterized by the *do*-calculus. The NDE, however, is counterfactual and requires more intricate conditions, as shown in Pearl (2001). When translated to graphical language these conditions read:

Assumption-Set A [Pearl, 2001]

There exists a set W of measured covariates such that:

A-1 No member of W is a descendant of X.

A-2 W blocks all back-door paths from M to Y, disregarding the one through X.

A-3 The W-specific effect of X on M is identifiable using *do*-calculus.

[1] Such derivations are illustrated in graphical details in [Pearl, 2009, p. 87].

A-4 The W-specific joint effect of $\{X, M\}$ on Y is identifiable using *do*-calculus.

The bulk of the literature on mediation analysis has chosen to express identification conditions in the language of "ignorability" (i.e., independence among counterfactuals) which is rather opaque and has led to significant deviation from assumption set A.

A typical example of overly stringent conditions that can be found in the literature reads as follows:

> "Imai, Keele and Yamamoto (2010) showed that the sequential ignorability assumption must be satisfied in order to identify the average mediation effects. This key assumption implies that the treatment assignment is essentially random after adjusting for observed pretreatment covariates and that the assignment of mediator values is also essentially random once both observed treatment and the same set of observed pretreatment covariates are adjusted for." [Imai, Jo, and Stuart, 2011]

When translated to graphical representation, these conditions read:

Assumption-Set B

There exists a set W of measured covariates such that:

B-1 No member of W is a descendant of X.

B-2 W blocks all back-door paths from X to M.

B-3 W and X block all back-door paths from M to Y.

We see that assumption set A relaxes that in B in two ways.

First, we need not insist on using "the same set of observe pretreatment covariates," two separate sets can sometimes accomplish what the same set does not. Second, conditions A-3 and A-4 invoke *do*-calculus and thus open the door for identification criteria beyond back-door adjustment (as in B-2 and B-3).

In the sequel we will show that these two features endow A with greater identification power. (See also [Shpitser, 2012].)

2.1 Divide and conquer

Fig. 2 demonstrates how the "divide and conquer" flexibility translates into an increase identification power. Here, the $X \rightarrow M$ relationship requires an adjustment

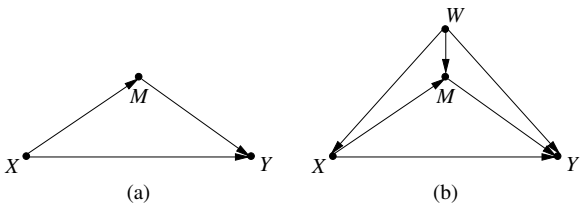


Figure 1: (a) The basic unconfounded mediation model, showing the treatment (X) mediator (M) and outcome (Y). (b) The mediator model with an added covariate (W) that confounds both the $X \rightarrow M$ and the $M \rightarrow Y$ relationships.

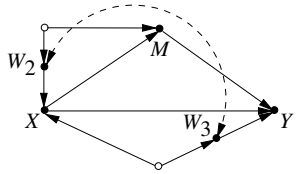


Figure 2: A mediation model with two dependent confounders, permitting the decomposition of Eq. (6). Hollow circles stand for unmeasured confounders. The model satisfies condition A but violates condition B.

for W_2, and the $X \rightarrow Y$ relationship requires an adjustment for W_3. If we make the two adjustments separately, we can identify NDE by the estimand:

$$NDE = \sum_m \sum_{w_2, w_3} P(W_2 = w_2, W_3 = w_3) \\ [E(Y \mid X = 1, M = m, W_3 = w_3)] \\ - [E(Y \mid X = 0, M = m, W_3 = w_3,)] \\ P(M = m \mid X = 0, W_2 = w_2) \qquad (6)$$

However, if we insist on adjusting for W_2 and W_3 simultaneously, as required by assumption set B, the $X \rightarrow M$ relationship would become confounded, by opening two colliders in tandem along the path $X \rightarrow W_3 \leftrightarrow W_2 \leftarrow M$. As a result, assumption set B would deem the NDE to be unidentifiable; there is no covariates set W that simultaneously deconfounds the two relationships.

2.2 Going beyond back-door adjustment

Figure 3 displays a model for which the natural direct effect achieves its identifiability through multi-step adjustment (in this case using the front-door procedure), permitted by A, though not through a single-step adjustment, as demanded by B. In this model, the null set $W = \{0\}$ satisfies conditions B-1 and B-3, but not condition B-2; there is no set of covariates that would enable us to deconfound the treatment-mediator relationship. Fortunately, condition A-3 requires only

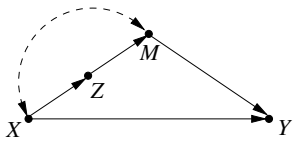


Figure 3: Measuring Z permits the identification of the effect of X on M through the front-door procedure.

that we identify the effect of X on M by *some* do-calculus method, not necessarily by rendering X random or unconfounded (or ignorable). The presence of observed variable Z permits us to identify this causal effect using the front-door condition [Pearl, 1995; 2009].

In Fig. 4, the front-door estimator needs to be applied to both the $X \rightarrow M$ and the $X \rightarrow Y$relationships. In addition, conditioning on W is necessary, in order to satisfy condition A-2. Still, the identification of $P(m|do(x), w)$ and $E(Y|do(m, x), w)$ presents no special problems to students of causal inference [Shpitser and Pearl, 2006].

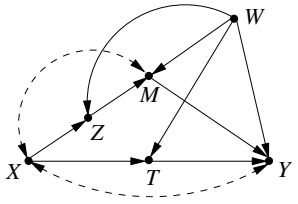


Figure 4: Measuring Z and T permits the identification of the effect of X on M and X on Y for each specific w and leads to the identification of the natural direct effect.

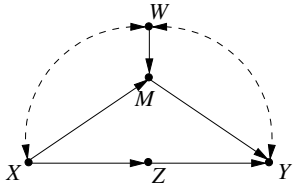


Figure 5: NDE is identified by adjusting for W and using Z to deconfound the $X \rightarrow Y$ relationship.

Figure 5, demonstrates the role that an observed covariate (Z) on the $X \rightarrow Y$ pathway can play in the identification of natural effects. In this model, conditioning on W deconfounds both the $M \rightarrow Y$ and $Y \rightarrow M$ relationships but confounds the $X \rightarrow Y$ relationship. However the W-specific joint effect of $\{X, M\}$ on Y is identifiable through observations on Z (using the front-door estimand).

In Fig. 6, a covariate Z situated along the path from M to Y leads to identifying NDE. Here the mediator$\rightarrow$ outcome relationship is unconfounded (once we fix X), so, we are at liberty to choose $W = \{0\}$ to satisfy condition A-2. The treatment$\rightarrow$ mediator relationship is confounded, and requires an adjustment for T (so does the treatment-outcome relationship). However, conditioning on T will confound the $\{MX\} \rightarrow Y$ relationship (in violation of condition A-4). Here, the presence of Z comes to our help, for it permits us to estimate $P(Y \mid do(x, m), t)$ thus rendering NDE identifiable.

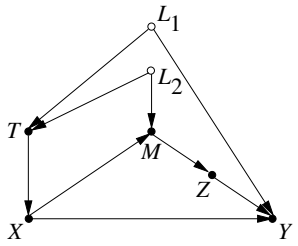


Figure 6: The confounding created by adjusting for T can be removed using measurement of Z.

3 Using *do*-Calculus to Decide Transportability

In applications involving identifiability, the role of the *do*-calculus is to remove the *do*-operator from the query expression. We now discuss a totally different application, to decide if experimental findings can be transported to a new, potentially different environment, where only passive observations can be perforemed. This problem, labeled "transportability" in [Pearl and Bareinboim, 2011] can also be reduced to syntactic operation using the *do*-calculus but here the aim will be to separate the *do*-operator from a set S of variables, that indicate disparities between the two environment.

We shall motivate the problem through the following three examples.

Example 1. *We conduct a randomized trial in Los Angeles (LA) and estimate the causal effect of exposure X on outcome Y for every age group $Z = z$ as depicted in Fig. 7(a). We now wish to generalize the results to the population of New York City (NYC), but data alert us to the fact that the study distribution $P(x, y, z)$ in LA is significantly different from the one in NYC (call the latter $P^*(x, y, z)$). In particular, we notice that the average age in NYC is significantly higher than that in LA. How are we to estimate the causal effect of X on Y in NYC, denoted $P^*(y|do(x))$.*

Example 2. *Let the variable Z in Example 1 stand for subjects language proficiency, and let us assume that*

Z does not affect exposure (X) or outcome (Y), yet it correlates with both, being a proxy for age which is not measured in either study (see Fig. 7(b)). Given the observed disparity $P(z) \neq P^*(z)$, how are we to estimate the causal effect $P^*(y|do(x))$ for the target population of NYC from the z-specific causal effect $P(y|do(x), z)$ estimated at the study population of LA?

Example 3. *Examine the case where Z is a X-dependent variable, say a disease bio-marker, standing on the causal pathways between X and Y as shown in Fig. 7(c). Assume further that the disparity $P(z) \neq P^*(z)$ is discovered in each level of X and that, again, both the average and the z-specific causal effect $P(y|do(x), z)$ are estimated in the LA experiment, for all levels of X and Z. Can we, based on information given, estimate the average (or z-specific) causal effect in the target population of NYC?*

To formalize problems of this sort, Pearl and Bareinboim devised a graphical representation called "selection diagrams" which encodes knowledge about differences between populations. It is defined as follows:

Definition 3. *(Selection Diagram).*
Let $\langle M, M^* \rangle$ be a pair of structural causal models (Definition 1) relative to domains $\langle \Pi, \Pi^* \rangle$, sharing a causal diagram G. $\langle M, M^* \rangle$ is said to induce a selection diagram D if D is constructed as follows:

1. *Every edge in G is also an edge in D;*
2. *D contains an extra edge $S_i \rightarrow V_i$ whenever there exists a discrepancy $f_i \neq f_i^*$ or $P(U_i) \neq P^*(U_i)$ between M and M^*.*

In summary, the S-variables locate the mechanisms where structural discrepancies between the two populations are suspected to take place. Alternatively, the absence of a selection node pointing to a variable represents the assumption that the mechanism responsible for assigning value to that variable is the same in the two populations, as shown in Fig. 7.

Using selection diagrams as the basic representational language, and harnessing the concepts of intervention, *do*-calculus, and identifiability (Section 1), we can now give the notion of transportability a formal definition.

Definition 4. *(Transportability)*
Let D be a selection diagram relative to domains $\langle \Pi, \Pi^* \rangle$. Let $\langle P, I \rangle$ be the pair of observational and interventional distributions of Π, and P^* be the observational distribution of Π^*. The causal relation $R(\Pi^*) = P^*(y|do(x), z)$ is said to be transportable from Π to Π^* in D if $R(\Pi^*)$ is uniquely computable from P, P^*, I in any model that induces D.

Theorem 1. *Let D be the selection diagram characterizing two populations, Π and Π^*, and S a set of selection variables in D. The relation $R = P^*(y|do(x), z)$ is transportable from Π to Π^* if the expression $P(y|do(x), z, s)$ is reducible, using the rules of do-calculus, to an expression in which S appears only as a conditioning variable in do-free terms.*

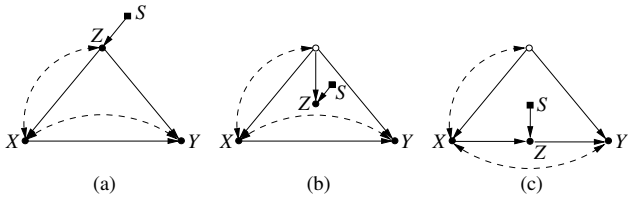


Figure 7: Selection diagrams depicting Examples 1–3. In (a) the two populations differ in age distributions. In (b) the populations differs in how Z depends on age (an unmeasured variable, represented by the hollow circle) and the age distributions are the same. In (c) the populations differ in how Z depends on X.

This criterion was proven to be both sufficient and necessary for causal effects, namely $R = P(y|do(x))$ [Bareinboim and Pearl, 2012b]. Theorem 1 does not specify the sequence of rules leading to the needed reduction when such a sequence exists. [Bareinboim and Pearl, 2012b] established a complete and effective graphical procedure of confirming transportability which also synthesizes the transport formula whenever possible. For example, the transport formulae derived for the three models in Fig. 7 are (respectively):

$$P^*(y|do(x)) = \sum_z P(y|do(x), z)P^*(z) \tag{7}$$

$$P^*(y|do(x)) = p(y|do(x)) \tag{8}$$

$$P^*(y|do(x)) = \sum_z P(y|z, x)P^*(z|x) \tag{9}$$

Each transport formula determines for the investigator what information need to be taken from the experimental and observational studies and how they ought to be combined to yield an unbiased estimate of R.

4 From "Meta-analysis" to "Meta-synthesis"

"Meta analysis" is a data fusion problem aimed at combining results from many experimental and observational studies, each conducted on a different population and under a different set of conditions, so as to synthesize an aggregate measure of effect size that is "better," in some sense, than any one study in isolation. This fusion problem has received enormous attention in the health and social sciences, where data are scarce and experiments are costly.

Unfortunately, current techniques of meta-analysis do little more than take weighted averages of the various studies, thus averaging apples and oranges to infer properties of bananas. One should be able to do better. Using "selection diagrams" to encode commonalities among studies, we should be able to "synthesize" an estimator that is guaranteed to provide unbiased estimate of the desired quantity based on information that each study share with the target environment. The basic idea is captured in the following definition and theorem.

Definition 5. *(Meta-identifiability):*
A relation R is said to be "meta-identifiable" from a set of populations $(\Pi_1, \Pi_2, \ldots, \Pi_K)$ to a target population Π^* iff it is identifiable from the information set $I = \{I(\Pi_1), I(\Pi_2), \ldots, I(\Pi_K), I(\Pi^*)\}$, where $I(\Pi_k)$ stands for the information provided by population Π_k.

Theorem 2. *(Meta-identifiability):*
Given a set of studies $\{\Pi_1, \Pi_2, \ldots, \Pi_K\}$ characterized by selection diagrams $\{D_1, D_2, \ldots, D_K\}$ relative to a target population Π^*, a relation $R(\Pi^*)$ is "meta-identifiable" if it can be decomposed into a set of subrelations of the form:

$$R_k = P(V_k|do(W_k), Z_k) \qquad k = 1, 2, \ldots, K$$

such that each R_k is transportable from some D_k.

Theorem 2 reduces the problem of Meta synthesis to a set of transportability problems, and calls for a systematic way of decomposing R to fit the information provided by I.

Exemplifying meta-synthesis

Consider the diagrams depicted in Fig. 8, each representing a study conducted on a different population and under a different set of conditions. Solid circles represent variables that were measured in the respective study and hollow circles variables that remained unmeasured. An arrow ■→ represents an external influence affecting a mechanism by which the study population is assumed to differ from the target population Π^*, shown in Fig. 8(a). For example, Fig. 8(c) represents an observational study on population Π_c in which variables X, Z and Y were measured, W was not measured and the prior probability $P_c(z)$ differs from that of the target population $P^*(z)$. Diagrams (b)–(f) represent observational studies while (g)–(j) stand for experimental studies with X randomized (hence the missing arrows into X).

Despite differences in populations, measurements and conditions, each of the studies may provide information that bears on the target relation $R(\Pi^*)$ which, in this example, we take to be the causal effect of X on Y, $P^*(y|do(x))$ or; given the structure of Fig. 8(a),

$$R(\Pi^*) = P^*(y|do(x)) = \sum_z P^*(y|x,z)P^*(z).$$

While $R(\Pi^*)$ can be estimated directly from some of the studies, (e.g., (g)) or indirectly from others (e.g., (b) and (d)), it cannot be estimated from those studies in which the population differs substantially from Π^*, (e.g., (c), (e), (f)). The estimates of R provided by the former studies may differ from each other due to sampling variations and measurement errors, and can be aggregated in the standard tradition of meta analysis. The latter studies, however, should not be averaged with the former, since they do not provide unbiased estimates of R. Still, they are not totally useless, for they can provide information that renders the former estimates more accurate. For example, although we cannot identify R from study 8(c), since $P_c(z)$ differs from the unknown $P^*(z)$, we can nevertheless use the estimates of $P_c(x|z), P_c(y|z,x)$ that 8(c) provides to improve the accuracy of $P^*(x|z)$ and $P^*(y|z,x)$[2] which may be needed for estimating R by indirect meth-

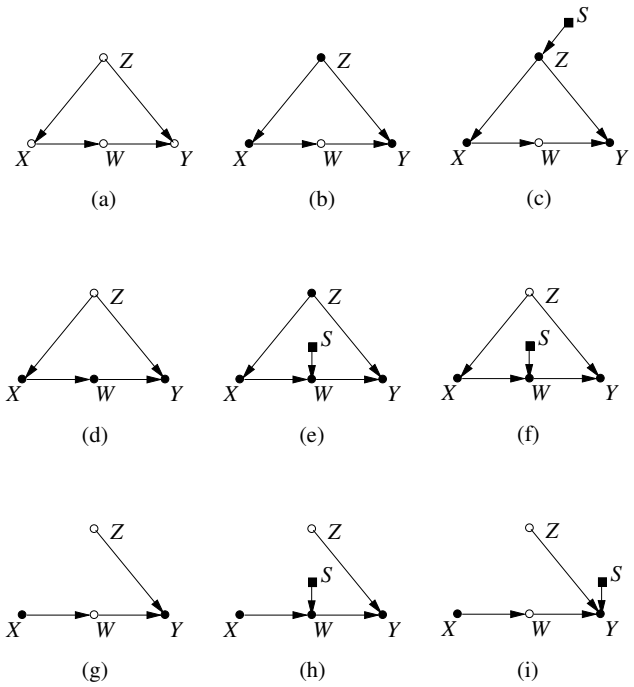


Figure 8: Diagrams representing 8 studies ((b)–(i)) conducted under different conditions on different populations, aiming to estimate the causal effect of X on Y in the target population, shown in 8(a).

[2]The absence of boxed arrows into X and Y in Fig. 8(c) implies the equalities

$$P_c(x|z) = P^*(x|z) \text{ and } P_c(y|z,x) = P^*(y|z,x).$$

ods. For example, $P^*(y|z,x)$ is needed in study 8(b) if we use the estimator $R = \sum_z P^*(y|x,z)P^*(z)$, while $P^*(x|z)$ is needed if we use the inverse probability estimator $R = \sum_z P^*(x,y,z)/P^*(x|z)$.

Similarly, consider the randomized studies depicted in 8(h) and 8(i). None is sufficient for identifying R in isolation yet, taken together, they permit us to borrow $P_i(w|do(x))$ from 8(i) and $P_h(y|w,do(x))$ from 8(h) and synthesize a bias-free estimator:

$$\begin{aligned} R &= \sum_w P^*(y|w,do(x))P^*(w|do(x)) \\ &= \sum_w P_h(y|w,do(x)P_i(w|do(x)) \end{aligned}$$

The challenge of meta synthesis is to take a collection of studies, annotated with their respective selection diagrams (as in Fig. 8), and construct an estimator of a specified relation $R(\Pi^*)$ that makes maximum use of the samples available, by exploiting the commonalities among the populations studied and the target population Π^*. As the relation $R(\Pi^*)$ changes, the synthesis strategy will change as well.

It is hard not to speculate that data-pooling strategies based on the principles outlined here will one day replace the blind methods currently used in meta analysis.

Knowledge-guided Domain Adaptation

It is commonly assumed that causal knowledge is necessary only when interventions are contemplated and that in purely predictive tasks, probabilistic knowledge suffices. When dealing with generalization across domains, however, causal knowledge can be valuable, and in fact necessary even in predictive or classification tasks.

The idea is simple; causal knowledge is essentially knowledge about the mechanisms that remain *invariant* under change. Suppose we learn a probability distribution $P(x,y,z)$ in one environment and we ask how this probability would change when we move to a new environment that differs slightly from the former. If we have knowledge of the causal mechanism generating P, we could then represent where we suspect the change to occur and channel all our computational resources to re-learn the local relationship that has changed or is likely to have changed, while keeping all other relationships invariant.

For example, assume that $P^*(x,y,z)$ is the probability distribution in the target environment and our interest lies in estimating $P^*(x|z)$ knowing that the causal diagram behind P is the chain $X \to Y \to Z$, and that the process determining Y, represented by $P^*(y|x)$, is the only one that changed. We can simply re-learn $P^*(y|x)$ and estimate our target relation $P^*(x|z)$ without measuring Z in the new environment. (This is done using $P^*(x,y,z) = P(x)P^*(y|x)P(z|y)$ with the first and third terms transported from the source environment.)

In complex problems, the savings gained by focusing on only a small subset of variables in P^* can be enormous, because any reduction in the number of measured variables translates into substantial reduction in the number of samples needed to achieve a given level of prediction accuracy.

As can be expected, for a given transported relation, the subset of variables that can be ignored in the target environment is not unique, and should therefore be chosen so as to minimize both measurement costs and sampling variability. This opens up a host of new theoretical questions about transportability in causal graphs. For example, deciding if a relation is transportable when we forbid measurement of a given subset of variables in P^* [Pearl and Bareinboim, 2011], or deciding how to pool studies optimally when sample size varies drastically from study to study [Pearl, 2012].

Conclusions

The *do*-calculus, which originated as a syntactic tool for identification problems, was shown to benefit three new areas of investigation. Its main power lies in reducing to syntactic manipulations complex problems concerning the estimability of causal relations under a variety of conditions.

In Section 2 we demonstrated that going beyond standard adjustment for covariates, and unleashing the full power of *do*-calculus, can lead to improved identification power of natural direct and indirect effects. Section 3 demonstrated how questions of transportability can be reduced to symbolic derivations in the *do*-calculus, yielding graph-based procedures for deciding whether causal effects in the target population can be inferred from experimental findings in the study population. When the answer is affirmative, the procedures further identify what experimental and observational findings need be obtained from the two populations, and how they can be combined to ensure bias-free transport.

In a related problem, Bareinboim and Pearl (2012b) show how the *do*-calculus can be used to decide whether the effect of X on Y can be estimated from experiments on a different set, Z, that is more accessible

to manipulations. Here the aim is to transform *do*-expressions to sentences that invoke only $do(z)$ symbols.

Finally, in Section 4, we tackled the problem of data fusion and showed that principled fusion (which we call meta-synthesis) can be reduced to a sequence of syntactic operations each involving a local transportability exercise. This task leaves many questions unsettled, because of the multiple ways a give relation can be decomposed.

Acknowledgments

This research was supported in parts by grants from NIH #1R01 LM009961-01, NSF #IIS-1018922, and ONR #N000-14-09-1-0665 and #N00014-10-1-0933.

References

[Bareinboim and Pearl, 2012a] Bareinboim, E., and Pearl, J. 2012a. Causal inference by surrogate experiments: z-identifiability. Technical Report R-397, Cognitive Systems Laboratory, Department of Computer Science, UCLA. To appear in *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence* (UAI), 2012.

[Bareinboim and Pearl, 2012b] Bareinboim, E., and Pearl, J. 2012b. Transportability of causal effects: Completeness results. Technical Report R-390, Cognitive Systems Laboratory, Department of Computer Science, UCLA. To appear in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence* (AAAI), 2012.

[Huang and Valtorta, 2006] Huang, Y., and Valtorta, M. 2006. Pearl's calculus of intervention is complete. In Dechter, R., and Richardson, T., eds., *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*. Corvallis, OR: AUAI Press. 217–224.

[Imai, Jo, and Stuart, 2011] Imai, K.; Jo, B.; and Stuart, E. A. 2011. Commentary: Using potential outcomes to understand causal mediation analysis. *Multivariate Behavioral Research* 46:842–854.

[Imai, Keele, and Yamamoto, 2010] Imai, K.; Keele, L.; and Yamamoto, T. 2010. Identification, inference, and sensitivity analysis for causal mediation effects. *Statistical Science* 25(1):51–71.

[Pearl and Bareinboim, 2011] Pearl, J., and Bareinboim, E. 2011. Transportability of causal and statistical relations: A formal approach. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI-11)*, 247–254. Menlo Park, CA: AAAI Press. <http://ftp.cs.ucla.edu/pub/stat_ser/r372a.pdf>.

[Pearl, 1995] Pearl, J. 1995. Causal diagrams for empirical research. *Biometrika* 82(4):669–710.

[Pearl, 2000] Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. New York: Cambridge University Press. Second ed., 2009.

[Pearl, 2001] Pearl, J. 2001. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann. 411–420.

[Pearl, 2009] Pearl, J. 2009. *Causality: Models, Reasoning, and Inference*. New York: Cambridge University Press, second edition.

[Pearl, 2012] Pearl, J. 2012. Some thoughts concerning transfer learning, with applications to meta-analysis and data-sharing estimation. Technical Report R-387, Cognitive Systems Laboratory, Department of Computer Science, UCLA.

[Robins and Greenland, 1992] Robins, J., and Greenland, S. 1992. Identifiability and exchangeability for direct and indirect effects. *Epidemiology* 3(2):143–155.

[Shpitser and Pearl, 2006] Shpitser, I., and Pearl, J. 2006. Identification of joint interventional distributions in recursive semi-Markovian causal models. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press. 1219–1226.

[Shpitser, 2012] Shpitser, I. 2012. Counterfactual graphical models for mediation analysis via path-specific effects. Technical report, Harvard University, MA.

[Tian and Pearl, 2002] Tian, J., and Pearl, J. 2002. A general identification condition for causal effects. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press/The MIT Press. 567–573.

[Tian and Shpitser, 2010] Tian, J., and Shpitser, I. 2010. On identifying causal effects. In Dechter, R.; Geffner, H.; and Halpern, J., eds., *Heuristics, Probability and Causality: A Tribute to Judea Pearl*. UK: College Publications. 415–444.

PAC-Bayesian Inequalities for Martingales

Yevgeny Seldin[1,4] **François Laviolette**[2] **Nicolò Cesa-Bianchi**[3] **John Shawe-Taylor**[4] **Peter Auer**[5]
[1]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[2]Département d'informatique, Université Laval, Québec, Canada
[3]Dipartimento di Informatica, Università degli Studi di Milano, Italy
[4]Department of Computer Science, University College London, UK
[5]Chair for Information Technology, Montanuniversität Leoben, Austria
seldin@tuebingen.mpg.de, francois.laviolette@ift.ulaval.ca, nicolo.cesa-bianchi@unimi.it,
jst@cs.ucl.ac.uk, auer@unileoben.ac.at

Abstract

We present a set of high-probability inequalities that control the concentration of weighted averages of multiple (possibly uncountably many) simultaneously evolving and interdependent martingales. Our results extend the PAC-Bayesian analysis in learning theory from the i.i.d. setting to martingales opening the way for its application in reinforcement learning and other interactive learning domains, as well as many other domains in probability theory and statistics, where martingales are encountered.

We also present a comparison inequality that bounds the expectation of a convex function of a martingale difference sequence shifted to the $[0, 1]$ interval by the expectation of the same function of independent Bernoulli variables. This inequality is applied to derive a tighter analog of Hoeffding-Azuma's inequality.

For the complete paper see Seldin et al. (2012).

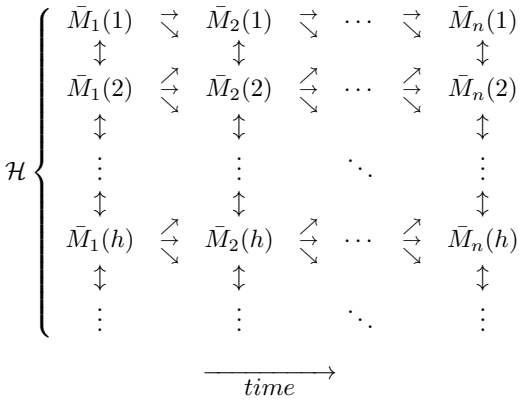


Figure 1: Illustration of an infinite set of simultaneously evolving and interdependent martingales. For a fixed h, the sequence $\bar{M}_1(h), \bar{M}_2(h), \ldots, \bar{M}_n(h)$ is a martingale. $\mathcal{H}$ is a set (possibly uncountably infinite) that indexes the individual martingales. The arrows represent the dependencies between the values of the martingales: the value of the h-th martingale at time i, denoted by $\bar{M}_i(h)$, depends on $\bar{M}_j(h')$ for all $j \leq i$ and $h' \in \mathcal{H}$ (everything that is "before" and "concurrent" with $\bar{M}_i(h)$ in time; some of the arrows are omitted for clarity). A mean value of the martingales with respect to a distribution ρ over $\mathcal{H}$ (or a probability density function, if $\mathcal{H}$ is uncountably infinite) is given by $\langle \bar{M}_n, \rho \rangle$. Our high-probability inequalities bound $|\langle \bar{M}_n, \rho \rangle|$ simultaneously for a large class of ρ.

References

Yevgeny Seldin, François Laviolette, Nicolò Cesa-Bianchi, John Shawe-Taylor, and Peter Auer. PAC-Bayesian inequalities for martingales. *IEEE Transactions on Information Theory*, 2012. Accepted. Preprint available at http://arxiv.org/abs/1110.6886.

Nested Markov Properties for Acyclic Directed Mixed Graphs

Thomas S. Richardson
thomasr@u.washington.edu

James M. Robins
robins@hsph.harvard.edu

Ilya Shpitser
ishpitse@hsph.harvard.edu

Abstract

Directed acyclic graph (DAG) models may be characterized in four different ways: via a factorization, the d-separation criterion, the moralization criterion, and the local Markov property. As pointed out by Robins [2, 1], Verma and Pearl [6], and Tian and Pearl [5], marginals of DAG models also imply equality constraints that are not conditional independences. The well-known 'Verma constraint' is an example. Constraints of this type were used for testing edges [3], and an efficient variable elimination scheme [4]. Using acyclic directed mixed graphs (ADMGs) we provide a graphical characterization of the constraints given in [5] via a nested Markov property that uses a 'fixing' transformation on graphs. We give four characterizations of our nested model that are analogous to those given for DAGs. We show that marginal distributions of DAG models obey this property.

References

[1] James M. Robins. Testing and estimation of direct effects by reparameterizing directed acyclic graphs with structural nested models. In C. Glymour and G. Cooper, editors, *Computation, Causation, and Discovery*, pages 349–405. MIT Press, Cambridge, MA, 1999.

[2] J.M. Robins. A new approach to causal inference in mortality studies with sustained exposure periods – application to control of the healthy worker survivor effect. *Mathematical Modeling*, 7:1393–1512, 1986.

[3] Ilya Shpitser, Thomas S. Richardson, and James M. Robins. Testing edges by truncations. In *International Joint Conference on Artificial Intelligence*, volume 21, pages 1957–1963, 2009.

[4] Ilya Shpitser, Thomas S. Richardson, and James M. Robins. An efficient algorithm for computing interventional distributions in latent variable causal models. In *27th Conference on Uncertainty in Artificial Intelligence (UAI-11)*. AUAI Press, 2011.

[5] Jin Tian and Judea Pearl. On the testable implications of causal models with hidden variables. In *Proceedings of UAI-02*, pages 519–527, 2002.

[6] T. S. Verma and Judea Pearl. Equivalence and synthesis of causal models. Technical Report R-150, Department of Computer Science, University of California, Los Angeles, 1990.

Proceedings

Learning to Rank With Bregman Divergences and Monotone Retargeting

Sreangsu Acharyya *
Dept. Electrical Engineering
University of Texas Austin

Oluwasanmi Koyejo*
Dept. Electrical Engineering
University of Texas Austin

Joydeep Ghosh
Dept. Electrical Engineering
University of Texas Austin

Abstract

This paper introduces a novel approach for learning to rank (LETOR) based on the notion of monotone retargeting. It involves minimizing a divergence between all monotonic increasing transformations of the training scores and a parameterized prediction function. The minimization is both over the transformations as well as over the parameters. It is applied to Bregman divergences, a large class of "distance like" functions that were recently shown to be the unique class that is statistically consistent with the normalized discounted gain (NDCG) criterion [19]. The algorithm uses alternating projection style updates, in which one set of simultaneous projections can be computed independent of the Bregman divergence and the other reduces to parameter estimation of a generalized linear model. This results in easily implemented, efficiently parallelizable algorithm for the LETOR task that enjoys global optimum guarantees under mild conditions. We present empirical results on benchmark datasets showing that this approach can outperform the state of the art NDCG consistent techniques.

1 Introduction

Structured output space models [1] have dominated the task of learning to rank (LETOR). Regression based models have been justifiably superseded by pairwise models [11], which in turn are being gradually displaced by list-wise approaches [6, 17]. This trend has on one hand greatly improved the quality of the predictions obtained but on the other hand has come at the cost of additional complexity and computation. The cost functions of structured models are often defined directly on the combinatorial space of permutations, which significantly increase the difficulty of learning and optimization compared to regression based approaches. We propose an approach to the LETOR task that retains the simplicity of the regression based models, is simple to implement, is embarrassingly parallelizable, and yet is a function of ordering alone. Furthermore, MR enjoys strong guarantees of convergence, statistical consistency under uncertainty and a global minimum under mild conditions. Our experiments on benchmark datasets show that the proposed approach outperforms state of the art models in terms of several common LETOR metrics.

We adapt regression to the LETOR task by using Bregman divergences and monotone retargeting (MR). MR is a novel technique that we introduce in the paper and Bregman divergences [5] are a family of "distance like" functions well studied in optimization [8], statistics and machine learning [2] due to their one to one connections with modeling uncertainty using exponential family distributions. Bregman divergences are the unique class of strongly statistically consistent surrogate cost functions for the NDCG criterion [19], a de facto standard of ranking quality. In addition to these statistical properties, Bregman divergences have several properties useful for optimization and specifically useful for ranking. The LETOR task decomposes into subproblems that are equivalent to estimating (unconstrained as well as constrained) generalized linear models. The Bregman divergence machinery provides easy to implement, scalable algorithms for them, with a user chosen level of granularity of parallelism. We hope the reader will appreciate the flexibility of choosing an appropriate divergence to encode desirable properties on the rankings while enjoying the strong guarantees that come with the family.

We introduce MR by first discussing direct regression of rank scores and highlighting its primary deficiency: its attempt to fit the scores exactly. An exact fit is unnecessary since any score that induces the correct ordering is sufficient. MR addresses this problem by searching for a order preserving transformation of the target scores that may be easier for the regressor to fit: hence the name "retargeting".

Let us briefly sketch our line of attack. In section 2 we

* Both authors contributed equally.

present a method to reduce the optimization over the infinite class of all monotonic increasing functions to that of alternating projection over a finite dimensional vector space. In section 3.2.3 we show when that optimization problem is jointly convex by resolving the question of joint convexity of the Fenchel-Young gap. This result is important in its own right. We introduce Bregman divergences in section 3 and discuss properties that make them particularly suited to the ranking task. We show (i) that one set of the alternating projections can be computed in a Bregman divergence independent fashion 3.2.1, and (ii) separable Bregman divergences allow us to use sorting 3.2.2 that would have otherwise required exhaustive combinatorial enumeration or solving a linear assignment problem repeatedly.

Notation: Vectors are denoted by bold lower case letters, matrices are capitalized. $\boldsymbol{x}^\dagger$ denotes the transpose of the vector $\boldsymbol{x}$, $||\boldsymbol{x}||$ denotes the L_2 norm. $\mathrm{Diag}(\boldsymbol{x})$ denotes a diagonal matrix with its diagonal set to the vector $\boldsymbol{x}$. $\text{Adj-Diff}(\boldsymbol{x})$ denotes a vector obtained by taking adjacent difference of consecutive components of $\left[\begin{smallmatrix}\boldsymbol{x}\\0\end{smallmatrix}\right]$. Thus $\text{Cum-Sum}(\text{Adj-Diff}(\boldsymbol{x})) = \boldsymbol{x}$. A vector $\boldsymbol{x}$ is defined to be in *descending order* if $x_i \geq x_j$ if $i > j$, the set of such vectors is denoted by $\mathcal{R}\!\downarrow$. Vector $\boldsymbol{x}$ is isotonic with $\boldsymbol{y}$ if $x_i \geq x_j$ then $y_i \geq y_j$. The unit simplex is denoted by Δ and the positive orthant by ${\mathbb{R}_+}^d$. $\psi(\cdot)$ is used to denote the Legendre dual of the function $\phi(\cdot)$. Partitions of sets are denoted by Π and P.

2 Monotone Retargeting

We introduce our formulation of learning to rank, this consists of a set of queries $\mathcal{Q} = \{q_1, q_i \ldots q_{|\mathcal{Q}|}\}$ and a set of items $\mathcal{V}$ that are to be ranked in the context of the queries. For every query q_i, there is a subset $\mathcal{V}_i \subset \mathcal{V}$ whose elements have been ordered, based on their relevance to the query. This ordering is customarily expressed via a rank score vector $\tilde{\boldsymbol{r}}_i \in \mathbb{R}^{d_i=|\mathcal{V}_i|}$ whose components $\tilde{r}_{ij}$ correspond to items in $\mathcal{V}_i$. Beyond establishing an order over the set $\mathcal{V}_i$, the actual values of $\tilde{r}_{ij}$ are of no significance. For a query q_i the index j of $\tilde{r}_{ij}$ is local to the set $\mathcal{V}_i$ hence $\tilde{r}_{ij}$ and $\tilde{r}_{kj}$ need not correspond to the same object. We shall further assume, with no loss in generality, that the subscript j is assigned such that $\tilde{r}_{ij}$ is in a descending order for any $\mathcal{V}_i$. Note that $\tilde{\boldsymbol{r}}_i$ induces a partial order if the number of unique values k_i in the vector is less than d_i.

For every query-object pair $\{q_i, v_{ij}\}$ a feature vector $\mathbb{R}^n \ni \boldsymbol{a}_{ij} = F(q_i, v_{ij})$ is pre-computed. The subset of training data pertinent to any query q_i is the pair $\{\tilde{\boldsymbol{r}}_i, \boldsymbol{A}_i\}$ and is called its qset. Thus, the column vector $\tilde{\boldsymbol{r}}_i$ consists of the rank-scores $\tilde{r}_{ij}$ and $\boldsymbol{A}_i$ is a matrix whose j^{th} row is ${\boldsymbol{a}_{ij}}^\dagger$.

Given a loss function $D_i : \mathbb{R}^{|\mathcal{V}_i|} \times \mathbb{R}^{|\mathcal{V}_i|} \mapsto \mathbb{R}_+$ we may define the regression problem $\min_{\boldsymbol{w}} \sum_i D(\tilde{\boldsymbol{r}}_i, f(\boldsymbol{A}_i, \boldsymbol{w}))$ where $f : \mathbb{R}^{|\mathcal{V}_i| \times n} \times \mathbb{R}^n \mapsto \mathbb{R}^{|\mathcal{V}_i|}$ is some fixed parametric form with the parameter $\boldsymbol{w}$. As discussed, this is unnecessarily stringent for ranking. A better alternative is:

$$\min_{\boldsymbol{w}, \boldsymbol{\Upsilon}_i \in \mathcal{M}} \sum_i D_i\big(\tilde{\boldsymbol{r}}_i, \boldsymbol{\Upsilon}_i \circ f(\boldsymbol{A}_i, \boldsymbol{w})\big),$$

where $\boldsymbol{\Upsilon}_i : \mathbb{R}^{|\mathcal{V}_i|} \mapsto \mathbb{R}^{|\mathcal{V}_i|}$ transforms the component of its argument by a fixed monotonic increasing function Υ_i, and $\mathcal{M}$ is the class of all such functions. Now $f(\boldsymbol{A}_i, \boldsymbol{w})$ no longer need to equal $\tilde{\boldsymbol{r}}_i$ point-wise to incur zero loss. It is sufficient for some monotonic increasing transform of $f(\boldsymbol{A}_i, \boldsymbol{w})$ to do so. With no loss in generality of modeling, we may apply the monotonic transform to $\tilde{\boldsymbol{r}}_i$ instead. This avoids the minimization over the function composition, but the need for minimizing over the set of all monotone functions remains. One possible way to eliminate the minimization over the function space is to restrict our attention to some parametric family in $\mathcal{M}$ at the expense of generality. Instead, with no loss in generality, the optimization over the infinite space of functions $\mathcal{M}$ can be converted into one over finite dimensional vector spaces $\mathbb{R}^{|\mathcal{V}_j|}$, provided we have a finite characterization of the constraint set $\mathcal{R}\!\downarrow_i$:

$$\min_{\boldsymbol{w}, \boldsymbol{r} \in \mathcal{R}\downarrow_i} \sum_i D_i(\boldsymbol{r}_i, f(\boldsymbol{A}_i, \boldsymbol{w})) \text{ s.t. } \mathcal{R}\!\downarrow_i = \{\boldsymbol{r}|^{\exists \boldsymbol{M} \in \mathcal{M}}_{\boldsymbol{M}(\tilde{\boldsymbol{r}}_i)=\boldsymbol{r}}\}. \tag{1}$$

The Set $\mathcal{R}\!\downarrow_i$: The convex composition $\boldsymbol{r} = \alpha \boldsymbol{r}_1 + (1-\alpha)\boldsymbol{r}_2$ of two isotonic vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ preserves isotonicity, as does the scaling $\alpha \boldsymbol{r}_1$ for any $\alpha \in \mathbb{R}_+$. Hence the set $\mathcal{R}\!\downarrow_i$ is a convex cone. This makes the problem computationally tractable because the set can be described entirely by its extreme rays, or by the extreme rays of its polar. We claim the set $\mathcal{R}\!\downarrow_i$ can be expressed as the image of the set $\{\mathbb{R}_+\}^{|\mathcal{V}_i|-1} \times \mathbb{R}$ under a linear transformation by a particular upper triangular matrix U with positive entries:

$$\mathcal{R}\!\downarrow_i = U\boldsymbol{x} \quad \text{s.t.} \quad \boldsymbol{x} \in \{\mathbb{R}_+\}^{|\mathcal{V}_i|-1} \times \mathbb{R}$$

The matrix U is not unique and can be generated from any vector $\boldsymbol{v} \in {\mathbb{R}_+}^{|\mathcal{V}_i|}$, but as we shall see, any member from the allowed class of U is sufficient for a *exhaustive* representation of $\mathcal{R}\!\downarrow_i$.

Lemma 1. *The set of all vectors in $\mathbb{R}^d$ that are sorted in a descending order is given by $U\boldsymbol{x}$ s.t. $\boldsymbol{x} \in \{\mathbb{R}_+\}^{|\mathcal{V}_i|-1} \times \mathbb{R}$ where U is a triangular matrix generated from a vector $\boldsymbol{v} \in {\mathbb{R}_+}^d$ such that the i^{th} row $U(i,:)$ is $\{0\}^{i-1} \times \boldsymbol{v}(i:)$*

Proof. Consider solving $U\boldsymbol{x} = \tilde{\boldsymbol{r}}_i$ for any vector $\tilde{\boldsymbol{r}}_i$ sorted in descending order. We have $\boldsymbol{x} = (\mathrm{Diag})^{-1}(\boldsymbol{v}) \times \text{Adj-Diff}(\tilde{\boldsymbol{r}}_i)$ which is in $\{\mathbb{R}_+\}^{|\mathcal{V}_i|-1} \times \mathbb{R}$ □

For regression functions capable of fitting an arbitrary additive offset, no generality is lost by constraining the last component to be non-negative.

In addition to the set $\mathcal{R}\!\downarrow_i$ we shall make frequent use of the set of all discrete probability distributions that are in descending order, i.e. $\mathcal{R}\!\downarrow_i \cap \Delta_i$ that we represent by Δ_o^i.

We give a similar representation of this set by generating an upper triangular matrix T from the vector $\boldsymbol{v}_\Delta = \{1, \frac{1}{2}, \cdots \frac{1}{i} \cdots \frac{1}{d}\}$ and considering $\boldsymbol{x} \in \Delta$.

Lemma 2. *The set Δ_o of all discrete probability distributions of dimension d that are in descending order is the image $T\boldsymbol{x}$ s.t. $\boldsymbol{x} \in \Delta$ where T is an upper triangular matrix generated from the vector $\boldsymbol{v}_\Delta = \{1, \frac{1}{2} \cdots \frac{1}{d}\}$ such that $T(i,:) = \{0\}^{i-1} \times \boldsymbol{v}_\Delta(i:)$*

Proof. The proof follows Lemma (1). $T\boldsymbol{x}$ is in the simplex Δ because it is a convex combination of vectors in Δ. □

With appropriate choices of the distance like function $D_i(\cdot,\cdot)$ and the curve fitting function $f(\cdot,\cdot)$ we can transform (1) into a bi-convex optimization [1] problem over a product of convex sets. We choose $D_i(\cdot,\cdot)$ to be a Bregman divergence $D_\phi\big(\cdot\big|\big|\cdot\big)$, defined in Section 3.1, and $f(\boldsymbol{A}_i, \boldsymbol{w})$ to be $(\nabla\phi)^{-1}(\langle \boldsymbol{A}_i, \boldsymbol{w}\rangle)$, leading to the formulation:

$$\min_{\boldsymbol{w}\in\mathcal{W}, \boldsymbol{r}\in\mathcal{R}\downarrow_i} \sum_{i=1}^{|\mathcal{Q}|} \frac{1}{|\mathcal{V}_i|} D_{\boldsymbol{\phi}}\big(\boldsymbol{r}_i\big|\big|(\nabla\phi)^{-1}(\langle \boldsymbol{A}_i, \boldsymbol{w}\rangle)\big). \quad (2)$$

Coordinate-wise updates of (2) are equivalent to learning canonical GLMs under linear constraints for which scalable techniques are known [9]. The LETOR task has additional structure that allows more efficient solutions.

3 Background

We make heavy use of identities and algorithms associated with Bregman divergences, some that to the best of our knowledge are new e.g. Theorem 2, Lemmata 3, 4 and independent proof of Theorem 1. Theorem 2, and Lemmata 3, and 4 are particularly relevant to ranking. The purpose of this section is to collect these results in a single place.

3.1 Definitions

Bregman Divergence: Let $\boldsymbol{\phi} : \Theta \mapsto \mathbb{R},\ \Theta = \operatorname{dom}\boldsymbol{\phi} \subseteq \mathbb{R}^d$ be a strictly convex, closed function, differentiable on $\operatorname{int}\Theta$. The corresponding Bregman divergence $D_\phi\big(\cdot\big|\big|\cdot\big) : \operatorname{dom}(\boldsymbol{\phi}) \times \operatorname{int}(\operatorname{dom}(\boldsymbol{\phi})) \mapsto \mathbb{R}_+$ is defined as $D_\phi\big(\boldsymbol{x}\big|\big|\boldsymbol{y}\big) \triangleq \boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{y}) - \langle \boldsymbol{x}-\boldsymbol{y}, \nabla\boldsymbol{\phi}(\boldsymbol{y})\rangle$. From strict convexity it follows that $D_\phi\big(\boldsymbol{x}\big|\big|\boldsymbol{y}\big) \geq 0$ and $D_\phi\big(\boldsymbol{x}\big|\big|\boldsymbol{y}\big) = 0$ iff $\boldsymbol{x} = \boldsymbol{y}$. Bregman divergences are (strictly) convex in their first argument, but not necessarily convex in their second.

In this paper we only consider functions of the form $\boldsymbol{\phi}(\cdot) : \mathbb{R}^n \ni \boldsymbol{x} \mapsto \sum_i w_i\phi(x_i)$ that are weighted sums of *identical* scalar convex functions applied to each component. We refer to this class as *weighted, identically separable* (**WIS**) or simply **IS** if the weights are equal. This class has properties particularly suited to ranking. Mahalonobis distance with diagonal W, weighted KL divergence $wKL(\boldsymbol{x}\|\boldsymbol{y})$ and weighted and shifted generalized I-divergence $wGI(\boldsymbol{x}\|\boldsymbol{y})$ are in this family (Table 1).

[1] A biconvex function is a function of two arguments such that with any one of its argument fixed the function is convex in the other argument.

$\boldsymbol{\phi}(\boldsymbol{x})$	$D_\phi\big(\boldsymbol{x}\big\|\big\|\boldsymbol{y}\big)$
$\frac{1}{2}\|\|\boldsymbol{x}\|\|^2_W$	$\frac{1}{2}\|\|\boldsymbol{x}-\boldsymbol{y}\|\|^2_W$
$\sum_i w_i x_i \log x_i\ \boldsymbol{x} \in \Delta$	$wKL(\boldsymbol{x}\|\boldsymbol{y}) = \sum_i w_i x_i \log(\frac{x_i}{y_i})$
$\sum_i w_i(x_i \log x_i - x_i)$ $\boldsymbol{x} \in \mathbb{R}_+{}^d$	$wGI(\boldsymbol{x}\|\boldsymbol{y})$ $= \sum_i w_i\big((x_i-1)\log(\frac{x_i-1}{y_i-1}) - x_i + y_i\big)$

Table 1: Examples of WIS Bregman divergences.

Bregman Projection: Given a closed convex set $\mathcal{S}$, the Bregman-projection of $\boldsymbol{q}$ on $\mathcal{S}$ is $\operatorname{Proj}^\phi(q,\mathcal{S}) \triangleq \operatorname{Argmin}_{\boldsymbol{p}} D_\phi\big(\boldsymbol{p}\big|\big|\boldsymbol{q}\big)\ \boldsymbol{p} \in S$.

A function $\phi(\cdot)$ has **modulus of strong convexity** s if $\phi(\alpha\boldsymbol{x}+(1-\alpha)\boldsymbol{y}) \leq \alpha\phi(\boldsymbol{x})+(1-\alpha)\phi(\boldsymbol{y}) - \frac{s}{2}\alpha(1-\alpha)||\boldsymbol{x}-\boldsymbol{y}||^2$. For a twice differentiable $\phi(\boldsymbol{x})$ this means that eigenvalues of its Hessian are lower bounded by s.

The **Legendre conjugate** $\psi(\cdot)$ of the function $\phi(\cdot)$ is defined as $(\phi)^*(\boldsymbol{x}) \triangleq \psi(\boldsymbol{x}) \triangleq \sup_{\boldsymbol{\lambda}}(\langle\boldsymbol{\lambda},\boldsymbol{x}\rangle - \phi(\boldsymbol{\lambda}))$. If $\phi(\cdot)$ is a convex function of Legendre type [21], as will always be the case in this paper, $\big((\phi)^*\big)^*(\cdot) = \phi(\cdot)$ and $(\nabla\phi(\cdot))^{-1} = \nabla\psi(\cdot)$ is a one to one mapping.

The **Fenchel-Young** inequality (3) is fundamental to convex analysis and plays an important role in our analysis.

$$\psi(\boldsymbol{y}) + \phi(\boldsymbol{x}) - \langle\boldsymbol{y},\boldsymbol{x}\rangle \geq 0. \quad (3)$$

3.2 Properties

The convexity of (2) in $\boldsymbol{r}$ and $\boldsymbol{w}$ (separately) can be proven by verifying the identity (by evaluating its LHS):

$$D_\psi\big(\nabla\phi(\boldsymbol{y})\big|\big|\nabla\phi(\boldsymbol{x})\big) = D_\phi\big(\boldsymbol{x}\big|\big|\boldsymbol{y}\big). \quad (4)$$

3.2.1 Universality Of Minimizers

A mean-variance like decomposition (described in the appendix, Theorem (3)) holds for all Bregman divergences. It plays a critical role in Theorem 1 that has significant impact in facilitating the solution of the LETOR problem.

Theorem 1. *For $\mathcal{R}\downarrow\ \subset\ \mathbb{R}^d$ the entire set of vectors with descending ordered components, the minimizer $\boldsymbol{y}^* = \operatorname{Argmin}_{\boldsymbol{y}\in\mathcal{R}\downarrow} D_\phi\big(\boldsymbol{x}\big|\big|\boldsymbol{y}\big)$ is independent of $\boldsymbol{\phi}(\cdot)$ if $\boldsymbol{\phi}(\cdot)$ is WIS.*

Proof: sketched in the appendix.

Following our independent proof of Theorem 1, we have since come across an older proof [20] in the context of maximum likelihood estimators of exponential family models under conic constraints that were developed prior to the

popularity of Bregman divergences. Whereas the older proof uses Moreau's cone decomposition[21], ours uses Theorem 3 (in appendix) and yields a much shorter proof.

Corrolary 1. *If* $\operatorname{dom}\psi(\cdot) = \mathbb{R}^d$ *where* $\psi(\cdot)$*is the Legendre conjugate of the WIS convex function* $\boldsymbol{\phi}(\cdot)$ *then* $\operatorname{Argmin}_{\boldsymbol{y}\in\mathcal{R}\downarrow\cap\operatorname{dom}\phi} D_\phi\big(\boldsymbol{y}\big|\big|(\nabla\phi)^{-1}(\boldsymbol{x})\big) = (\nabla\boldsymbol{\phi})^{-1}(\boldsymbol{y}^*)$ *where* $\boldsymbol{y}^* = \operatorname{Argmin}_{\boldsymbol{y}\in\mathcal{R}\downarrow} ||\boldsymbol{x}-\boldsymbol{y}||^2$.

Corollary (1) implies that for choices of convex function $\boldsymbol{\phi}(\cdot)$ indicated, the minimization over $\boldsymbol{r}_i \in \mathcal{R}\downarrow \cap \operatorname{dom}\boldsymbol{\phi}$ can be obtained by transforming the equivalent squared loss minimizer by $(\nabla\boldsymbol{\phi})^{-1}(\cdot)$. The squared loss minimization is not only simpler but its implementation can now be shared across all different $\boldsymbol{\phi}(\cdot)$s where Corollary 1 applies. This class of convex functions is the same as "essentially smooth" [21]. Three such functions are listed in Table 1.

3.2.2 Optimality of Sorting

For any sorted vector $\boldsymbol{x}$, finding the permutation of $\boldsymbol{y}$ that minimizes $D_\phi\big(\boldsymbol{x}\big|\big|\boldsymbol{y}\big)$ shows up as a subproblem in our formulation that needs to be solved in an inner loop. Thus solving it efficiently is critical and this is yet another instance where Bregman divergences are very useful.

For an arbitrary divergence function the search over the optimal permutation is a *non-linear assignment* problem that can be solved only by exhaustive enumeration. For an arbitrary separable divergence the optimal permutation may be found by solving a linear assignment problem, which is an integer linear program and hence also expensive to solve (especially in an inner loop, as required in our algorithm).

On the other hand, if $\boldsymbol{\phi}(\cdot)$ is IS, the solution is remarkably simple, as shown in Lemma 3 where $\left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]$ denotes a vector in $\mathbb{R}^2$ with components x_1 and x_2.

Lemma 3. *If* $x_1 \geq x_2$ *and* $y_1 \geq y_2$ *and* $\boldsymbol{\phi}(\cdot)$ *is IS, then* $D_\phi\big(\left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]\big|\big|\left[\begin{smallmatrix}\boldsymbol{y}_1\\ \boldsymbol{y}_2\end{smallmatrix}\right]\big) \leq D_\phi\big(\left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]\big|\big|\left[\begin{smallmatrix}\boldsymbol{y}_2\\ \boldsymbol{y}_1\end{smallmatrix}\right]\big)$ *and* $D_\phi\big(\left[\begin{smallmatrix}\boldsymbol{y}_1\\ \boldsymbol{y}_2\end{smallmatrix}\right]\big|\big|\left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]\big) \leq D_\phi\big(\left[\begin{smallmatrix}\boldsymbol{y}_2\\ \boldsymbol{y}_1\end{smallmatrix}\right]\big|\big|\left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]\big)$

Proof.
$D_\phi\big(\left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]\big|\big|\left[\begin{smallmatrix}\boldsymbol{y}_1\\ \boldsymbol{y}_2\end{smallmatrix}\right]\big) - D_\phi\big(\left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]\big|\big|\left[\begin{smallmatrix}\boldsymbol{y}_2\\ \boldsymbol{y}_1\end{smallmatrix}\right]\big) = \langle(\nabla\phi(y_2) - \nabla\phi(y_1)), x_1 - x_2\rangle\,.$
There exists $c \geq 0$ s.t. $x_1 - x_2 = c(y_1 - y_2)$. Proof follows from monotonicity of $\nabla\phi$, ensured by convexity of ϕ. We can exchange the order of the arguments using (4). □

Using induction over d for $\boldsymbol{y} \in \mathbb{R}^d$ the optimal permutation is obtained by sorting. Not only is Lemma 3 extremely helpful in generating descent updates, it has fundamental consequences in relation to the local and global optimum of our formulation (see Lemma 4).

3.2.3 Joint Convexity and Global Minimum

Using Legendre duality one recognizes that equation (2) quantifies the gap in the Fenchel-Young inequality (3).

$$D_{\boldsymbol{\phi}}\big(\boldsymbol{r}_i\big|\big|(\nabla\phi)^{-1}\left(\boldsymbol{A}_i\boldsymbol{w}\right)\big) = \psi(\boldsymbol{A}_i\boldsymbol{w}) + \phi(\boldsymbol{r}_i) - \langle\boldsymbol{r}_i, \boldsymbol{A}_i\boldsymbol{w}\rangle\,.$$

Although this clarifies the issue of separate convexity in $\boldsymbol{w}$ and $\boldsymbol{r}_i$, the conditions under which joint convexity is obtained is not obvious. Joint convexity, if ensured, guarantees global minimum even for a coordinate-wise minimization because our constraint set is a product of convex sets. We resolve this important question in Theorem 2.

Theorem 2. *The gap in the Fenchel-Young inequality* $\psi(\boldsymbol{y}) + \phi(\boldsymbol{x}) - \langle\boldsymbol{x}, \boldsymbol{w}\rangle$ *for any twice differentiable strictly convex* $\phi(\cdot)$ *with a differentiable conjugate* $(\phi)^*(\cdot) = \psi(\cdot)$ *is jointly convex if and only if* $\phi(\boldsymbol{x}) = c||\boldsymbol{x}||^2$ *for all* $c > 0$.

Proof: sketched in the appendix.

3.3 Algorithms

Now we discuss Bregman's algorithm associated with Bregman divergences. The original motivation for introducing [5] Bregman divergence was to generalize alternating orthogonal projection. A significant advantage of the algorithm is its scalability and suitability for parallelization. It solves the following (Bregman projection) problem:

$$\min_{\boldsymbol{x}} D_\phi\big(\boldsymbol{x}\big|\big|\boldsymbol{y}\big) \text{ s.t. } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{5}$$

Bregman's algorithm:

Initialize: $\boldsymbol{\lambda}^0 \in \mathbb{R}+^d$ and z^0 such that
$\nabla\phi(z^0) = \left[\boldsymbol{A}^\dagger|\nabla\phi(\boldsymbol{y})\right]\left[\boldsymbol{\lambda}^{0\dagger}, 1\right]^\dagger$

Repeat: Till convergence

Update: Apply **Sequential** or **Parallel Update**

Solve: $\nabla\phi(z^{t+1}) = \left[\boldsymbol{A}^\dagger|\nabla\phi(\boldsymbol{y})\right]\left[\boldsymbol{\lambda}^{t+1\dagger}, 1\right]^\dagger$

Sequential Bregman Update:

Select i: Let $\mathcal{H}_i = \{\boldsymbol{z}|\,\langle\boldsymbol{a}_i, \boldsymbol{z}\rangle \leq b_i\}$

If in violation: Compute $\operatorname{Proj}^\phi\left(\boldsymbol{z}^t, \mathcal{H}_i\right)$ i.e.
$\nabla\phi(\operatorname{Proj}^\phi\left(\boldsymbol{z}^t, \mathcal{H}_i\right)) = \nabla\phi(\boldsymbol{z}^t) + c_i^t\boldsymbol{a}_i,$

Update: $\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + c_i^t\mathbf{1}_i$

Parallel Bregman Update:

For all i **in parallel:** Compute $\operatorname{Proj}^\phi\left(\boldsymbol{z}^t, \mathcal{H}_i\right), c_i^t$

Update: $\boldsymbol{\lambda}_i^{t+1} = \boldsymbol{\lambda}^t + c_i^t\mathbf{1}_i$

Synchronize: $\boldsymbol{\lambda}^{t+1} = \nabla^{-1}\phi(\sum_i \nabla\phi(\boldsymbol{\lambda_i}^{t+1}))$

4 LETOR with Monotone Retargeting

Our cost function is an instantiation of (2) with a WIS Bregman divergence. In addition, we include regularization and a query specific offset. Note that the cost function (2) is not invariant to scale. For example squared Euclidean, KL

divergence and generalized I-divergence are homogeneous functions of degree 2, 1 and 1 respectively. Thus the cost can be reduced just by scaling its arguments down, without actually learning the task. To remedy this, we restrict the $\boldsymbol{r}_i$s from shrinking below a pre-prescribed size. This is accomplished by constraining $\boldsymbol{r}_i$s to lie in an appropriate closed convex set separated from the origin, for example, an unit simplex or a shifted positive orthant. This yields:

$$\min_{\beta_i,\boldsymbol{w},\boldsymbol{r}_i\in\mathcal{R}\downarrow_i\cap\mathcal{S}_i}\sum_{i=1}^{|\mathcal{Q}|}\frac{1}{|\mathcal{V}_i|}D_{\boldsymbol{\phi}}\big(\boldsymbol{r}_i\Big|\Big|(\nabla\phi)^{-1}\left(\boldsymbol{A}_i\boldsymbol{w}+\beta_i\mathbf{1}\right)\big)+\frac{C}{2}||\boldsymbol{w}||^2, \quad (6)$$

or equivalently

$$\min_{\beta_i,\boldsymbol{w},\boldsymbol{r}_i\in\mathcal{R}\downarrow_i\cap\mathcal{S}_i}\sum_{i=1}^{|\mathcal{Q}|}\frac{1}{|\mathcal{V}_i|}D_{\boldsymbol{\psi}}\big(\boldsymbol{A}_i\boldsymbol{w}+\beta_i\mathbf{1}\Big|\Big|\nabla\phi\left(\boldsymbol{r}_i\right)\big)+\frac{C}{2}||\boldsymbol{w}||^2, \quad (7)$$

where $\mathcal{S}_i$ are bounded sets excluding $\mathbf{0}$, chosen to suit the divergence. The parameter C is the regularization parameter. In non-transductive settings, the query specific offsets β_i will not be available for the test queries. This causes no difficulty because β_i does not affect the relative ranks over the documents. We update the $\boldsymbol{r}_i$'s and $\{\boldsymbol{w},\{\beta_i\}\}$ alternately. Note that each is a Bregman projection.

If $\mathcal{S}_i=\operatorname{dom}\boldsymbol{\phi}$ and $\operatorname{dom}\boldsymbol{\psi}=\mathbb{R}^d$, the optimization over $\boldsymbol{r}_i$ reduces to an order constrained least squares problem (corrolary-1). Examples of such matched pairs are (i) $wKL\left(\cdot\|\cdot\right)$ and Δ_i, and (ii) shifted $wGI\left(\cdot\|\cdot\right)$ and $\mathbf{1}+{\mathbb{R}_+}^d$. A well studied, scalable algorithm for the ordered least squares problem is pool of adjacent violators (PAV) algorithm [3]. One can verify that PAV, like Bregman's algorithm (5) is a dual feasible method. One may also use Lemma 1 to solve it as a non-negative least squares problem for which several scalable algorithm exists [15].

To be able to use Bregman's algorithm it is essential that $\mathcal{R}\downarrow_i$ be available as an intersection of linear constraints, as is readily obtained for any prescribed total order, as shown:

$$\mathcal{R}\downarrow_i=\{r_{i,j+1}-r_{i,j}\leq 0\}_{\forall j\in\mathcal{J}_i},$$
$$\Delta_i^o=\mathcal{R}\downarrow_i\cap\{\sum_j r_{ij}=1\}\cap\{r_{i,d_i}>0\}. \quad (8)$$

Partial orders are discussed in section 4.1.

The advantages of the Bregman updates (3.3), are that they are easy to implement (more so when $\operatorname{Proj}^{\phi}\left(\cdot,\cdot\right)$ is available in closed form e.g. squared Euclidean), have minimal memory requirements, and hence they scale readily and allow easy switch from a sequential to a parallel update.

The parallel Bregman updates applied to (2), (8) clearly exposes massive amounts of fine grained parallelism at the level of individual inequalities in $\mathcal{R}\downarrow_i$ or Δ_i^o, and is well suited for implementation on a GPGPU[18]. We note further that the optimization for $\boldsymbol{r}_i$ is independent for each query, thus can be embarrassingly parallelized.

For optimizing over $\boldsymbol{w}$ one may use several techniques available for parallelizing a sum of convex functions, for example parallelize the gradient computation across the terms or use more specialized technique such as alternating direction of multipliers [4]. Further, $\{\boldsymbol{w},\{\beta_i\}\}$ can be solved jointly simply by augmenting the feature matrix $\boldsymbol{A}_i$ with $\mathbf{1}$. We hope the readers will appreciate this flexibility of being able to exploit parallelism at different levels of granularity of choice.

4.1 Partial Order

Recall that a partial order is induced if the number of unique rank scores k_i in $\tilde{\boldsymbol{r}_i}$ is less than d_i. In this case our convention of indexing $\mathcal{V}_i$ in a descending order is ambiguous. To resolve this, we break ties arbitrarily. Consider a subset of $\mathcal{V}_i$ whose elements have the same training rank-score. We distinguish between two modeling choices: (a) the items in that subset are not really equivalent, but the training set used a resolution that could not make fine distinctions between the items,[2] we call this the "hidden order" case, and (b) the items in the subset are indeed equivalent and the targets are constrained to reflect the same block structure, we call this case "block equivalent" and can model it appropriately. Although we have removed the discussion on the latter in the interest of space, this too can be modeled efficiently by MR.

4.1.1 Partially Hidden Order

In this model we assume that the items are totally ordered, though the finer ordering between similar items is not visible to the ranking algorithm. Let $P_i=\{P_{ik}\}_{k=1}^{k_i}$ be a partition of the index set of $\mathcal{V}_i$, such that all items in P_{ik} have the same training rank-score. We denote their sizes by $d_{ik}=|P_{ik}|$. The sets $\mathcal{V}_i$ effectively get partitioned further into $\{P_{ik}\}_{k=1}^{k_i}$ by the $k_i<d_i$ unique scores given to each of its members. Though such a score specifies an order between items from any two different sets P_{ij} and P_{il}, the order within any set P_{ik} remains unknown. This is very common in practice and is usually an artifact of the high cost of acquiring training data in a totally ordered form. The optimization problem may be solved using either an inner or an outer representation of the constraint sets.

Outer representation: Recall that Bregman's algorithm 3.3 is better suited for the outer representation (8).

Denote the set of rank-score vectors having the same partially ordered structure as $\tilde{\boldsymbol{r}_i}$ by $\mathfrak{R}_i$. For partial order we may describe $\mathfrak{R}_i$ by linear inequalities as follows:

$$\{r_{im}>r_{in}\}_{j=1}^{k_i-1}\forall_{i\in[1,|\mathcal{Q}|]},\ m\in P_{ij},\ n\in P_{i,j+1},$$

with each j generating $d_{ij}d_{i,j+1}$ inequalities, which is very high. The proliferation of inequalities may be reduced by

[2] or that, we only care to reduce the error of predicting $r_{ij}>r_{ik}$ when $\tilde{r}_{ij}<\tilde{r}_{ik}$, note the strict inequality.

$$\boldsymbol{x}_i^{t+1} = \underset{\boldsymbol{x}\in\Delta}{\text{Argmin}}\, D_{\boldsymbol{\phi}}\left(T\boldsymbol{x}\middle|\middle|(\nabla\phi)^{-1}\left(\mathbb{P}_i^t\boldsymbol{A}_i\boldsymbol{w}^t+\beta_i^t\right)\right)\ \ \forall i \tag{10}$$

$$\mathbb{P}_i^{t+1} = \underset{\pi}{\text{Argmin}}\, D_{\boldsymbol{\phi}}\left(T\boldsymbol{x}_i^{t+1}\middle|\middle|(\nabla\phi)^{-1}\left(\pi\boldsymbol{A}_i\boldsymbol{w}^t+\beta_i^t\right)\right)\ \ \forall i \tag{11}$$

$$\boldsymbol{w}^{t+1}, \{\beta_i^{t+1}\} = \tag{12}$$
$$\underset{\boldsymbol{w},\{\beta_i\}}{\text{Argmin}} \sum_{i=1}^{|\mathcal{Q}|} D_{\boldsymbol{\phi}}\left(T\boldsymbol{x}^{t_i+1}\middle|\middle|(\nabla\phi)^{-1}\left(\mathbb{P}_i^{t+1}\boldsymbol{A}_i\boldsymbol{w}+\beta_i^t\right)\right)+\frac{C}{2}||\boldsymbol{w}||^2$$

Figure 1: Algorithm for Partially Hidden Order

introducing auxiliary variables $\{\bar{r}_{i,l}\}_{l=1}^{k_i-1}$ and the following inequalities:

$$\{\bar{r}_{i,j+1} > r_{iP_{ij}} > \bar{r}_{i,j}\}\forall_{i\in[1,|\mathcal{Q}|]}. \tag{9}$$

However, since Bregman's algorithms are essentially coordinate-wise ascent methods, their convergence may slow unless fine grained parallelism can be exploited. For commodity hardware, an alternative to the exterior point methods are interior point methods that use an inner representation of the convex constraint set.

Inner representation: For our experiments we use the updates in figure 1. In particular, we use the method of D proximal gradients for (10) where the proximal term is a Bregman divergence defined by a convex function whose domain is the required constraint set [13], [22], [16]. This automatically enforces the required constraints.

To handle partial order we introduce a block-diagonally restricted permutation matrix $\mathbb{P}_i$ that can permute indices in each P_{ij} independently. Since the items in P_{ij} are not equivalent they are available for re-ordering as long as that minimizes the cost (6). Block weighted IS Bregman divergences have the special property that sorting minimizes the divergence over all permutations (Lemma 3). Thus update (11) can be accomplished by sorting.

The updates (10), (11) and (12) each reduce the lower bounded cost (6), and therefore the algorithm described in figure 1 converges. However, the vital question about whether the updates converge to a stationary point remains.

Convergence to a Stationary Point If repeated application of (10) and (11) (sorting) for a fixed $\boldsymbol{w}^{t+1}, \{\beta_i^{t+1}\}$ achieves the minimum then convergence to the stationary point is guaranteed. Thus we explore the question whether (11) and (10) together achieves a local minimum.

The tri-factored form $\boldsymbol{r}_i\mathbb{P}_iU\boldsymbol{x}_i$ is a cause for concern. Somewhat re-assuring is the fact that the range of $\mathbb{P}_iU\boldsymbol{x}_i$ is $\mathfrak{R}_i$ which again is a convex cone and that the tri-factored representation of any point in that cone is described uniquely. This however is not sufficient to ensure that a minimum is achieved by (10) and (11) because though the constraint set is convex, the cost function (6) is not convex in the tri-factored parameterization. Worse still, the parameterization is discontinuous because of the discrete nature of $\mathbb{P}$.

While one may address the discreteness problem via a real-relaxation of $\mathbb{P}$ to doubly stochastic matrices, the local minima attained in such a case will be in the interior of the Birkhoff polytope and not at the vertices that (11) sorting would have obtained. Therefore such a convex relaxation cannot answer the question whether (10) and sorting achieves the local minimum. Thus it is surprising that sorting followed by the $\boldsymbol{x}_i$ updates does achieve the local minimum of (6) on the cone $\mathfrak{R}_i$, as a consequence of the following Lemma.

Lemma 4. *Let vectors* $\left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]$ *and* $\left[\begin{smallmatrix}\boldsymbol{y}_1\\ \boldsymbol{y}_2\end{smallmatrix}\right]$ *be conformally partitioned. Let* $\left[\begin{smallmatrix}\boldsymbol{y}_1\\ \boldsymbol{y}_2\end{smallmatrix}\right]^* = \underset{\boldsymbol{y}_i'\in\Pi(\boldsymbol{y}_1),\ \boldsymbol{y}_1'\geq\boldsymbol{y}_2'}{\text{Argmin}} D_\phi\left(\left[\begin{smallmatrix}\boldsymbol{y}'_1\\ \boldsymbol{y}'_2\end{smallmatrix}\right]\middle|\middle|\left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]\right)$ *where* $\Pi(\boldsymbol{y}_i)$ *is the set of all permutations of the vector* $\boldsymbol{y}_i$*. If the Bregman divergence* $D_\phi\left(\cdot\middle|\middle|\cdot\right)$ *is conformally separable then* $\boldsymbol{y}_i^*$ *is isotonic with* $\boldsymbol{x}_i$ $\forall i = 1, 2$

Proof. The proof is by contradiction. Assume $\boldsymbol{y}_i^*$ is a minimizer that is not isotonic with $\boldsymbol{x}_i$, then one may permute $\boldsymbol{y}_i^*$ to match the order of $\boldsymbol{x}_i$ to obtain a reduced cost, yielding a contradiction. □

The utility of Lemma 4 is that in spite of the caveats mentioned it can correctly identify the internal ordering of the components of the left hand side that achieves the minimum for a fixed $\boldsymbol{w}^{t+1}, \{\beta_i^{t+1}\}$, given a fixed right hand side. With the knowledge of the order obtained, one may then compute the actual values with relative ease with (10).

5 Experiments

We evaluated the ranking performance of the proposed monotone retargeting approach on the benchmark LETOR 4.0 datasets (MQ2007, MQ2008) [23] as well as the OHSUMED dataset [12]. Each of these datasets is pre-partitioned into five-fold validation sets for easy comparison across algorithms. For OHSUMED, we used the *QueryLevelNorm* partition. Each dataset contains a set of queries, where each document is assigned a relevance score from irrelevant ($r = 0$) to relevant ($r = 2$).

All algorithms were trained using a regularized linear ranking function, with a regularization parameter chosen from the set $C \in \{10^{-50}, 10^{-20}, 10^{-10}, 10^{-5}, 10^{0}, 10^{1}\}$. The best model was identified as the model with highest mean average precision (MAP) on the validation set. All presented results are of average performance on the test set. As the baseline, we implemented the NDCG consistent re-normalization approach in [19] (using the NDCG$_m$ normalization) for the squared loss and the I-divergence (generalized KL-divergence). ListNet was implemented [7] as the KL divergence baseline since their normalization has

no effect on KL-divergence. MR was implemented using the *partially hidden order* monotone retargeting approach (Section 4.1). We compared the performance of MR (Normalized MR) to the MR method with the normalization $\frac{1}{|\mathcal{V}_i|}$ removed (Unnormalized MR).

The algorithms were implemented in Python and executed on a 2.4GHz quad-core Intel Xeon processor without paying particular attention to writing optimized code. Ample room for improvement remains. Square loss was the fastest with respect to average execution times per iteration at 0.58 seconds whereas KL achieved 1.01 seconds per iteration and I-div 1.14 seconds per iteration. We found that although MQ2007 is more than 4 times larger than MQ2008, MQ2007 only required about twice the time execution on average, highlighting the scalability of MR. On average SQ, KL and I-div took 99, 90 and 65 iterations.

Table 5 compares the algorithms in terms of expected reciprocal return (ERR) [10], Mean average precision (MAP) and NDCG. Unnormalized KL divergence cost function led to the best performance across datasets. The most significant gains over the baseline were for the I-divergence cost function. Monotone retargeting showed consistent performance gains over the baseline across metrics (NDCG, ERR, Precision), suggesting the effectiveness of MR for improving the overall ranking performance.

Figure 2 shows a subset of performance comparisons using the NDCG@N and Precision@N metrics. Our experiments show a significant improvement in performance on the range of datasets and cost functions. Across datasets, the difference between the baseline and our results were most significant with the I-divergence (generalized KL divergence) cost function.

There are two things worth taking special note of: (i) Although the baseline algorithms were proposed specifically for improving NDCG performance, MR improves the ranking accuracy further, even in terms of NDCG. (ii) MR seems to be consistently peaking early. This property is particularly desirable and is encoded specifically in the cost functions such as NDCG and ERR. In our initial formulation we used WIS Bregman divergence so that the weights could be tuned to obtained the early peaking behavior. However that proved unnecessary because even the unweighted model produced satisfactory performance. The effect of query length normalization was, however, inconsistent. Some of our results were insensitive to it, whereas other results were adversely affected. We conjecture that it is an artifact of using the same amount of regularization as in the un-normalized case.

MQ 2007 ERR			
	I-div	SQ	KL
Unnormalized MR	0.3698	0.3703	**0.3737**
Normalized MR	0.3702	0.3601	0.3731
Baseline	0.1953	0.3639	0.3643
MQ 2007 MAP			
	I-div	SQ	KL
Unnormalized MR	0.5379	0.5361	0.5398
Normalized MR	0.5358	0.5282	**0.5399**
Baseline	0.3611	0.5330	0.5380
MQ 2007 NDCG			
	I-div	SQ	KL
Unnormalized MR	0.6961	**0.7398**	0.6978
Normalized MR	0.6954	0.6953	0.6981
Baseline	0.5512	0.6927	0.6952
MQ 2008 ERR			
	I-div	SQ	KL
Unnormalized MR	0.4137	0.41559	**0.4238**
Normalized MR	0.4144	0.41392	0.4085
Baseline	0.2724	0.40978	0.4132
MQ 2008 MAP			
	I-div	SQ	KL
Unnormalized MR	0.6439	0.6532	**0.6571**
Normalized MR	0.6449	0.6549	0.6461
Baseline	0.4513	0.6428	0.6530
MQ 2008 NDCG			
	I-div	SQ	KL
Unnormalized MR	0.7339	0.7398	**0.7451**
Normalized MR	0.7346	0.7396	0.7330
Baseline	0.5892	0.7344	0.7399
OHSUMED ERR			
	I-div	SQ	KL
Unnormalized MR	0.5657	0.5410	0.5410
Normalized MR	**0.5796**	0.5093	0.5093
Baseline	0.2255	0.5450	0.5467
OHSUMED MAP			
	I-div	SQ	KL
Unnormalized MR	**0.4537**	0.4417	0.4531
Normalized MR	0.4463	0.4394	0.4506
Baseline	0.3421	0.4465	0.4524
OHSUMED NDCG			
	I-div	SQ	KL
Unnormalized MR	**0.7000**	0.6878	0.6997
Normalized MR	0.6935	0.6798	0.6916
Baseline	0.5805	0.6892	0.6947

Table 2: Test ERR, MAP and NDCG on different datasets. The best results are in bold.

6 Conclusion and Related Work

One technique that shares our motivation of learning to rank is ordinal regression [14] which optimizes parameters

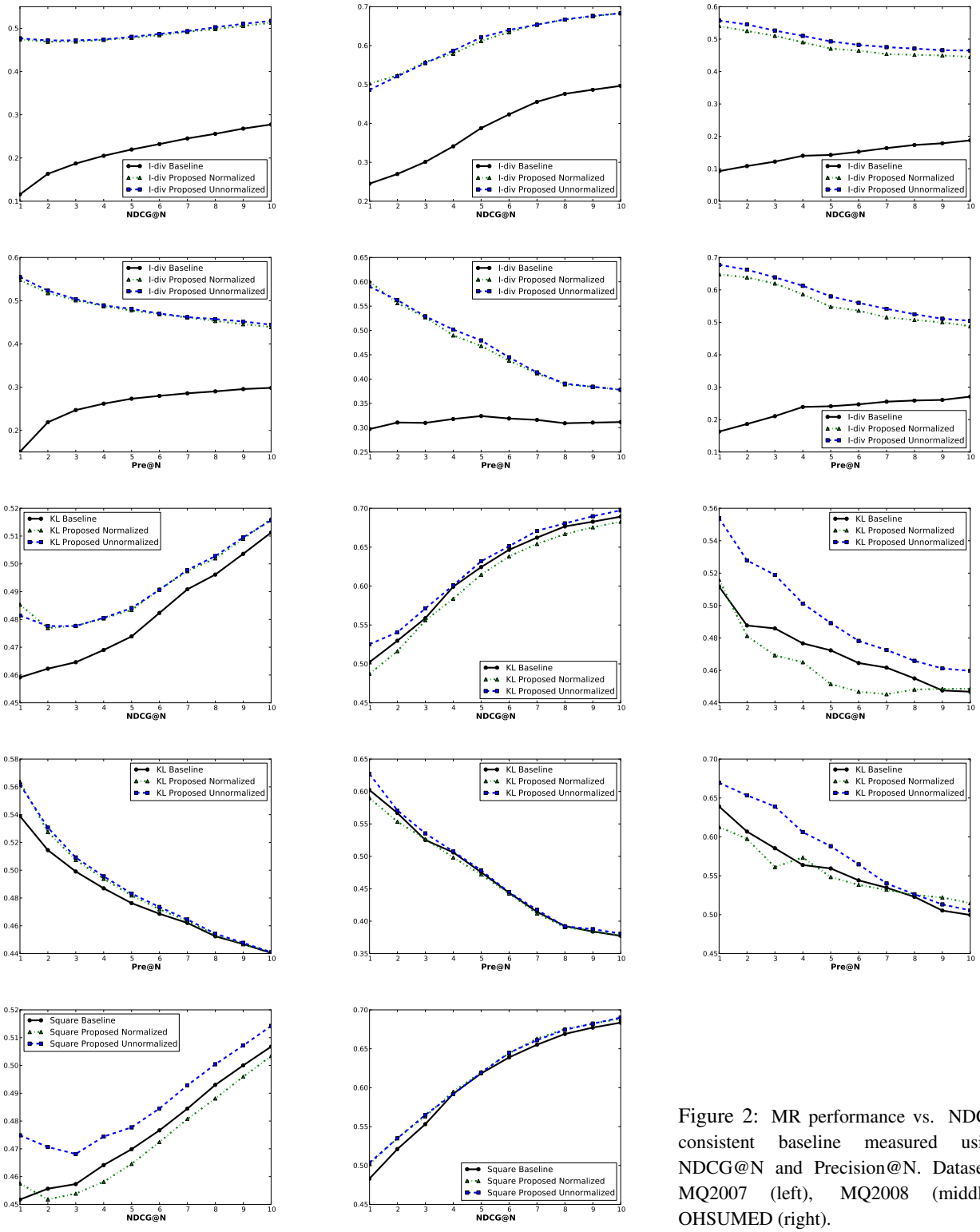


Figure 2: MR performance vs. NDCG consistent baseline measured using NDCG@N and Precision@N. Datasets: MQ2007 (left), MQ2008 (middle), OHSUMED (right).

of a regression function as well as thresholds. Unfortunately we do not have space for a full literature survey and only mention a few key differences. The log-likelihood of the classic ordinal regression methods are a sum of logarithms of differences of monotonic functions and are much more cumbersome to optimize over. To our knowledge they do not share the strong guarantees that MR with Bregman divergences enjoys. The prevalent technique there seems to require fixing a finite number of thresholds up-front, an arbitrary choice that MR does not make. However, it is not clear if that is a restriction of ordinal regression or is a prevalent practice.

In this paper we introduced a family of new cost functions for ranking. The cost function takes into account all possible monotonic transforms of the target scores, and we show how such a cost function can be optimized efficiently. Because the sole objective of learning to rank is to output good permutations on unseen data, it is desirable that the cost function be a function of such permutations. Though several permutation dependent cost functions have been proposed, they are extremely difficult to optimize over and one has to resort to surrogates and/or cut other corners. We show that with monotone retargeting with Bregman divergences such contortions are unnecessary. In addition, the proposed cost function and algorithms have very favorable statistical, optimization theoretic, as well as empirically observed properties. Other advantages include extensive parallelizability due to simple simultaneous projection updates that optimize a cost function that is convex not only in each of the arguments separately but also jointly, with a proper choice of the cost function from the family.

A Appendix: Proof Sketches

Theorem 1

Proof. Let the components of $\boldsymbol{y}^*$ take k unique values. Partition the set indexing the components into $\Pi = \{\Pi_i\}_{i=1}^k$ s.t. $\forall_j \in \Pi_i \; y_j^* = c_i \, \forall_{i \in [1,k]}$. Let the scalar mean of $\boldsymbol{x}$ on Π_i be μ_{Π_i}. By (14), $\sum_{j \in \Pi_i} D_\phi(x_j \big|\big| y_j^*) = \sum_{j \in \Pi_i} D_\phi(x_j \big|\big| \mu_{\Pi_i}) + D_\phi(\mu_{\Pi_i} \big|\big| c_i)$. First, we prove by contradiction that $y_j^* = \mu_{\Pi_i} \, \forall_j \in \Pi_i$, otherwise $\exists \; \boldsymbol{y'} \in \mathbb{R}^d$ s.t. $y'_l = y_l^* \; \forall l \notin \Pi_i$, and $\forall_j \in \Pi_i \;\; c_{i+1} \leq y'_j = c' \leq c_{i-1}$ s.t. $D_\phi\big(\mu_{\Pi_i} \big|\big| c'\big) < D_\phi\big(\mu_{\Pi_i} \big|\big| c_i\big)$. Thus $D_\phi\big(\boldsymbol{x} \big|\big| \boldsymbol{y'}\big) < D_\phi\big(\boldsymbol{x} \big|\big| \boldsymbol{y}^*\big)$, clearly a contradiction.

Let $\operatorname{Argmin}_{\boldsymbol{y} \in \mathcal{R}\downarrow} D_\theta\big(\boldsymbol{x} \big|\big| \boldsymbol{y}\big) = \boldsymbol{z}^*$ for $\boldsymbol{\theta} \neq \boldsymbol{\phi}$. Let $\boldsymbol{z}^*$ induce the partition $P = \{P_l\}_{l=1}^m$. If $\Pi = P$ always, then $y_j^* = z_j^*$ completing the proof. Shift the indexing of the partitions to the first j where Π_j, P_j differs. Now with new index, WLOG [3] assume $\Pi_1 \subset P_1$, $P_1 \subset \Pi_1 \cup \Pi_2$. Define $\dot{\Pi}_2 = \Pi_2 \cap P_1, \ddot{\Pi}_2 = \Pi_2 \setminus \dot{\Pi}_2 \neq \emptyset$, else P_1 can be refined into Π_1, Π_2 obtaining a lower cost (by Corollary 3). Let the means of $\dot{\Pi}_2, \ddot{\Pi}_2 = \dot{y}_2, \ddot{y}_2$. By definition y_2^* is their convex combination. Now $\dot{y}_2 \geq \ddot{y}_2$, else one can reduce the cost by refining Π_2 into $\dot{\Pi}_2, \ddot{\Pi}_2$. Therefore $\dot{y}_2 \geq y_2^* \geq \ddot{y}_2$. Note $y_1^* \geq \dot{y}_2$ or else we can refine P_1 into $\Pi_1, \dot{\Pi}_2$. Thus we have $y_1^* \geq \dot{y}_2 \geq \ddot{y}_2$ which is in contradiction with $y_1^* < y_2^*$. □

[3] We encourage the reader to draw a picture for clarity.

Theorem 2

Proof. For succinctness we use the abbreviations: $\boldsymbol{x}(\alpha) = \alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2$, $\boldsymbol{y}(\alpha) = \alpha \boldsymbol{y}_1 + (1-\alpha)\boldsymbol{y}_2$, $\phi_i = \phi(\boldsymbol{x}_i)$, $\psi_i = \psi(\boldsymbol{x}_i)$, $\Phi(\alpha) = \alpha\phi_1 + (1-\alpha)\phi_2$ and $\Psi(\alpha) = \alpha\psi_1 + (1-\alpha)\psi_2$. Joint convexity is equivalent to $\phi(\boldsymbol{x}(\alpha)) + \psi(\boldsymbol{y}(\alpha)) - \langle \boldsymbol{x}(\alpha), \boldsymbol{y}(\alpha) \rangle \leq \Phi(\alpha) + \Psi(\alpha) - \alpha \langle \boldsymbol{x}_1, \boldsymbol{y}_1 \rangle - (1-\alpha) \langle \boldsymbol{x}_2, \boldsymbol{y}_2 \rangle \;\; \forall \, \boldsymbol{x}_1, \boldsymbol{x}_2 \in \operatorname{dom} \phi, \; \boldsymbol{y}_1, \boldsymbol{y}_2 \in \operatorname{dom} \psi$. Thus we have to show:

$$\phi(\boldsymbol{x}(\alpha)) + \psi(\boldsymbol{y}(\alpha)) \leq \Phi(\alpha) + \Psi(\alpha) + \overbrace{\alpha(1-\alpha) \langle \boldsymbol{x}_1 - \boldsymbol{x}_2, \boldsymbol{y}_1 - \boldsymbol{y}_2 \rangle}^{B} \tag{13}$$

for all arguments in the domain. Assume with no loss in generality that $\phi(\cdot)$ and $\psi(\cdot)$ are strongly convex with modulus of strong convexity $(1+s1), (1-s2)$ with $s1 > -1$, $s2 < 1$, respectively. Reciprocal of the modulus of strong convexity of the Legendre dual is the Lipschitz constant of the gradient of a convex function [21], therefore $(1+s1) \leq \frac{1}{1-s2}$, being the lower and upper bounds of the eigenvalues of the Hessian of $\phi(\cdot)$ respectively. Simplifying expression (13) using our strong convexity assumptions and positivity of $\alpha(1-\alpha)$, we obtain that we have to show $(1+s1)||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2 + (1-s2)||\boldsymbol{y}_1 - \boldsymbol{y}_2||^2 - 2B \leq 0$. Or

$$||(\boldsymbol{x}_1 - \boldsymbol{x}_2) - (\boldsymbol{y}_1 - \boldsymbol{y}_2)||^2 + s1||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2 - s2||(\boldsymbol{y}_1 - \boldsymbol{y}_2)||^2 \leq 0.$$

Let $\boldsymbol{p} = \boldsymbol{x}_1 - \boldsymbol{x}_2$ and $\boldsymbol{q} = \boldsymbol{y}_1 - \boldsymbol{y}_2$. By choosing $(1+s)\boldsymbol{p} = \boldsymbol{q}$ we obtain $s1 > s2 + s1s2$, or equivalently $(1+s1) \geq \frac{1}{1-s2}$. Thus the lower and upper bounds of the eigenvalues of Hessian of $\phi(\cdot)$ must coincide. □

B Appendix: Optimality of Means

Theorem 3. *[2] Let $\boldsymbol{\pi}$ be a distribution over $\boldsymbol{x} \in \operatorname{dom} \phi$ and $\boldsymbol{\mu} = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \boldsymbol{\pi}} [\boldsymbol{x}]$ then the expected divergence about $\boldsymbol{s}$ is*

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \boldsymbol{\pi}} \Big[D_\phi\big(\boldsymbol{x} \big|\big| \boldsymbol{s}\big) \Big] = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \boldsymbol{\pi}} \Big[D_\phi\big(\boldsymbol{x} \big|\big| \boldsymbol{\mu}\big) \Big] + D_\phi\big(\boldsymbol{\mu} \big|\big| \boldsymbol{s}\big). \tag{14}$$

From non-negativity of Bregman divergence it follows that

Corrolary 2. *[2]* $\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \boldsymbol{\pi}} [\boldsymbol{x}] = \operatorname*{Argmin}_{\boldsymbol{y} \in \operatorname{dom} \phi} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \boldsymbol{\pi}} \Big[D_\phi\big(\boldsymbol{x} \big|\big| \boldsymbol{y}\big) \Big]$.

Corrolary 3. *If random variable $\boldsymbol{x}$ takes values in $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ with $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$ then* $\operatorname*{Argmin}_{\boldsymbol{\mu} \in \mathcal{X}} \mathop{\mathbb{E}}_{\boldsymbol{x}|\mathcal{X}} \Big[D_\phi\big(\boldsymbol{x} \big|\big| \boldsymbol{\mu}\big) \Big] \geq$

$$\operatorname*{Argmin}_{\boldsymbol{\mu}_1 \in \mathcal{X}_1} \mathop{\mathbb{E}}_{\boldsymbol{x}|\mathcal{X}_1} \Big[D_\phi\big(\boldsymbol{x} \big|\big| \boldsymbol{\mu}_1\big) \Big] + \operatorname*{Argmin}_{\boldsymbol{\mu}_2 \in \mathcal{X}_2} \mathop{\mathbb{E}}_{\boldsymbol{x}|\mathcal{X}_2} \Big[D_\phi\big(\boldsymbol{x} \big|\big| \boldsymbol{\mu}_2\big) \Big].$$

Acknowledgements

Authors acknowledge support from NSF grant IIS 1016614 and thank Cheng H. Lee for suggesting improvements over our initial submission.

References

[1] Gükhan H. Bakir, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007.

[2] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[3] Michael J. Best and Nilotpal Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47:425–439, 1990.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[5] L. M. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM.

[7] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 129–136, New York, NY, USA, 2007. ACM.

[8] Y Censor and A Lent. An iterative row-action method for interval convex programming. *Journal of Optimization Theory and Applications*, 34(3):321–353, 1981.

[9] Yair Censor. Row-action methods for huge and sparse systems and their applications. *SIAM Review*, 23:444–466, 1981.

[10] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 621–630, New York, NY, USA, 2009. ACM.

[11] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

[12] William Hersh, Chris Buckley, T. J. Leone, and David Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 192–201, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[13] Alfredo N Iusem. Steepest descent methods with generalized distances for constrained optimization. *Acta Applicande Mathematicae*, 46:225–246, 1997.

[14] Valen E. Johnson and James H. Albert. *Ordinal data modeling*. Statistics for social science and public policy. 1999.

[15] Dongmin Kim, Suvrit Sra, and Inderjit S. Dhillon. Fast projection-based methods for the least squares nonnegative matrix approximation problem. *Stat. Anal. Data Min.*, 1(1):38–51, 2008.

[16] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132, 1995.

[17] Yanyan Lan, Tie-Yan Liu, Zhiming Ma, and Hang Li. Generalization analysis of listwise learning-to-rank algorithms. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 577–584, New York, NY, USA, 2009. ACM.

[18] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March 2008.

[19] Pradeep Ravikumar, Ambuj Tewari, and Eunho Yang. On NDCG consistency of listwise ranking methods. In *Proceedings of 14th International Conference on Artificial Intelligence and Statistics*, AISTATS, 2011.

[20] R.E.Barlow and H.D.Brunk. The isotonic regression problem and its dual. *Journal of American Statistical Association*, 67(337):140–147, 1972.

[21] R T. Rockafellar. *Convex Analysis (Princeton Landmarks in Mathematics and Physics)*. Princeton University Press, December 1996.

[22] Censor Y and Zenios S. The proximal minimization algorithmwith d-functions. *Journal of Optimazation Theory and Applications*, 73:451–464, 1992.

[23] Tie yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *In Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

Markov Determinantal Point Processes

Raja Hafiz Affandi
Department of Statistics
The Wharton School
University of Pennsylvania
rajara@wharton.upenn.edu

Alex Kulesza
Dept. of Computer and Information Science
University of Pennsylvania
kulesza@cis.upenn.edu

Emily B. Fox
Department of Statistics
The Wharton School
University of Pennsylvania
ebfox@wharton.upenn.edu

Abstract

A determinantal point process (DPP) is a random process useful for modeling the combinatorial problem of subset selection. In particular, DPPs encourage a random subset $\boldsymbol{Y}$ to contain a *diverse* set of items selected from a base set $\mathcal{Y}$. For example, we might use a DPP to display a set of news headlines that are relevant to a user's interests while covering a variety of topics. Suppose, however, that we are asked to sequentially select *multiple* diverse sets of items, for example, displaying new headlines day-by-day. We might want these sets to be diverse not just individually but also through time, offering headlines today that are unlike the ones shown yesterday. In this paper, we construct a *Markov* DPP (M-DPP) that models a *sequence* of random sets $\{\boldsymbol{Y}_t\}$. The proposed M-DPP defines a stationary process that maintains DPP margins. Crucially, the induced union process $\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1}$ is also marginally DPP-distributed. Jointly, these properties imply that the sequence of random sets are encouraged to be diverse both at a given time step as well as across time steps. We describe an exact, efficient sampling procedure, and a method for incrementally learning a quality measure over items in the base set $\mathcal{Y}$ based on external preferences. We apply the M-DPP to the task of sequentially displaying diverse and relevant news articles to a user with topic preferences.

1 INTRODUCTION

Consider the combinatorial problem of subset selection. Binary Markov random fields are commonly applied in this setting, and in the case of positive correlations, yield subsets that favor similar items. However, in many applications there is naturally a sense of *repulsion*. For example, repulsive processes arise in nature—trees tend to grow in the least occupied space (Neeff et al., 2005), and ant hill locations are likewise over-dispersed relative to uniform placement (Bernstein and Gobbel, 1979). Likewise, many practical tasks can be posed in terms of diverse subset selection. For example, one might want to select a set of frames from a movie that are representative of its content. Clearly, diversity is preferable to avoid redundancy; likewise, each frame should be of high quality. A motivating example we consider throughout the paper is the task of selecting a diverse yet relevant set of news headlines to display to a user. One could imagine employing binary Markov random fields with negative correlations, but such models often involve notoriously intractable inference problems.

Determinantal point processes (DPPs), which arise in random matrix theory (Mehta and Gaudin, 1960; Ginibre, 1965) and quantum physics (Macchi, 1975), are a class of *repulsive processes* and are natural models for subset selection problems where diversity is preferred. DPPs define the probability of a subset in terms of the determinant of a kernel submatrix, and with an appropriate definition of the kernel matrix they can be interpreted as inherently balancing quality and diversity. DPPs are appealing in practice since they offer interpretability and tractable algorithms for exact inference. For example, one can compute marginal and conditional probabilities and perform exact sampling. DPPs have recently been employed for human pose estimation, search diversification, and document summarization (Kulesza and Taskar, 2010, 2011a,b).

In this paper, our focus is instead on modeling diverse *sequences* of subsets. For example, in displaying news headlines from day to day, one aims to select articles that are relevant and diverse on any given day. Additionally, it is desirable to select articles that are diverse relative to those previously shown. We construct a *Markov* DPP (M-DPP) for a sequence of random sets $\{\boldsymbol{Y}_t\}$. The proposed M-DPP defines a stationary process that maintains DPP margins, implying that $\boldsymbol{Y}_t$ is encouraged to be diverse at time t. Crucially, the induced union process $\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1}$ is also marginally DPP-distributed. Since this property implies the diversity of $\boldsymbol{Z}_t$, in addition to the individual diversity of $\boldsymbol{Y}_t$ and $\boldsymbol{Y}_{t-1}$, we conclude that $\boldsymbol{Y}_t$ is diverse from $\boldsymbol{Y}_{t-1}$.

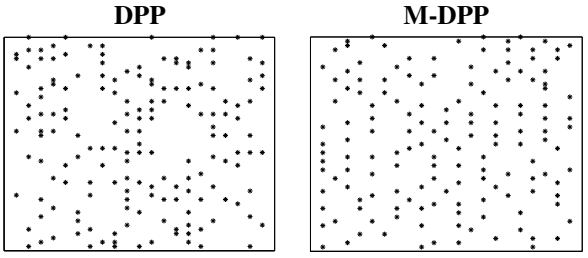


Figure 1: A set of points on a line (y axis) drawn from a DPP independently over time (left) and from a M-DPP (right). While DPP points are diverse only within time steps (columns), M-DPP points are also diverse across time steps.

For an illustration of the improved overall diversity when sampling from a M-DPP rather than independent sequential sampling from a DPP, see Fig. 1.

Our specific construction of the M-DPP yields an exact sampling procedure that can be performed in polynomial time. Additionally, we explore a method for incrementally learning the quality of each item in the base set $\mathcal{Y}$ based on externally provided preferences. In particular, a decomposition of the DPP kernel matrix has an interpretation as defining the quality of each item and pairwise similarities between items. Our incremental learning procedure assumes a well-defined similarity metric and aims to learn features of items that a user deems as preferable. These features are used to define the quality scores for each item. The M-DPP aids in the exploration of items of interest to the user by providing sequentially diverse results.

We study the empirical behavior of the M-DPP on a news task where the goal is to display diverse and high quality articles. Compared to choosing articles based on quality alone or to sampling from an independent DPP at each time step, we show that the M-DPP produces articles that are significantly more diverse across time steps without large sacrifices in quality. Furthermore, within a time step the M-DPP chooses articles with diversity comparable to that of the independent DPP; this is a direct result of the fact that the M-DPP maintains DPP margins. We also consider learning the quality function over time from the feedback of a user with topic preferences. In this setting, the M-DPP returns high quality results that are preferred by the user while simultaneously exploring the topic space more quickly than baseline methods, leading to improved coverage.

2 DETERMINANTAL POINT PROCESSES

A random point process $\mathcal{P}$ on a discrete base set $\mathcal{Y} = \{1, \ldots, N\}$ is a probability measure on the set $2^{\mathcal{Y}}$ of all subsets of $\mathcal{Y}$. Let K be a semidefinite matrix with rows and columns indexed by the elements of $\mathcal{Y}$. $\mathcal{P}$ is called a determinantal point process (DPP) if there exists $K \preceq I$ (all eigenvalues less than or equal to 1) such that if $\boldsymbol{Y}$ is a random set drawn according to $\mathcal{P}$, then for every $A \subseteq \mathcal{Y}$:

$$\mathcal{P}(\boldsymbol{Y} \supseteq A) = \det(K_A) \,. \tag{1}$$

Here, $K_A \equiv [K_A]_{i,j \in A}$ denotes the submatrix of K indexed by elements in A, and we adopt the convention that $\det(K_\emptyset) = 1$. We will refer to K as the marginal kernel. If we think of K_{ij} as measuring the similarity between items i and j, then

$$\mathcal{P}(\boldsymbol{Y} \supseteq \{i, j\}) = K_{ii}K_{jj} - K_{ij}^2 \tag{2}$$

implies that $\boldsymbol{Y}$ is unlikely to contain both i and j when they are very similar; that is, a DPP can be seen as modeling a collection of diverse items from the base set $\mathcal{Y}$.

DPPs can alternatively be constructed via L-ensembles (Borodin and Rains, 2005). An L-ensemble is a probability measure on $2^{\mathcal{Y}}$ defined via a positive semidefinite matrix L indexed by elements of $\mathcal{Y}$:

$$\mathcal{P}_L(\boldsymbol{Y} = A) = \frac{\det(L_A)}{\det(L + I)} \,, \tag{3}$$

where I is the $N \times N$ identity matrix. It can be shown that an L-ensemble is a DPP with marginal kernel $K = L(I+L)^{-1}$. Conversely, a DPP with marginal kernel K has L-ensemble kernel $L = K(I - K)^{-1}$ (when the inverse exists).

An intuitive way to think of the L-ensemble kernel L is as a Gram matrix (Kulesza and Taskar, 2010):

$$L_{ij} = q_i \phi_i^\top \phi_j q_j \,, \tag{4}$$

interpreting $q_i \in \mathbb{R}^+$ as representing the intrinsic *quality* of an item i, and $\phi_i, \phi_j \in \mathbb{R}^n$ as unit length feature vectors representing the *similarity* between items i and j with $\phi_i^\top \phi_j \in [-1, 1]$. Under this framework, we can model quality and similarity separately to encourage the DPP to choose *high quality* items that are *dissimilar* to each other. This is very useful in many applications. For example, in response to a search query we can provide a very relevant (i.e. high quality) but diverse (i.e. dissimilar) list of results.

Conditional DPPs For any $A, B \subseteq \mathcal{Y}$ with $A \cap B = \emptyset$, it is straightforward to show that

$$\mathcal{P}_L(\boldsymbol{Y} = A \cup B | \boldsymbol{Y} \supseteq A) = \frac{\det(L_{A \cup B})}{\det(L + I_{\mathcal{Y} \setminus A})} \,, \tag{5}$$

where $I_{\mathcal{Y} \setminus A}$ is a matrix with ones on the diagonal entries indexed by the elements of $\mathcal{Y} \setminus A$ and zeros elsewhere.

This conditional distribution is itself a DPP over the elements of $\mathcal{Y} \setminus A$ (Borodin and Rains, 2005). In particular, suppose $\boldsymbol{Y}$ is DPP-distributed with L-ensemble kernel L, and condition on the fact that $\boldsymbol{Y} \supseteq A$. Then the set $\boldsymbol{Y} \setminus A$ is DPP-distributed with marginal and L-ensemble kernels

$$K^A = \left[I - (L + I_{\mathcal{Y} \setminus A})^{-1}\right]_{\mathcal{Y} \setminus A} \tag{6}$$

$$L^A = \left(\left[(L + I_{\mathcal{Y} \setminus A})^{-1}\right]_{\mathcal{Y} \setminus A}\right)^{-1} - I \,. \tag{7}$$

Here, $[\cdot]_{\mathcal{Y}\setminus A}$ denotes the submatrix of the argument indexed by elements in $\mathcal{Y} \setminus A$. Thus, DPPs as a class are closed under most natural conditioning operations.

In selecting a diverse collection of elements in $\mathcal{Y}$, a DPP jointly models both the size of a set and its content. In some applications, the goal is to select (diverse) sets of a fixed size. In order to achieve this goal, we can instead consider a fixed-size determinantal point processes, or kDPP (Kulesza and Taskar, 2011a), which gives a distribution over all random subsets $\boldsymbol{Y} \subseteq \mathcal{Y}$ with fixed cardinality k. The L-ensemble construction of a kDPP, denoted $\mathcal{P}_L^k$, gives probabilities

$$\mathcal{P}_L^k(\boldsymbol{Y} = A) = \frac{\det(L_A)}{\sum_{|B|=k} \det(L_B)} \tag{8}$$

for all sets A with cardinality k and any positive semidefinite kernel L. Kulesza and Taskar (2011a) developed efficient algorithms to normalize, sample and marginalize kDPPs using properties of elementary symmetric polynomials.

3 MARKOV DETERMINANTAL POINT PROCESSES (M-DPPS)

In certain applications, such as in the task of displaying news headlines, our goal is not only to generate a diverse collection of items at one time point, but also to generate collections of items at subsequent time points that are both highly relevant and dissimilar to the previous collection. To address these goals, we introduce the Markov determinantal point process (M-DPP), which emphasizes both marginal and conditional diversity of selected items. Harnessing the *quality* and *similarity* interpretation of the DPP in (4), the M-DPP provides a dynamic way of selecting high quality and diverse collections of items as a temporal process.

We constructively define a first-order, discrete-time autoregressive point process on $\mathcal{Y}$ by specifying a Markov transition distribution (and initial distribution). Throughout, we use the notation $\{\boldsymbol{Y}_t\}$, $\boldsymbol{Y}_t \subseteq \mathcal{Y}$, to represent a sequence of sets following a M-DPP. We consider two such constructions: one based on marginal kernels, and the other on L-ensembles. Both yield equivalent stationary processes with DPP margins. Additionally, and quite intuitively, the induced union process $\{\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1}\}$ has DPP margins with a closely related kernel. Combining these two properties, we conclude that the constructed M-DPPs yield a sequence of sets $\{\boldsymbol{Y}_t\}$ that are diverse at any time t and across time steps $t, t-1$.

Marginal construction. Define $\mathcal{P}(\boldsymbol{Y}_1 \supseteq A) = \det(K_A)$ and

$$\mathcal{P}(\boldsymbol{Y}_t \supseteq B | \boldsymbol{Y}_{t-1} \supseteq A) = \frac{\det(K_{A\cup B})}{\det(K_A)} , \tag{9}$$

where $K \prec \frac{1}{2}I$ and $A \cap B = \emptyset$. Throughout, we adopt the implicit constraint that $\boldsymbol{Y}_t \cap \boldsymbol{Y}_{t-1} = \emptyset$. We have immediately the joint probability

$$\mathcal{P}(\boldsymbol{Y}_2 \supseteq B, \boldsymbol{Y}_1 \supseteq A) = \det(K_{A\cup B}) , \tag{10}$$

and therefore

$$\mathcal{P}(\boldsymbol{Y}_2 \supseteq B) = \mathcal{P}(\boldsymbol{Y}_2 \supseteq B, \boldsymbol{Y}_1 \supseteq \emptyset) = \det(K_B) . \tag{11}$$

Inductively, the process is stationary and marginally DPP.

Finally, we have the union of consecutive sets:

$$\begin{aligned}
&\mathcal{P}(\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1} \supseteq C) \\
&\quad = \sum_{A\subseteq C} \mathcal{P}(\boldsymbol{Y}_t \supseteq C \setminus A, \boldsymbol{Y}_{t-1} \supseteq A) = \sum_{A\subseteq C} \det(K_C) \\
&\quad = 2^{|C|} \det(K_C) = \det((2K)_C) .
\end{aligned} \tag{12}$$

That is, $\boldsymbol{Z}_t$ is marginally distributed as a DPP with marginal kernel $2K$. Since a randomly sampled subset of a DPP-distributed set also follows a DPP, marginally we can imagine this process as sampling $\boldsymbol{Z}_t$ and then splitting its elements randomly into two sets, $\boldsymbol{Y}_{t-1}$ and $\boldsymbol{Y}_t$.

L-ensemble construction. The above is appealingly simple, but the marginal form of the conditional in (9) is not particularly conducive to a sequential sampling process. Instead, we can rewrite everything as L-ensembles. Assume that at the first time step $\mathcal{P}(\boldsymbol{Y}_1 = Y_1) = \frac{\det(L_{Y_1})}{\det(L+I)}$ and define the transition distribution as

$$\mathcal{P}(\boldsymbol{Y}_t = Y_t | \boldsymbol{Y}_{t-1} = Y_{t-1}) = \frac{\det(M_{Y_t\cup Y_{t-1}})}{\det(M + I_{\mathcal{Y}\setminus Y_{t-1}})} , \tag{13}$$

for $M = L(I-L)^{-1}$. Note that the transition distribution is essentially a conditional DPP with L-ensemble kernel M (Eq. (5)). M is well-defined as long as $L \prec I$, which is equivalent to $K \prec \frac{1}{2}I$, as in the marginal construction.

Now we have the joint probability

$$\mathcal{P}(\boldsymbol{Y}_2 = Y_2, \boldsymbol{Y}_1 = Y_1) = \frac{\det(M_{Y_1\cup Y_2})}{\det(M + I_{\mathcal{Y}\setminus Y_1})} \frac{\det(L_{Y_1})}{\det(L+I)} . \tag{14}$$

Using the fact that $\det(M + I_{\mathcal{Y}\setminus Y_1})/\det(M+I) = \det(L_{Y_1})$,

$$\mathcal{P}(\boldsymbol{Y}_2 = Y_2, \boldsymbol{Y}_1 = Y_1) = \frac{1}{\det(L+I)} \frac{\det(M_{Y_1\cup Y_2})}{\det(M+I)} . \tag{15}$$

Therefore, marginally,

$$\begin{aligned}
\mathcal{P}(\boldsymbol{Y}_2 = Y_2) &= \sum_{Y_1\subseteq\mathcal{Y}} \frac{1}{\det(L+I)} \frac{\det(M_{Y_1\cup Y_2})}{\det(M+I)} \\
&= \sum_{(Y_1\cup Y_2)\supseteq Y_2} \frac{1}{\det(L+I)} \frac{\det(M_{Y_1\cup Y_2})}{\det(M+I)} \\
&= \frac{\det(M + I_{\mathcal{Y}\setminus Y_2})}{\det(L+I)\det(M+I)} = \frac{\det(L_{Y_2})}{\det(L+I)} .
\end{aligned} \tag{16}$$

Here, we used $\sum_{B \supseteq A} \det(M_B) = \det(M + I_{\mathcal{Y} \setminus A})$, which is immediately derived from (5). By induction, we conclude

$$\mathcal{P}(\boldsymbol{Y}_t = Y_t) = \frac{\det(L_{Y_t})}{\det(L+I)} \,. \tag{17}$$

Thus, our construction yields a stationary process with $\boldsymbol{Y}_t$ marginally distributed as a DPP with L-ensemble kernel L.

One can likewise analyze the margin of the induced union process $\{\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1}\}$:

$$\begin{aligned}
&\mathcal{P}(\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1} = C) \\
&\quad = \sum_{A \subseteq C} \mathcal{P}(\boldsymbol{Y}_t = C \setminus A, \boldsymbol{Y}_{t-1} = A) \\
&\quad = \sum_{A \subseteq C} \frac{1}{\det(L+I)} \frac{\det(M_C)}{\det(M+I)} \\
&\quad = \frac{2^{|C|}}{\det(L+I)} \frac{\det(M_C)}{\det(M+I)} \qquad (18) \\
&\quad = \frac{1}{\det(L+I)} \frac{\det((2M)_C)}{\det(M+I)} \,. \qquad (19)
\end{aligned}$$

Noting that

$$\begin{aligned}
&\det(M+I)\det(L+I) = \det((M+I)(L+I)) \\
&\quad = \det((M+I)(I-(M+I)^{-1}+I)) \\
&\quad = \det(2M+2I-I) = \det(2M+I) \,, \qquad (20)
\end{aligned}$$

we conclude

$$\mathcal{P}(\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1} = C) = \frac{\det((2M)_C)}{\det(2M+I)} \,. \tag{21}$$

We have shown that $\boldsymbol{Z}_t$ is marginally distributed as a DPP with L-ensemble kernel $2M$. The corresponding marginal kernel is

$$\begin{aligned}
2M(2M+I)^{-1} &= 2L(I-L)^{-1}\left[(L+I)(I-L)^{-1}\right]^{-1} \\
&= 2L(L+I)^{-1} = 2K \,. \qquad (22)
\end{aligned}$$

Thus, we have reproduced the same characterization of $\boldsymbol{Z}_t$ as in (12) for the marginal kernel construction.

To summarize the marginal properties of the M-DPP, using the notation $\boldsymbol{Y} \sim L, K$ to denote that $\boldsymbol{Y}$ is from a DPP with L-ensemble kernel L and marginal kernel K, we have:

$$\boldsymbol{Y}_t \sim L, K \tag{23}$$

$$\boldsymbol{Z}_t \sim 2L(I-L)^{-1}, 2K \,. \tag{24}$$

3.1 COMMENTS ON THE M-DPP

While we have shown that the M-DPP subsets are diverse at subsequent time steps, this does not necessarily imply diversity at longer intervals. In fact, it is possible for realizations to have oscillations, where groups of high-quality items recur every two (or more) time steps. However, this is not necessarily a problem in practice for several reasons. First, the M-DPP construction straightfowardly extends to higher order models with longer memory, if desired. Second, it is possible to show that the M-DPP does not *harm* long-term diversity relative to independent sampling from a DPP. If we make a separate copy of each item at each time step, then the M-DPP can be seen as a large DPP on item/time pairs (i,t). Denoting the marginal kernel by $\hat{K}$, the Markov property implies that $\hat{K}_{(i,t)(j,u)}$ is only nonzero when $|t-u| \leq 1$. The probability of item i appearing at any set of time steps, given by the appropriate determinant of $\hat{K}$, can only be reduced by the off-diagonal entries compared to independent DPP samples at each time step. Thus, the M-DPP can only make global repetition less likely. Finally, in our experiments (Sec. 5) we report results that suggest that M-DPP oscillations do not arise in the task we study.

3.2 MARKOV kDPPS

One can also construct a *Markov kDPP* (M-kDPP). Although we define a stationary process, our construction does not yield $\boldsymbol{Y}_t$ marginally kDPP. Instead, the M-kDPP simply ensures that $\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1}$ follows a $2k$DPP. Since $\boldsymbol{Z}_t$ is encouraged to be diverse, the subsets $\boldsymbol{Y}_t$ and $\boldsymbol{Y}_{t-1}$ will likewise be diverse despite not following a kDPP themselves.

We start by defining the margin and transition distributions:

$$\mathcal{P}(\boldsymbol{Y}_{t-1} = Y_{t-1}) = \frac{\sum_{|A|=k} \det(L_{Y_{t-1} \cup A})}{\binom{2k}{k} \sum_{|B|=2k} \det(L_B)} \tag{25}$$

$$\mathcal{P}(\boldsymbol{Y}_t = Y_t | \boldsymbol{Y}_{t-1} = Y_{t-1}) = \frac{\det(L_{Y_{t-1} \cup Y_t})}{\sum_{|A|=k} \det(L_{Y_{t-1} \cup A})} \,, \tag{26}$$

where A and Y_t are disjoint from Y_{t-1}. Then, jointly

$$\mathcal{P}(\boldsymbol{Y}_t = Y_t, \boldsymbol{Y}_{t-1} = Y_{t-1}) = \frac{\det(L_{Y_{t-1} \cup Y_t})}{\binom{2k}{k} \sum_{|B|=2k} \det(L_B)} \,, \tag{27}$$

from which we confirm the stationarity of the process:

$$\mathcal{P}(\boldsymbol{Y}_t = Y_t) = \frac{\sum_{|Y_{t-1}|=k} \det(L_{Y_t \cup Y_{t-1}})}{\binom{2k}{k} \sum_{|B|=2k} \det(L_B)} \,. \tag{28}$$

The implied union process has margins

$$\begin{aligned}
&\mathcal{P}(\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1} = C) \\
&\quad = \sum_{A \subseteq C, |A|=k} \mathcal{P}(\boldsymbol{Y}_t = C \setminus A, \boldsymbol{Y}_{t-1} = A) \\
&\quad = \sum_{A \subseteq C, |A|=k} \frac{\det(L_C)}{\binom{2k}{k} \sum_{|B|=2k} \det(L_B)} \\
&\quad = \frac{\det(L_C)}{\sum_{|B|=2k} \det(L_B)} \,, \qquad (29)
\end{aligned}$$

which is a $2k$DPP with L-ensemble kernel L.

Algorithm 1 Sampling from a DPP

Input: L-ensemble kernel matrix L
$\{(v_n, \lambda_n)\}_{n=1}^N \leftarrow$ eigenvector/value pairs of L
$J \leftarrow \emptyset$
for $n = 1, \ldots, N$ **do**
  $J \leftarrow J \cup \{n\}$ with prob. $\frac{\lambda_n}{\lambda_n + 1}$
$V \leftarrow \{v_n\}_{n \in J}$
$Y \leftarrow \emptyset$
while $|V| > 0$ **do**
  Select y_i from $\mathcal{Y}$ with $\Pr(y_i) = \frac{1}{|V|} \sum_{v \in V} (v^\top \boldsymbol{e}_i)^2$
  $Y \leftarrow Y \cup \{y_i\}$
  $V \leftarrow V_\perp$, an orthonormal basis for the subspace of V orthogonal to e_i
Output: Y

3.3 SAMPLING FROM M-DPPS AND M-kDPPS

In the previous subsections we showed how our constructions of M-(k)DPPs lead to DPP (and DPP-like) marginals for $\{\boldsymbol{Y}_t\}$ and the union process $\{\boldsymbol{Z}_t\}$. These connections to DPPs give us valuable intuition about the diversity induced both within and across time steps. They serve another purpose as well: since DPPs and kDPPs can be sampled in polynomial time, we can leverage existing algorithms to efficiently sample from M-DPPs and M-kDPPs.

Hough et al. (2006) first described the DPP sampling algorithm shown in Algorithm 1. The first step is to compute an eigendecomposition $L = \sum_{n=1}^N \lambda_n v_n v_n^\top$ of the kernel matrix; from this, a random subset V of the eigenvectors is chosen by using the eigenvalues to bias a sequence of coin flips. The algorithm then proceeds iteratively, on each iteration selecting a new item y_i to add to the sample and then updating V in a manner that de-emphasizes items similar to the one just selected. Note that $\boldsymbol{e}_i$ is the ith elementary basis vector whose elements are all zero except for a one in position i. Algorithm 1 runs in time $O(N^3 + Nk^3)$, where N is the number of available items and k is the cardinality of the returned sample.

To adapt this algorithm for sampling M-DPPs, we will proceed sequentially, first sampling $\boldsymbol{Y}_1$ from the initial distribution and then repeatedly selecting $\boldsymbol{Y}_t$ from the transition distribution given $\boldsymbol{Y}_{t-1}$. The initial distribution is a DPP with L-ensemble kernel L and can therefore be sampled directly using Algorithm 1. As shown in Sec. 3, the transition distribution (13) is a conditional DPP with L-ensemble kernel $M = L(L - I)^{-1}$; using (7), the L-ensemble kernel for $\boldsymbol{Y}_t$ given $\boldsymbol{Y}_{t-1} = Y_{t-1}$ can be written as

$$L^{(t)} = \left((M + I_{\mathcal{Y} \setminus Y_{t-1}})^{-1}_{\mathcal{Y} \setminus Y_{t-1}} \right)^{-1} - I \,. \qquad (30)$$

Thus we can sample simply and efficiently from a M-DPP using Algorithm 2. The runtime is $O(TN^3 + TNk_{\max}^3)$, where $k_{\max}$ is the maximum number of items chosen at a single time step. Note that for constant $k_{\max}$ this is the same runtime as a Kalman filter with a state vector of size N.

Algorithm 2 Sampling from a Markov DPP

Input: matrix L
$M \leftarrow L(L - I)^{-1}$
$Y_1 \leftarrow$ DPP-SAMPLE(L)
for $t = 2, \ldots, T$ **do**
  $L^{(t)} \leftarrow \left((M + I_{\mathcal{Y} \setminus Y_{t-1}})^{-1}_{\mathcal{Y} \setminus Y_{t-1}} \right)^{-1} - I$
  $Y_t \leftarrow$ DPP-SAMPLE$(L^{(t)})$
Output: $\{Y_t\}$

Algorithm 3 Sampling from a kDPP

Input: L-ensemble kernel matrix L, size k
$\{(v_n, \lambda_n)\}_{n=1}^N \leftarrow$ eigenvector/value pairs of L
$J \leftarrow \emptyset$
for $n = N, \ldots, 1$ **do**
  if $u \sim U[0,1] < \lambda_n \frac{e_{k-1}^{n-1}}{e_k^n}$ **then**
    $J \leftarrow J \cup \{n\}$
    $k \leftarrow k - 1$
    if $k = 0$ **then**
      break
{continue with the rest of Algorithm 1}

Kulesza and Taskar (2011a) proved that a modification to the first loop in Algorithm 1 allows sampling from a kDPP with no change in the asymptotic complexity. The result is Algorithm 3; here e_k^n denotes the elementary symmetric polynomial $e_k^n = \sum_{|J|=k} \prod_{n \in J} \lambda_n$, which can be computed efficiently using recursion.

We can now use Algorithm 3 to perform sequential sampling for a M-kDPP. At first glance, the initial distribution (which is not a kDPP) seems difficult to sample; however, from Sec. 3 we know that it can be obtained by harnessing the union process form of (29) and first sampling a $2k$DPP with L-ensemble kernel L and then throwing away half of the resulting items at random. Transitionally, we have a conditional kDPP whose kernel can be computed as in (30). Algorithm 4 summarizes the M-kDPP sampling process, which runs in time $O(TN^3 + TNk^3)$.

Algorithm 4 Sampling from a Markov kDPP

Input: matrix L, size k
$Z_1 \leftarrow k$DPP-SAMPLE$(L, 2k)$
$Y_1 \leftarrow$ random half of Z_1
for $t = 2, \ldots, T$ **do**
  $L^{(t)} \leftarrow \left((L + I_{\mathcal{Y} \setminus Y_{t-1}})^{-1}_{\mathcal{Y} \setminus Y_{t-1}} \right)^{-1} - I$
  $Y_t \leftarrow k$DPP-SAMPLE$(L^{(t)}, k)$
Output: $\{Y_t\}$

4 LEARNING USER PREFERENCES

A broad class of problems suited to M-(k)DPP modeling are also applications in which we would like to learn preferences from a user over time. Recall the news headlines scenario. Here, the goal is to present articles on a daily basis that are both relevant to the user's interests and also non-redundant. With feedback from a user in the form of click-through behavior, we can attempt to simultaneously learn features of the articles that the user regards as preferable. While the diversity offered by a M-DPP is intrinsically valuable for this task, e.g., to keep the user from getting bored, in the context of learning it also has an important secondary benefit: it promotes exploration of the preference space.

Consider the following simple learning setup. At each time step t, the algorithm shows the user a set of k items drawn from some base set $\mathcal{Y}_t$, for instance, articles from the day's news. The user then provides feedback by identifying each shown item as either preferred or not preferred, perhaps by clicking on the preferred ones. The algorithm then incorporates this feedback and proceeds to the next round. The learner has two goals. First, as often as possible at least some of the items shown to the user should be preferred. Second, over the long term, many different items preferred by the user should be shown. In other words, the algorithm should not focus on a small set of preferred items.

Perhaps the most important consideration in this framework is balancing showing articles that the user is known to like (*exploitation*) against showing a variety of articles so as to discover new topics in which the user is also interested (*exploration*). Neither extreme is likely to be successful. However, using the L-ensemble kernel decomposition in (4), a DPP seeks to propose sets of items that are simultaneously high quality and diverse. The M-DPP takes this a step further and encourages diversity from step to step while maintaining DPP margins, exposing the user to an even greater variety of items without significantly sacrificing quality. Thus, we might expect that M-(k)DPPs can be used to enable fast and successful learning in this setting.

The tradeoff between exploration and exploitation is a fundamental issue for interactive learning, and has received extensive treatment in the literature on multi-armed bandits. However, our setup is relatively unusual for two reasons. First, we show multiple items per time step, sometimes called the *multiple plays* setting (Anantharam et al., 1987). Second, we use feature vectors to describe the items we choose, allowing us to generalize to unseen items (e.g., new articles); this is a special case of *contextual bandits* (Langford and Zhang, 2007). Each of these scenarios has received some attention on its own, but it is only in combination that a notion of diversity becomes relevant, since we have both the need to select multiple items as well as a basis for relating them. This combination has been considered recently by Yue and Guestrin (2011), who showed an algorithm that yields bounded regret under the assumption that the reward function is submodular. Here, on the other hand, our goal is primarily to illustrate the empirical effects on learning when the items shown at each time step are sampled from a M-DPP. To that end, we propose a very simple quality learning algorithm that appears to work well in practice. Whether formal regret guarantees can be established for learning with M-DPPs is an open question for future work.

4.1 SETUP

To naturally accommodate user feedback and transfer knowledge across items, we will consider algorithms that learn a log-linear quality model assigning item i the score

$$q_i = \exp(\theta^\top f_i)\ , \tag{31}$$

where $f_i \in \mathbb{R}^m$ is a feature vector for item i and $\theta \in \mathbb{R}^m$ is the parameter vector to be learned. Learning iterates between two distinct steps: (1) sampling articles according to the current quality scores and (2) using user feedback to revise the quality scores via updates to θ.

Let $\theta^{(t)}$ denote the parameter vector prior to time step t and let $q_i^{(t)}$ denote the corresponding quality scores for the items $i \in \mathcal{Y}_t$. We initialize $\theta^{(1)} = \mathbf{0}$ so that $q_i^{(1)} = 1$ for all $i \in \mathcal{Y}_1$. (At this point we are effectively in a purely exploratory mode.) Denote the items preferred by the user at iteration t by $\{a_i^{(t)}\}_{i=1}^{R_t}$, and the non-preferred items by $\{b_i^{(t)}\}_{i=1}^{S_t}$. Inspired by standard online algorithms, we define the parameter update rule as follows:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \eta\left(\frac{1}{R_t}\sum_{i=1}^{R_t} f_{a_i^{(t)}} - \frac{1}{S_t}\sum_{i=1}^{S_t} f_{b_i^{(t)}}\right) \tag{32}$$

That is, we add to θ the average features of the preferred items, and subtract from θ the average features of non-preferred items. This increases the quality of the former and decreases the quality of the latter. η is a learning rate hyperparameter. We can then proceed to the next time step, computing the new quality scores $q_i^{(t+1)} = \exp(\theta^{(t+1)\top} f_i)$ for each $i \in \mathcal{Y}_{t+1}$.

The updated quality scores are then used to select subsequent items to be shown to the user. In order to separate the challenges of learning the quality scores, which is not our primary interest, from the benefits of incorporating the M-DPP, we consider five sampling methods:

- **Uniform.** We ignore the quality scores and choose k items uniformly at random without replacement.
- **Weighted.** We draw k items with probabilities proportional to their quality scores without replacement.
- k**DPP.** We sample the set of items from a kDPP with L-ensemble kernel L given by the decomposition in

Algorithm 5 Interactive learning of quality scores

Input: learning rate η
$\theta^{(1)} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  $q_i^{(t)} \leftarrow \exp(\theta^{(t)\top} f_i) \quad \forall i \in \mathcal{Y}_t$
  Select items to display given $q_i^{(t)}$
    (using one of the methods described in Sec. 4.1)
  Receive user feedback $\{a_i^{(t)}\}_{i=1}^{R_t}$ and $\{b_i^{(t)}\}_{i=1}^{S_t}$
  $\theta^{(t+1)} \leftarrow \theta^{(t)} + \eta \left(\frac{1}{R_t} \sum_{i=1}^{R_t} f_{a_i^{(t)}} - \frac{1}{S_t} \sum_{i=1}^{S_t} f_{b_i^{(t)}} \right)$

(4), where ϕ is fixed in advance and q_i are the current quality scores.

- k**DPP + heuristic (threshold).** We sample the set of items from a kDPP after removing articles whose similarity to the previously selected articles exceeds a predetermined threshold. At threshold > 1, the heuristic is equivalent to the kDPP.

- **M-kDPP.** We sample the set of items from the M-kDPP transition distribution given the items selected at the previous time step. The L-ensemble transition kernel is as in (30), with L defined as for the kDPP.

The learning algorithm is summarized in Algorithm 5.

4.2 LIKELIHOOD-BASED ALTERNATIVE

Instead of the additive learning rule proposed above, one could instead take advantage of the probabilistic nature of the M-DPP and perform likelihood-based learning, which has associated theoretical guarantees. In particular, based on a sequence of user feedback, we could solve for the penalized DPP maximum likelihood estimate of $q^{(t)} = [q_1^{(t)}, \ldots, q_N^{(t)}]$ as:

$$\arg\max_q \prod_{t=1}^{t} \mathcal{P}_q(\{a_i^{(t)}\} \subseteq \boldsymbol{Y}_t, \{b_i^{(t)}\} \cap \boldsymbol{Y}_t = \emptyset) + \lambda ||q||_2, \tag{33}$$

where $\mathcal{P}_q$ is a DPP with L-ensemble kernel defined by quality scores q and λ is a regularization parameter. We have

$$\mathcal{P}_q(\{a_i^{(t)}\} \subseteq \boldsymbol{Y}_t, \{b_i^{(t)}\} \cap \boldsymbol{Y}_t = \emptyset) = \left(1 - \mathcal{P}_q(\{b_i^{(t)}\} \subseteq \boldsymbol{Y}_t \mid \{a_i^{(t)}\} \subseteq \boldsymbol{Y}_t)\right) \cdot \mathcal{P}_q(\{a_i^{(t)}\} \subseteq \boldsymbol{Y}_t), \tag{34}$$

which has computable terms in a DPP given the quality scores q. The M-DPP has obvious extensions. However, in both cases the objective function is not convex so computations are intensive and only converge to local maxima. Due to its simplicity and good performance in practice (see Sec. 5.2), we use the heuristic algorithm described previously for illustrating the behavior of the M-DPP.

5 EXPERIMENTS

We study the performance of the M-kDPP for selecting daily news items from a selection of over 35,000 New York Times newswire articles obtained between January and June of 2005 as part of the Gigaword corpus (Graff and Cieri, 2009). On each day of a given week, we display 10 articles from a base set of the roughly 1400 articles written that week. This process is repeated for each of the 26 weeks in our dataset. The goal is to choose a collection of articles that is high quality but also diverse, both marginally and between time steps.

To examine performance in the absence of confounding issues of quality learning, we first consider a scenario in which the quality scores are fixed. Here, we measure both the diversity and quality of articles chosen each day by the different methods. We then turn to quality learning based on user feedback to examine how the properties of the M-kDPP influence the discovery of a user's preferences.

5.1 FIXED QUALITY

Similarity To generate similarity features ϕ_i, we first compute standard normalized tf-idf vectors, where the idf scores are computed across all 26 weeks worth of articles. We then compute the cosine similarity between all pairs of articles. Due to the sparsity of the tf-idf vectors, these similarity scores tend to be quite low, leading to poor diversity if used directly as a kernel matrix. Instead, we let the similarity features be given by binary vectors where the jth coordinate of ϕ_i is 1 if article j is among the 150 nearest neighbors of article i in that week based on our cosine distance metric, and 0 otherwise.

Quality In the fixed scenario, we need a way to assign quality scores to articles. A natural approach is to score articles based on their proximity to the other articles; this way, an article that is close to many others (as measured by cosine similarity) is considered to be of high quality. In this data set, for example, we find that there is a large cluster of articles that talk about *politics* and articles that fall under this topic generally have much higher quality than articles that talk about, say, *food*. To model this, we compute quality scores as $q_i = \exp(\alpha d_i)$, where d_i is the sum of the cosine similarities between article i and all other articles in our collection and α is a hyperparameter that determines the dynamic range. We chose $\alpha = 5$ for our data set, although a range of values gave qualitatively similar results.

For each method, we sample sets of articles on a daily basis for each of the 26 weeks. To measure diversity within a time step, we compute the average cosine similarity between articles chosen on a given day. We then subtract the result from 1 so that larger values correspond to greater diversity. Diversity between time steps is obtained by measuring the average cosine similarity between each article at time t and

Table 1: Average Diversity and Quality of Selected Articles

Method	Marginal diversity	1-step diversity	2-step diversity	Quality
M-kDPP	0.899	0.849	0.843	0.654
k-DPP	0.896	0.786	0.779	0.668
k-DPP + heuristic (0.4)	0.904	0.849	0.804	0.651
k-DPP + heuristic (0.2)	0.946	0.891	0.889	0.587
Weighted Rand.	0.750	0.681	0.677	0.756
Uniform Rand.	0.975	0.949	0.947	0.457

the single most similar article at time $t+1$ (or $t+2$ for 2-step diversity), and again subtracting the result from 1. We also report the average quality score of the articles chosen across all 182 days. All measures are averaged over 100 random runs; statistical significance is computed by bootstrapping.

Table 1 displays the results for all methods. The M-kDPP shows a marked increase in between-step diversity, on average, compared to the kDPP and weighted random sampling. All of the differences are significant at 99% confidence. The average marginal diversities for the M-kDPP and kDPP are statistically significantly higher than for weighted random sampling, but are not statistically significantly different from each other. This is to be expected since, as we have seen in Sec. 3, the marginal distribution for the M-kDPP does not greatly differ from the kDPP process. On the other hand, the uniform sampling shows much higher diversity than the other methods, which can be attributed to the fact that it is a purely exploratory method that ignores the quality of the articles it chooses.

Table 1 also shows the average quality of the selected articles. The weighted random sampling chooses, on average, higher quality articles compared to the rest of the methods since it does not have to balance issues of diversity within the set. The kDPP on average chooses slightly higher quality articles than the M-kDPP, perhaps due to the additional between-step diversity sought by the M-kDPP; however, the difference is not statistically significant. It is evident from Table 1 that the M-kDPP achieves a balance between the diversity of the articles it chooses (both marginally and across time steps) and their quality.

As for the kDPP + heuristic baseline, our experiments show that by tuning the threshold carefully we can mimic the performance of the M-kDPP, but without the associated probabilistic interpretation and theoretical properties. When the threshold is too low, quality degrades significantly.

5.2 LEARNING PREFERENCES

We also study the performance of the M-kDPP when learning from user feedback, as outlined in Sec. 4. For simplicity, we use only a week's worth of news articles (1427 articles). To create feature vectors, we first generate topics by running LDA on the entire corpus (Blei et al., 2003). We then manually label the most prevalent 10 topics as *finance*, *health*, *politics*, *world news*, *baseball*, *football*, *arts*, *technology*, *entertainment*, and *justice*, and associate each article with its LDA-inferred mixture of these topics (a 10-dimensional feature vector f_i). We define a synthetic user by a sparse topic preference vector (0.7 for *finance*, 0.2 for *world news*, 0.1 for *politics*, and 0 for all other topics), and preselect as "preferred" the 200 articles whose feature vectors f_i maximize the dot product with the user preference vector.

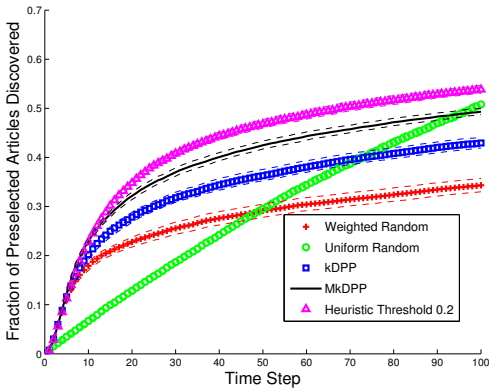


Figure 2: Performance of the methods at recovering the preselected preferred articles. Solid lines indicate the mean over 100 random runs, and dashed lines indicate the corresponding confidence intervals, computed by bootstrapping.

Similar to our previous experiment, we define the similarity features between articles to be binary vectors based on 50 nearest neighbors using the tf-idf cosine distances. The quality is defined as in Sec. 4, $q_i^{(t)} = \exp(\theta^{(t)\top} f_i)$, where f_i is the feature vector of article i (based on the mixture of topics) normalized to sum to 1. We set the learning rate $\eta = 2$; however, varying η did not change the qualitative behavior of each method, only the time scale at which these behaviors became noticeable. We also note that although we base the similarity on 50 nearest neighbors, the results were not sensitive to the size of this neighborhood.

The goal of this experiment is to illustrate how the different methods balance between exploring the space of all articles to discover the 200 preselected articles (*recall*) and exploiting a learned set of features to keep showing preferred articles (*precision*). On one end of the spectrum, uniform sampling simply explores the space of articles without taking advantage of the user feedback, leading to high recall and low precision. On the other end, the weighted random sampling fully exploits the learned preference in selecting articles, but does not have a mechanism to encourage exploration. We demonstrate that the M-kDPP balances these two extremes, taking advantage of the user feedback while also exploring diverse articles.

Results We use each method to select 10 articles per day over a period of 100 days, using the current quality scores $q^{(t)}$ on each day t. We measure recall by keeping track of the fraction of preselected preferred articles (out of the 200 total) that have been displayed so far. We also compute, out

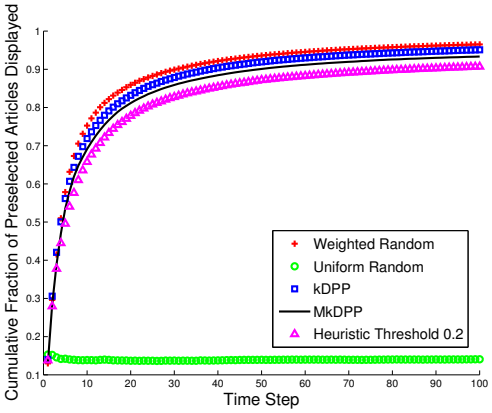


Figure 3: Cumulative fraction of preferred articles displayed to the user.

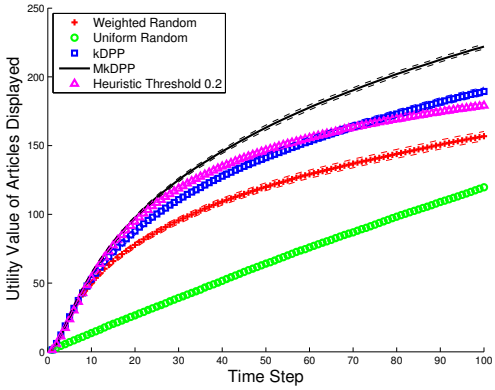


Figure 4: Performance measured by a marginally decreasing utility function.

of the 10 articles shown on a given day, the fraction that are preferred. This serves as a measure of precision. All measures are averaged over 100 random runs.

Figure 2 shows the recall performance of the methods we tested. Uniform sampling discovers the articles at a somewhat linear rate of about 5% per day; given a larger base set relative to the size of the preferred set, however, we would expect a slower rate of discovery. The methods that incorporate user feedback discover a larger set of preferred articles more rapidly by harnessing learned features of the user's interests. The M-kDPP dominates both the kDPP and weighted random sampling in this metric since it encourages exploration by introducing both marginal and between-step diversity of displayed articles. In contrast, the kDPP does not penalize repeating similar marginally diverse sets and the weighted random sampling does not have any explicit mechanism for exploration. It takes uniform random sampling nearly 100 time steps to discover the same number of unique preferred articles as the M-kDPP. For the sake of clarity, we omit the results of kDPP + heuristic with threshold 0.4 since they are not statistically significantly different from the M-kDPP. The supplementary material includes a randomly selected example of the articles displayed on days 99 and 100 for the various methods.

Figure 3 shows the cumulative fraction of displayed articles that were preferred, reflecting precision. (The supplement includes a sample non-cumulative version of Figure 3.) All methods besides uniform sampling quickly achieve high precision. Weighted random sampling displays the largest number of preferred articles per day, almost always having precision of at least 0.9. However, as we have observed, this large precision is at the cost of lower recall. In particular, weighted random sampling quickly homes in on features related to a small subset of preferred articles, thereby increasing the probability of them being repeatedly selected with no force to counteract this behavior. As expected, by only requiring marginal diversity, the kDPP achieves slightly higher precision than the M-kDPP on average (both typically above 0.8), but again at the cost of reduced exploration. Overall, the differences in precision between these methods are not large. In many applications, having 8 out of 10 results preferred may be more than sufficient.

Finally, to examine the balance between exploration and exploitation, we compute a metric based on the idea of marginally decreasing utility. Under this metric, at every time step, the user experiences a utility of 1 for each preferred article shown for the first time. If a previously displayed preferred article is once again chosen, the user gets a utility of $\frac{1}{l+1}$ where l is the number of times that article has appeared in the past. The underlying assumption is that a user benefits from seeing preferred articles, but in decreasing amounts as the same articles are repeatedly displayed. Figure 4 shows the performance of the methods under this utility metric; the M-kDPP scores highest.

6 CONCLUSION

We introduced the Markov DPP, a combinatorial process for modeling diverse sequences of subsets. By establishing the theoretical properties of this process, such as stationary DPP margins and a DPP union process, we showed how our construction yields sets that are diverse at each time step as well as from one time step to the next, making it appropriate for interactive tasks like news recommendation. Additionally, by explicitly connecting with DPPs, further properties of M-DPPs are straightforwardly derived, such as the marginal and conditional expected set cardinality.

We showed how to efficiently sample from a M-DPP, and found empirically that the model achieves an improved balance between diversity and quality compared to baseline methods. We also studied the effects of the M-DPP on learning, finding significant improvements in recall at minimal cost to precision for a news task with user feedback.

7 ACKNOWLEDGMENTS

This work was supported in part by AFOSR Grant FA9550-10-1-0501 and NSF award 0803256.

References

V. Anantharam, P. Varaiya, and J. Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-Part I: IID rewards. *Automatic Control, IEEE Transactions on*, 32(11):968–976, 1987.

R. A. Bernstein and M. Gobbel. Partitioning of space in communities of ants. *Journal of Animal Ecology*, 48(3): 931–942, 1979.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3: 993–1022, 2003.

A. Borodin and E. Rains. Eynard-Mehta theorem, Schur process, and their Pfaffian analogs. *Journal of Statistical Physics*, 121:291–317, 2005.

J. Ginibre. Statistical ensembles of complex, quaternion, and real matrices. *Journal of Mathematical Physics*, 6:440, 1965.

D. Graff and C. Cieri. English Gigaword, 2009.

J. B. Hough, M. Krishnapur, Y. Peres, and B. Virág. Determinantal processes and independence. *Probability Surveys*, 3:206–229, 2006.

A. Kulesza and B. Taskar. Structured determinantal point processes. In *Advances in Neural Information Processing Systems*, 2010.

A. Kulesza and B. Taskar. k-DPPs: Fixed-size determinantal point processes. In *Proc. International Conference on Machine Learning*, 2011a.

A. Kulesza and B. Taskar. Learning determinantal point processes. In *Proc. Conference on Uncertainty in Artificial Intelligence*, 2011b.

J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

O. Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975.

M. L. Mehta and M. Gaudin. On the density of eigenvalues of a random matrix. *Nuclear Physics*, 18(0):420 – 427, 1960.

T. Neeff, G. S. Biging, L. V. Dutra, C. C. Freitas, and J. R. Dos Santos. Markov point processes for modeling of spatial forest patterns in Amazonia derived from interferometric height. *Remote Sensing of Environment*, 97(4): 484–494, 2005.

Y. Yue and C. Guestrin. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems*, 2011.

Toward Large-Scale Agent Guidance in an Urban Taxi Service

Lucas AGUSSURJA
School of Information Systems
Singapore Management University
80 Stamford Road, S178902

Hoong Chuin LAU
School of Information Systems
Singapore Management University
80 Stamford Road, S178902

Abstract

Empty taxi cruising represents a wastage of resources in the context of urban taxi services. In this work, we seek to minimize such wastage. An analysis of a large trace of taxi operations reveals that the services' inefficiency is caused by drivers' greedy cruising behavior. We model the existing system as a continuous time Markov chain. To address the problem, we propose that each taxi be equipped with an intelligent agent that will guide the driver when cruising for passengers. Then, drawing from AI literature on multi-agent planning, we explore two possible ways to compute such guidance. The first formulation assumes fully cooperative drivers. This allows us, in principle, to compute system-wide optimal cruising policy. This is modeled as a Markov decision process. The second formulation assumes rational drivers, seeking to maximize their own profit. This is modeled as a stochastic congestion game, a specialization of stochastic games. Nash equilibrium policy is proposed as the solution to the game, where no driver has the incentive to singly deviate from it. Empirical result shows that both formulations improve the efficiency of the service significantly.

1 INTRODUCTION

Taxis are a major mode of transport in every urban city in the world. In Singapore, as of April 2009, there were about 24,000 taxis and 87,000 licensed drivers, providing around 850,000 trips daily. These are operated by a small number of companies. The largest company, ComfortDelgro, operates over 15,000 taxis, and captures the majority of the market share in terms of ridership. Like many congested cities in the world, commuters in Singapore often view taxis as a more efficient mode of transport compared to private cars. Data from the Singapore Land Transport Authority (the regulatory agency for land transportation) show that taxis on average chalk up higher mileage than private cars, with single-shift taxis traveling some 120,000 km a year. More than a third of this travel is empty cruising, as shown in the next section, see (Tan et al., 2009) as well, which represents a significant wastage of resources.

Taxi services have been studied extensively in the literature. Research has been conducted to investigate the services' demand-supply interaction and the resulting market equilibrium (Cairns and Liston-Heyes, 1996; Yang et al., 2002). These studies aim to predict the economic consequences of regulatory policies, such as entry restriction and fare control. At the operational level, quantitative models have been built to capture, for example, passengers' and drivers' bilateral searching behavior (Wong et al., 2005). While these models tend to be descriptive, our work is operative by nature. We seek to provide solutions to a specific problem, namely, we improve the inefficiency of existing cruising policy by providing alternative policies derived from multi-agent planning and decision making models. We believe the public transport arena offers a rich domain for application of multi-agent concepts and methodologies.

We start by analyzing a dataset obtained from a major taxi operator, which traces the movement of a large number of taxis in Singapore. Using the data for the month of July 2009, our analysis shows that the current system's inefficiency, measured in terms of cruising hours, is due to the inherently greedy behavior of drivers. This, in turn, is caused by the lack of visibility in regards to the distribution of cruising taxis on the network at different time periods. We then model the existing taxi service by a continuous-time Markov chain, whose parameters are derived from the dataset. The result is used as the baseline for empirical com-

parison with our approach.

We proceed to propose two models: a cooperative and a non-cooperative model. In our formulations, each driver is endowed with a guidance agent that provides suggestions on how to cruise in search of passengers. In the first formulation, we assume that each driver is fully cooperative, and therefore willing to follow a centralized policy (that could be suboptimal for him/her individually). This allows us, in principle, to compute the system-wide optimal policy, where the objective is to maximize overall occupied time. For this purpose, we model the problem as a Markov decision process. In the second formulation, we assume that drivers are rational who seek to maximize their respective occupied time. Here, there is an implicit competition among drivers, simply because an increase in the number of cruising taxis in a zone decreases each driver's chance of finding passengers within a given time. This leads directly to a game theoretic formulation. We model the problem as a stochastic congestion game (a specialization of stochastic games) and seek to find a Nash equilibrium policy. Given an equilibrium policy, no driver has the incentive to singly deviate from it. Both problems are solved for finite horizon using value iteration coupled with a sampling technique.

There has been a surge of interest, in recent years, among AI and complexity theory communities in computational problems related to game theory. The central problem in the area is the computation of Nash equilibrium in different game settings. Classical algorithms to solve the standard simultaneous games have relied on homotopy methods, which solve fixpoint (one instance of which is the Nash equilibrium) problems. The most well-known of such methods is the Lemke-Howson algorithm. See (Herings and Peeters, 2009) for a recent survey, and (Goldberg et al., 2011) for a discussion on the complexity of such methods. Non-homotopy attempts in the literature include enumeration of strategy supports (Porter et al., 2004) and mixed-integer programming (Sandholm et al., 2005).

In the setting of dynamic games, specifically in stochastic games (Shapley, 1953), the problem of computing Nash equilibrium has been cast in the context of multiagent reinforcement learning, first introduced by (Littman, 1994), who shows the convergence of value iteration in 2-player zero-sum stochastic games. (Hu and Wellman, 2003) attempts to generalize the result to n-player general-sum games by extending Q-learning algorithm. Their algorithm converges to Nash equilibrium in a restrictive case. Recent attempt by (Kearns et al., 2000) proposes an algorithm that converges to an equilibrium-like joint policy but not Nash equilibrium. Finding an algorithm that converges to Nash equilibrium in n-player general-sum stochastic games remains an important open problem.

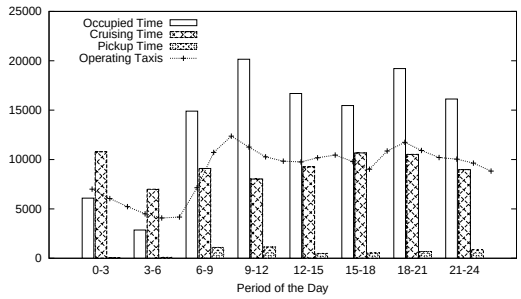
Despite the tremendous interest, many interesting real-world problems (such as the one presented here) are so large that even the best algorithms have no hope of computing an equilibrium directly. Furthermore, it has been shown that the problem is likely to be intractable even for the case of two-player simultaneous games (Chen and Deng, 2006), it being complete for the complexity class PPAD. The standard approach to overcome this problem is to construct a smaller game that is similar to the original game, and solve the smaller game. Then, the solution is mapped to a strategy profile in the original game (Ganzfried and Sandholm, 2011).

In this work, we tackle a moderate-size real-world problem, by solving for finite horizon approximate equilibrium. Our aim is to extend this solution approach to cover the full scale problem in the future. Our experimental results show that the optimal policy (derived from the cooperative model) manages to reduce the cruising time of the existing policy by approximately 30%. This result may be viewed as what can be achieved theoretically. In real-life implementation, however, the improvement will be closer to that of equilibrium policy, which is approximately 20%.

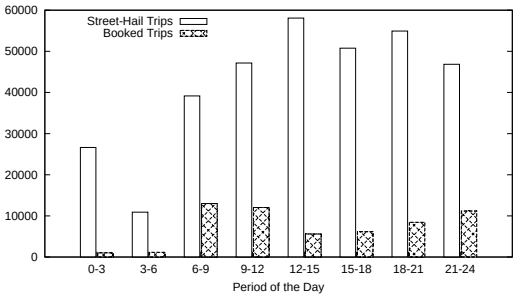
2 ANALYSIS OF CURRENT SYSTEM

Figure 1 and 2 summarize the taxi operation on weekdays (Mon-Thu) for the month of July 2009. In Figure 1a, we see the comparison of average daily time spent between delivering passengers, cruising, and picking up passengers (for booked trips) for different periods of the day. The average number of operating taxis is also shown in the figure. To book a taxi, passengers may send their request to the central operator and specify their current location. The central operator will then broadcast the request to cruising taxis around the specified location. Drivers who receive the request may bid for the job by specifying the time required to reach the pickup point. The job is given to the driver with the shortest pickup time. The proportion of street-hail vs. booked trips, for different periods of the day, is shown in Figure 1b. From Figure 1a, we observe that taxi cruising time is almost constant throughout the day, accounting for roughly half of the total operating hours. On the other hand, pickup time accounts for only a small percentage of operating hours. For this reason, in this work, we are focusing on reducing the cruising time for street-hail jobs.

Figure 2 describes the cruising behavior of drivers. In

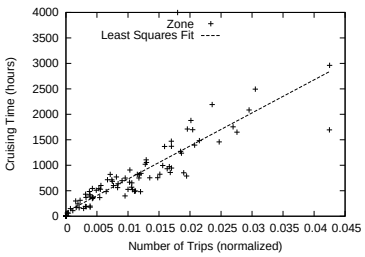


(a) The histogram shows the average total time spent daily, in hours, on: delivering passengers, cruising, and reaching pickup points (for booked trips) respectively, in different periods of the day. The curve shows the number of operating taxis.

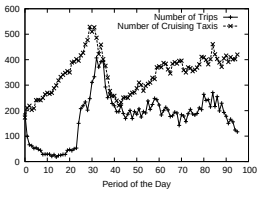


(b) The histogram shows the average daily number of street-hail and booked trips respectively in different periods.

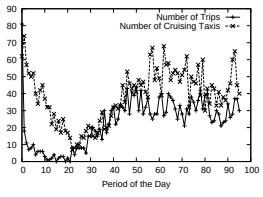
Figure 1: Summary of weekday data for July 2009



(a) The correlation between trip frequency and cruising time in a zone



(b) Number of trips vs. cruising taxis in a high trip-frequency zone with peaks



(c) Number of trips vs. cruising taxis in a low trip-frequency zone

Figure 2: Inefficiency of existing cruising policy

Figure 2a, we see that cruising time is high for zones with a high number of available trips. This indicates that drivers are spending time cruising in high trip-frequency zones. The same can be observed from Figure 2b and 2c. Figure 2b shows that taxis are entering high trip-frequency zones in anticipation for the surge of passengers. The number of cruising taxis will drop with the surge, but starts to build up again for the next surge, and the cycle continues. On the other hand, in low trip-frequency zones, except for early morning hours, the number of cruising taxis stays proportional to trip-frequency. We can conclude that drivers are employing a greedy cruising policy, spending a large amount of time cruising in high trip-frequency zone. This is one of the causes for the system's inefficiency, which is verified in the empirical study, when this policy is substituted for better cruising policies.

2.1 A MODEL FOR THE EXISTING SYSTEM

In this section, we model the aggregate behavior of a taxi service as a continuous time Markov chain. In our model, the taxi service operates on a road network which can be divided into logical cruising zones. We denote the network of zones by a directed graph $G = (N, E)$. At any point of time, a taxi is in one of the following states, which corresponds to the states in the Markov chain $\mathcal{S} = \{\mathcal{O}_{kl}, \mathcal{C}_k, \mathcal{W}_k | k, l \in N\}$: (1) the taxi is occupied and is delivering its passenger from zone k to l, denoted by the state $\mathcal{O}_{kl}$, or (2) the taxi is empty and cruising in a zone k, denoted by $\mathcal{C}_k$, or (3) the taxi is in zone k but not in operation, denoted by $\mathcal{W}_k$. Here, we assume that drivers have uniform cruising behavior that are independent of each other. As an example, Figure 3 shows a subset of Singapore's road network and the corresponding Markov chain, modeling the taxi service operating on the network.

In a continuous time Markov chain, the time spent in a state before moving to another state is a continuous random variable that is exponentially distributed. The rates of transition between the states constitute the generator matrix (or Q-matrix) of the Markov chain. In our model, the generator matrix consists of the following four components. The first component, $\{\lambda_{kl} | (k, l) \in E\}$, describes a driver's cruising behavior. Since a driver does not have visibility in regards to the state and location of other drivers, we can assume that they are approximately independent of each other. The second component, $\{\pi_{kl} | k, l \in N\}$, describes the likelihood of finding passengers in a zone, where π_{kl} is the rate of finding passengers in zone k with a destination point in zone l. Assuming independence for π_{kl}, the total rate of finding passengers in zone k is given by $\sum_l \pi_{kl}$ (a combination of Poisson

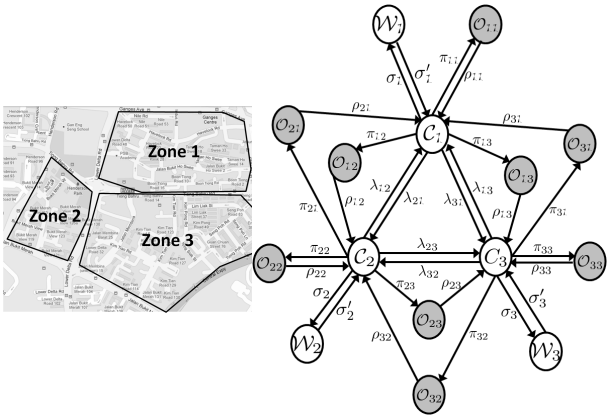


Figure 3: An example of a road network, divided into its cruising zones, and the corresponding Markov chain, modeling a taxi operating on the network. The shaded nodes indicate the states where the taxi is occupied. Here, we assume that taxis are uniform and independent of each other.

processes). The third component, $\{\rho_{kl}|k,l \in N\}$, describes the time needed to deliver passengers to their destination point. The nondeterminism of these variables is due to the variability of pickup and drop-off points within zones and the congestion on the road network. The forth component, $\{\sigma_k, \sigma'_k|k \in N\}$, describes the likelihood of a driver taking a break within a zone (σ_k) and the length of the break (σ'_k). Most taxis are operated by two or more drivers taking turns continuously. This accounts for the nondeterminism of the forth component.

We define the system efficiency as the steady state (stationary) probability of a taxi being in the occupied state. Let $\theta(s)$, for $s \in \mathcal{S}$, denote the steady state probability of being in the state s. This probability distribution can be obtained by solving the following system of equations:

$$\begin{aligned}
\forall k \in N, \quad & \sum_{l:(l,k)\in E} \lambda_{lk}\theta(\mathcal{C}_l) + \sum_{l:l\in N} \rho_{lk}\theta(\mathcal{O}_{lk}) + \sigma'_k\theta(\mathcal{W}_k) \\
& = \left(\sum_{l:(k,l)\in E} \lambda_{kl} + \sum_{l:l\in N} \pi_{kl} + \sigma_k\right)\theta(\mathcal{C}_k), \\
\forall k,l \in N, \quad & \pi_{kl}\theta(\mathcal{C}_k) = \rho_{kl}\theta(\mathcal{O}_{kl}), \\
\forall k \in N, \quad & \sigma_k\theta(\mathcal{C}_k) = \sigma'_k\theta(\mathcal{W}_k),
\end{aligned}$$

and normalizing the solution such that $\sum_{s\in\mathcal{S}} \theta(s) = 1$. The steady state probability of a taxi being in the occupied state is thus given by $\sum_{k,l\in N} \theta(\mathcal{O}_{kl})$.

We estimate the transition rates of the Markov chain using the dataset. The random variables cruising time (with rate λ_{kl}) and passenger find time (with rate π_k)

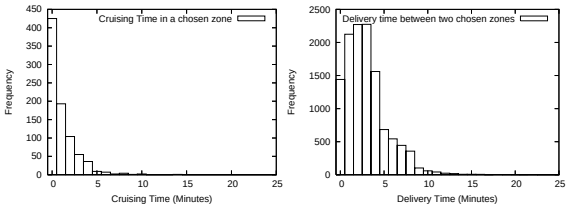


Figure 4: The frequency histograms (for one day) for cruising time in a randomly chosen zone, and delivery time between two randomly chosen zones, respectively.

can be estimated accurately with exponential distributions. The other variables, however, are closer to Erlang distributions than exponential distributions. Figure 4 shows, for example, two frequency histograms, one for each of these cases. In this work, we approximate both cases by exponential distributions using maximum likelihood estimate. These approximations are better when the zones are adequately close to each other, which is the case for Singapore.

One can derive the system efficiency once the parameters of the Markov chain are obtained. Since we also like to compute efficiency under different customer arrival rates and different numbers of operating taxis (for empirical study), we model the passenger-driver dynamics in a zone as an M/M/1 queue. Consider the subset of the Markov chain consisting only the states $\{\mathcal{C}_k|k \in N\}$ and the transitions rate between them $\{\lambda_{kl}|(k,l) \in E\}$. Each zone is modeled as a FIFO queue, where its arrival rate is the arrival rate of cruising taxis in the zone, and its service rate is the passengers arrival rate in the zone. Let μ_k denote the passengers arrival rate in zone k, then the waiting time in the queue (queueing time + service time) is exponentially distributed with the following rate, which we associate with π_k:

$$\pi_k = \mu_k - n\left(\sum_{l:(l,k)\in E} \phi(\mathcal{C}_l)\lambda_{lk}\right) \quad ,$$

where n is the number of taxis in operation and $\phi(\mathcal{C}_l)$ the steady state probability of being in the state $\mathcal{C}_l$. The steady state probabilities ϕ can be obtained, similar to θ, by solving the following system of equations:

$$\forall k \in N, \quad \sum_{l:(l,k)\in E} \lambda_{l,k}\phi(\mathcal{C}_l) = \phi(\mathcal{C}_k) \sum_{l:(k,l)\in E} \lambda_{kl} \quad ,$$

subject to $\sum_{k\in N} \phi(\mathcal{C}_k) = 1$. Now, given the drivers' cruising behaviors (in the form of smaller Markov chain parameterized by $\{\lambda_{kl}|(k,l) \in E\}$), travel time information $\{\rho_{kl}|k,l \in N\}$, and nonoperating profile $\{\sigma_k, \sigma'_k|k \in N\}$, we can derive the system efficiency as the function of passenger arrival rates $\{\mu_k|k \in N\}$ and

the total number of taxis n, by first constructing the full Markov chain, and then deriving its steady state probabilities.

3 A COOPERATIVE MODEL

The cooperative model is based on Markov decision process, which is a special case of stochastic games, where there is only one player. The player, in this case, is a central operator whose actions are all possible joint actions of the drivers, and whose objective is the overall occupied time of the service. The model is similar to the noncooperative case. The difference lies in the formulation of the utility function. In the noncooperative case, we have a set of utility functions to be optimized simultaneously, while in the cooperative case, we have a single aggregated utility function. We choose to present our models under the more general setting of stochastic games (next section). We will highlight the differences for the cooperative case.

4 A NONCOOPERATIVE MODEL

Stochastic games (Shapley, 1953; Littman, 1994) are a generalization of Markov decision processes to a multi-agent setting by allowing the state transitions to be influenced by their joint action. They are also a generalization of sequential games with perfect information, by introducing different states. The state (and thus the payoff matrix) changes as a result of both nature and the joint action of the agents. In congestion games, agents share a set of facilities (facilities can be viewed as resources, such as machines), and the utility an agent derives from using a facility reduces as the number of agents using the same facility increases. In the taxi context, the facilities correspond to the zones where taxi agents can cruise. Each zone has a fixed arrival rate of passengers, and therefore, an increase in the number of cruising taxis in the zone, decreases the likelihood that each would find passengers within a certain time period. Stochastic congestion games therefore are a generalization of congestion games, allowing the games to be played indefinitely. They are also a specialization of stochastic games, to the case where the underlying games are congestion games. A state in a stochastic congestion game defines an assignment of agents to facilities, and transitions to a new state are dependent on the old state and the agents' joint action. In this work, we consider only games with finite horizon.

4.1 STOCHASTIC CONGESTION GAMES

Formally, a finite stochastic congestion game (SCG) Γ is a tuple $(I, J, P, \mathcal{R})$, where

- $I = \{1, \ldots, n\}$ is the set of agents.
- $J = \{1, \ldots, m\}$ is the set of facilities. An action of an agent i, denoted by a^i where $a^i \in J$, is to choose a facility to which it would like to be assigned. We denote an agents' joint action by $a = (a^1, \ldots, a^n)$ and the set of possible joint actions by A, that is, $A = J^n$. Next, we denote by S, the set of possible states, where a state $s \in S$ is an assignment of agents to facilities, and we define s_j as the number of agents assigned to facility j in the state s.
- $P : S \times A \to \triangle(S)$ is the state transition function. For convenience, we will also write $P(s, a, s')$ for the probability that the next state is s' given that the current state is s and the players' joint action is $a = (a^1, \ldots, a^n) \in A$,
- $\mathcal{R} = \{r_j\}_{j \in J}$ is the reward function, where each $r_j : \{0, \ldots, n\} \to \mathbb{R}$ is a nonincreasing function that maps the number of agents assigned to facility j to the reward that each of the agents in j receives. We define $s(i, j)$ such that $s(i, j) = 1$ if agent i is assigned to facility j in the state s, and $s(i, j) = 0$ otherwise. The reward received by agent i in a state s, denoted by $R^i(s)$, is thus given by $\sum_{j \in J} s(i, j) r_j(s_j)$.

The game proceeds in steps, starting from some initial state s_T (the subscript indicates the number of remaining steps). In each step, the agents first observe the current state s_t, the number of remaining steps t, and simultaneously choose actions according to their respective policy. An agent i's policy is the function $\pi^i : S \times \{1, \ldots, T\} \to \triangle(J)$, where $\pi^i(s_t, t)$ computes agent i's mixed strategy, i.e., a probability distribution over the set of facilities, given the current state s_t and the number of remaining steps t. Each agent i's action in this step, denoted by a^i_t, is then drawn from the distribution given by $\pi^i(s_t, t)$, forming the joint action $a_t = (a^i_t, \ldots, a^n_t)$. Nature then selects the next state s_{t-1} according to the probabilities given by $P(s_t, a_t)$. In the new state s_{t-1}, each agent i receives $R^i(s_{t-1})$ as its reward, and the game proceeds to the next step.

Given a possible outcome of the game, agent i's utility, denoted by U^i, is defined such that

$$U^i_T(s_T, a_T, s_{T-1}, \ldots, s_1, a_1, s_0) = \sum_{t=0}^{T-1} R^i(s_t).$$

A Nash equilibrium of the game is a joint policy π, such that, for each agent i, π^i maximizes the expected utility of agent i given that the other players follow their respective policy specified by π. The expectation is taken over all possible outcomes of the game. For the cooperative case, there is only one utility formed

by summing up individual agent's utility. Here, we seek to find a joint policy that maximizes the expected value of this utility.

4.2 MODELING TAXI SYSTEM AS AN SCG

The drivers correspond to the set of agents I. The set of facilities J corresponds to the states of the Markov chain, $\mathcal{S} = \{\mathcal{C}_k, \mathcal{O}_{kl}, \mathcal{W}_k | k, l \in N\}$, defined previously. We will refer to them as facilities here. A state of the game is an assignment of agents to facilities. When an agent is assigned to facility $\mathcal{C}_k$, it means that the corresponding taxi is cruising in zone k. Similarly, when it is assigned to facility $\mathcal{O}_{kl}$ and $\mathcal{W}_k$, it means that the corresponding taxi is occupied and not in operation, respectively. One step in the SCG represents the period of one minute.

Next, we define available actions and the state transition function. The set of available actions for an agent depends on the state of the agent. When in facility $\mathcal{C}_k$, the agent may take one of the following actions: $(a1)$ continue cruising in the current zone, or $(a2)$ make an attempt to move to an adjacent zone. In this state, the agent has the chance of getting a passenger and be moved to one of the facilities $\mathcal{O}_{kl}$ in the next step. Given that an agent is in facility $\mathcal{C}_k$ and takes the action $a1$, the following may happen in a step: (1) The agent manages to find a passenger with l as the destination zone. In the next step, the agent will move to facility $\mathcal{O}_{kl}$. The probability of this event happening is given by:

$$\frac{\mu_k}{n(\mathcal{C}_k)}\gamma_{kl},$$

where γ_{kl} is the probability that a passenger's destination zone is l given that its starting zone is k, and $n(\mathcal{C}_k)$ is the number of agents in facility $\mathcal{C}_k$ in the current state. γ_{kl} can be estimated by the following:

$$\gamma_{kl} \approx \frac{\pi_{kl}}{\sum_{l \in N} \pi_{kl}}.$$

The expression $\mu_k / n(\mathcal{C}_k)$ also defines the reward of the agent in this state. (2) The agent doesn't find any passenger and stays in the same facility. The probability of this event is given by: $(1 - \mu_k / n(\mathcal{C}_k))e^{-\sigma_k}$, or (3) the agent doesn't find any passenger and moves to facility $\mathcal{W}_k$ with probability $(1 - \mu_k / n(\mathcal{C}_k))(1 - e^{-\sigma_k})$. On the other hand, if the agent chooses $a2$, one of the following may occur: (1) a passenger, with destination l, is found before the agent manages to move to a new zone. The agent moves to facility $\mathcal{O}_{kl}$ in the next step. The probability of this happening is given by $\mu_k \gamma_{kl} / n(\mathcal{C}_k)$. (2) Otherwise, no passenger is found, and the agent moves to a new zone.

Algorithm FINITENASHPOLICIES(Γ, T):
Initialization phase:
  For all $s \in S$, joint action $a \in A$:
    $\pi(s, 0)[a] \leftarrow \frac{1}{|A|}$;
    For all $i \in I$:
      $V_{s,0}(a, i) \leftarrow 0$;
Iteration phase:
  For $t = 1$ to T, all $s \in S$:
    For all pure strategy profile $a \in A$, $i \in I$:
      $V_{s,t}(a, i) \leftarrow \sum_{s'} P(s, a, s') \times$
      $\left\{R^i(s') + \sum_{a'} \pi(s', t-1)[a'] V_{s', t-1}(a', i)\right\}$;
    $\pi(s, t) \leftarrow$ FINDNASH($V_{s,t}$);
  Return π;

Figure 5: Value Iteration for Finite-horizon SCGs

When in facility $\mathcal{O}_{kl}$, the agent has only one available action. It will try to deliver its passengers to their destination point and move to facility C_l. The probability of this happening in the next step is given by $1 - e^{-\rho_{kl}}$, while the probability of staying in the same facility is $e^{-\rho_{kl}}$. The reward for the agent in this state is one. Similarly, when in facility $\mathcal{W}_k$, the agent may move to $\mathcal{C}_k$ with probability $1 - e^{-\sigma'_k}$, or otherwise stays in the same zone. The reward for being in this facility is zero. Note that in our model, the move to facility $\mathcal{W}_k$ is involuntary on the part of the agents. An agent will never choose to stop its operation because of the way the reward function is structured. Rather than being an available action, the move is treated as a requirement instead, for example, when the driver has to change shift or rest.

4.3 COMPUTING EQUILIBRIUM POLICY

In this section, we describe an algorithm to computing equilibrium policy in SCGs based on value iteration algorithm, shown in Figure 5. This algorithm takes an SCG Γ, and a time horizon T as input. It outputs a vector of policies $(\pi^i(s, t))_{i \in I}$, where each π^i maps any state s, and the number of remaining steps $t \leq T$ to a mixed strategy for player i, which is a probability distribution over the set of facilities. The algorithm outputs a vector of policies that is a Nash equilibrium for the T-step SCG Γ from any start state. This algorithm is a generalization of the classical finite-horizon value iteration for Markov decision processes (Kaelbling et al., 1998). Instead of backup values, the algorithm maintains backup matrices, denoted by $V_{s,t}$. Each $V_{s,t}(a, i)$ is a function that takes a joint action a, a player i, and returns the expected utility of player i in the t-step SCG Γ, if the game starts at the state s and the first joint action of the agents is a. We can view, therefore, each $V_{s,t}$ as defining a game in the usual sense, and associated with each $V_{s,t}$ is a Nash

equilibrium $\pi(s,t)$, where $\pi^i(s,t)$ denotes the mixed strategy of player i. We denote by $\pi(s,t)[a]$ the probability assigned by $\pi(s,t)$ to the joint strategy a. Both $V_{s,t}$ and $\pi(s,t)$ are constructed iteratively. The algorithm uses the function FINDNASH that returns a Nash equilibrium given the game $V_{s,t}$. For the cooperative case, the function FINDNASH is replaced by FINDOPT which computes social optimum policy instead of Nash equilibrium.

Computing Nash equilibrium in stochastic games is a very difficult problem. See, for example, (Bowling, 2000; Hu and Wellman, 2003; Ganzfried and Sandholm, 2009) for recent works in this area. And the value iteration algorithm (for finite horizon stochastic games) is computationally very expensive for large scale problem. In this work, we employ the sparse sampling technique proposed by (Kearns et al., 2000).

5 EMPIRICAL RESULT

We run a simulation (on a macro level) to evaluate the performance of our proposed solutions. We choose, as the case study, 15 connected zones that represent the congested central business district of Singapore and its surrounding areas. We consider only passengers with origins and destinations within these zones, and restrict the cruising area of drivers to these zones as well. The number of operating taxis is 500. From the dataset we estimate the the passenger arrival rates $\{\mu_k | k \in N\}$, delivery time rate $\{\rho_{kl} | k, l \in N\}$, and taxis nonoperating profile $\{\sigma_k, \sigma'_k | k \in N\}$ for these zones in different periods. Using this setting we compare existing cruising policy (also derived from the dataset), the optimal policy (computed from the cooperative model), and the equilibrium policy (computed from the noncooperative model). The measure of comparison is cruising time. For both cooperative and noncooperative model, joint actions are computed every minute with 2-step look ahead ($T = 2$).

We simulate one day of operation on a weekday. The driver-passenger dynamics in each zone is implemented as a FIFO queue. As a passenger appears in a zone, it joins the queue associated with the zone. If there are some taxis cruising in the zone, the passenger at the head of the queue is removed and assigned to a randomly chosen taxi. The average time spent in the queue models the average passengers waiting time.

The experimental results are shown in Figures 6, 7 and 8. Figure 6 compares the average cruising time (over multiple runs) of the three policies. On average, the optimal policy and the equilibrium policy reduce the cruising time of existing policy by approximately 30% and 20% respectively. Most of the savings are obtained during the peak periods. This matches our intuition,

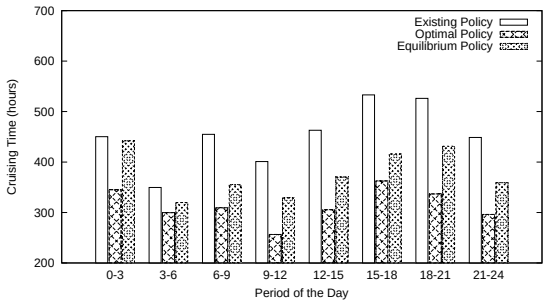


Figure 6: Comparison of cruising time between three policies. The optimal and equilibrium policies reduce the cruising time by approximately 30% and 20% respectively.

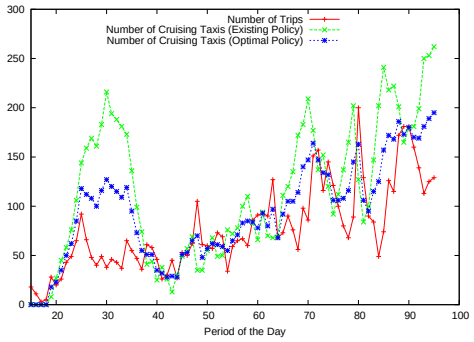


Figure 7: Number of trips vs. number of cruising taxi in a high trip-frequency zone. In general, the new policies send lower number of cruising taxis to high trip-frequency zones compared to existing policy.

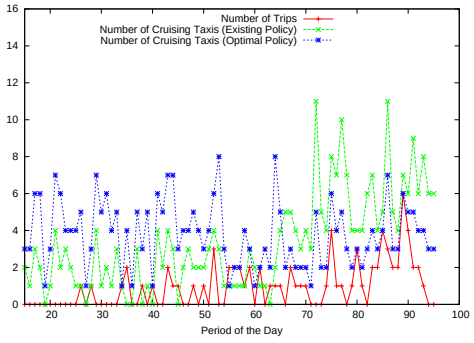


Figure 8: Number of trips vs. number of cruising taxi in a low trip-frequency zone. In general, the new policies send higher number of cruising taxis to low trip-frequency zones compared to existing policy.

that the new policies are able to better distribute cruising taxis among the zones, while the greedy policy sends too many cruising taxis to high trip-frequency zones. This is confirmed when we look at the number of trips vs. cruising taxis in both high and low trip-frequency zone (see Figure 7 and 8). The new policies are able to maintain balance between fulfilling trips requirement and managing cruising time.

6 Conclusion

We presented in this paper an interesting and useful application of Markov chains to manage an urban taxi service. The basic premise is the need to minimize cruising time (and therefore maximize utilization) of the taxis. There are several assumptions we made for this to work. We assume that each taxi has a device that guides the taxi driver. Using MDPs the system then generates a policy that optimizes cruising among all taxis. We showed, with the use of real data from a taxi operator in Singapore for the study, a cooperative model where taxis follow instructions and a noncooperative one where drivers compete with one another. Our study also assumes rational drivers that try to maximize the time their cars are occupied. For future works, we would need to incorporate real-world behavior of the passengers as well as relax the independence assumption of passengers and their destinations.

References

[1] Michael Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *Proceedings of the 17st International Conference on Machine Learning*, pages 89–94, 2000.

[2] Robert D. Cairns and Catherine Liston-Heyes. Competition and regulation in the taxi industry. *Journal of Public Economics*, 59(1):1–15, 1996.

[3] Xi Chen and Xiaotie Deng. Settling the complexity of two-player nash equilibrium. In *Proceedings of the 47th Symposium on Foundations of Computer Science*, pages 261–272, 2006.

[4] Sam Ganzfried and Tuomas Sandholm. Computing Nash equilibria in multiplayer stochastic games of imperfect information. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 140–146, 2009.

[5] Sam Ganzfried and Tuomas Sandholm. Game theory-based opponent modeling in large imperfect-information games. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 533–540, 2011.

[6] Paul W. Goldberg, Christos H. Papadimitriou, and Rahul Savani. The complexity of the homotopy method, equilibrium selection, and Lemke-Howson solutions. In *Proceedings of the 52nd Symposium on Foundations of Computer Science*, pages 67–76, 2011.

[7] P. Jean-Jacques Herings and Ronald Peeters. Homotopy methods to compute equilibria in game theory. *Economic Theory*, 42(1):119–156, 2009.

[8] Junling Hu and Michael P. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[9] Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[10] Michael Kearns, Yishay Mansour, and Satinder Singh. Fast planning in stochastic games. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 309–316, 2000.

[11] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

[12] Ryan Porter, Eugene Nudelman, and Yoav Shoham. Simple search methods for finding a Nash equilibrium. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence*, pages 664–669, 2004.

[13] Tuomas Sandholm, Andrew Gilpin, and Vincent Conitzer. Mixed-integer programming methods for finding nash equilibria. In *Proceedings of the 20th AAAI Conference on Artificial Intelligence*, pages 495–501, 2005.

[14] Lloyd S. Shapley. Stochastic games. *Proceedings of National Academy of Science*, 39:1095–1100, 1953.

[15] S. T. Tan, L. F. Lin, A. L. Chan, C. M. Tan, and D. H. Tan. Chapter 3 : Land transportation. In *Economics in Public Policy - The Singapore Story*. Marshall Cavendish Educ., 2009.

[16] K. I. Wong, S. C. Wong, M. G. H. Bell, and Hai Yang. Modeling the bilateral micro-searching behavior for urban taxi services using the absorbing Markov chain approach. *Advanced Transportation*, 39:81–104, 2005.

[17] Hai Yang, S. C. Wong, and K. I. Wong. Demand-supply equilibrium of taxi services in a network under competition and regulation. *Transportation Research Part B: Methodological*, 36:799–819, 2002.

Uncertain Congestion Games with Assorted Human Agent Populations

Asrar Ahmed, Pradeep Varakantham, Shih-Fen Cheng
School of Information Systems, Singapore Management University, Singapore 178902
{masrara,pradeepv,sfcheng}@smu.edu.sg

Abstract

Congestion games model a wide variety of real-world resource congestion problems, such as selfish network routing, traffic route guidance in congested areas, taxi fleet optimization and crowd movement in busy areas. However, existing research in congestion games assumes: (a) deterministic movement of agents between resources; and (b) perfect rationality (i.e. maximizing their own expected value) of all agents. Such assumptions are not reasonable in dynamic domains where decision support has to be provided to humans. For instance, in optimizing the performance of a taxi fleet serving a city, movement of taxis can be involuntary or non-deterministic (decided by the specific customer who hires the taxi) and more importantly, taxi drivers may not follow advice provided by the decision support system (due to bounded rationality of humans).

To that end, we contribute: (a) a general framework for representing congestion games under uncertainty for populations with assorted notions of rationality. (b) a scalable approach for solving the decision problem for perfectly rational agents which are in the mix with boundedly rational agents; and (c) a detailed evaluation on a synthetic and real-world data set to illustrate the usefulness of our new approach with respect to key social welfare metrics in the context of an assorted human-agent population.

An interesting result from our experiments on a real-world taxi fleet optimization problem is that it is better (in terms of revenue and operational efficiency) for taxi drivers to follow perfectly rational strategies irrespective of the percentage of drivers not following the advice.

1 Introduction

Congestion games [5] and its variants have numerous applications in domains ranging from urban transportation [14] (e.g. movement of vehicles between different regions of an area) to capturing company dynamics in a congested industry [12] (e.g., strategizing on marketing investments by different companies selling the same product). There are many interesting challenges that exist in these problem domains: (a) Representing and accounting for implicit interactions between agents. For example, vehicles trying to get on the same road are implicitly competing (even though there may not be an actual intention to compete); (b) large-scale nature of problems; (c) involuntary movements of agents (in some decision epochs, agents might need to *follow* given instructions and cannot freely choose their own actions); and (d) accounting for humans in the loop, who may not follow perfectly rational policies.

While existing research in congestion games is extensive, we are unaware of research that addresses the issues mentioned in points (b), (c) and (d) mentioned above. To that end, we make three key contributions in this paper: (1) By extending on the cognitive hierarchy model [1] (introduced in behavioral game theory), we introduce a general model to represent the decision problem for perfectly rational agents in an agent population of assorted rationalities; (2) A scalable approach called Soft-max based Flow update for Assorted Populations (SoFA) for solving the decision problem of perfectly rational agents in an assorted population; and (3) Finally, we provide a detailed evaluation on both real-world and synthetic data sets for the taxi fleet optimization problem.

This paper is motivated by a real-world problem of improving the performance of a taxi fleet. The decision making problem faced by a typical taxi driver is interesting both practically and theoretically, since a driver needs to make both voluntary (driver's own decision) and involuntary (when customers board taxis) movements. In such a problem, customers are considered

as the resources, due to whom an implicit competition exists between taxis. Since demands are both zone-dependent and time-dependent, the problem becomes even more challenging. The goal here is to provide decision support to taxi drivers such that the operational efficiency of the fleet and the revenues obtained are improved. Similar problems exist in analyzing industry dynamics (where different companies strategize to maintain their competitive advantage) and labor mobility (where individuals reason about their movement to different geographical regions). Although the concept of *user equilibrium* [10] is well-adopted in modeling either static or dynamic traffic route selection, it cannot be applied in this class of problems due to the presence of involuntary movements.

Experimentally, we are able to show that SoFA converges on all our problems (both real-world and synthetic ones) with assorted human-agent populations (agents with different levels of rationality). It is an important result since we are able to show that if our algorithm converges, the solution for perfectly rational agents is an equilibrium strategy (i.e., no incentive w.r.t expected value). Another nice empirical result of our approach in the taxi fleet optimization problem is that perfectly rational drivers (i.e. ones who adopt policies suggested by SoFA) always outperform agents who do not follow the suggested policies.

2 Motivation: Taxi Fleet Optimization

Our research is motivated by the problem of optimizing a fleet of self-interested taxis[1]. For a fleet of taxi population $\mathcal{P}$, serving a city divided into a set of zones M, the goal is to provide decision support for taxi drivers to independently decide a sequence of zones to roam in.

The demand fulfillment is modeled at zone-level, assuming demands are common knowledge among all taxis. If the number of taxis in a particular zone during a time period is fewer than the number of customers, all taxis will be hired (the determination of their destinations is described in the following paragraph). Otherwise, only a fraction of the roaming taxis (up to the number of customers) will be hired.

The movements of taxis between zones depend on whether they are hired or not. If a taxi is hired, the movement is involuntary (decided by the customer onboard) and is governed by the probability distribution computed from the outgoing flows of customers to different zones. Therefore, if a taxi is hired by a customer in zone i, the probability of moving to zone j is computed based on the fraction of customers moving from zone i to j over all flows out of zone i. Furthermore, a hired taxi in such a case will receive a revenue of $Re^t(i,j)$ and incur a cost of $Co^t(i,j)$. On the other hand, if the taxi is not hired, the movement will be voluntary and deterministic, and the taxi will receive no revenue but incur the same cost of $Co^t(i,j)$.

The goal is to provide decision support (in terms of policies) for drivers who request for decision support on moving through the island, such that there is no incentive for them to deviate from their policies.

[1]In our definition, we assume that each taxi is driven by an independent driver (thus we use *taxi* and *taxi driver* interchangeably), who pays for all costs (e.g., monthly rent, fuel, maintenance) and keeps all earned revenue. We also assume that there will be no communication/collaboration among taxi drivers.

3 Model: DAAP

We now introduce a general framework for modeling problems such as taxi fleet optimization called the Distributed decision model for Assorted Agent Populations (DAAP). DAAP is an extension of the DDAP (Distributed Decision model for Agent Populations) model introduced by Varakantham *et al.* [7]. DAAP can be viewed as a framework for representing generalized congestion games with movement uncertainty. In particular, we generalize the notion of resources in congestion games to states and the movement uncertainty to transition functions (akin to the one in Markov Decision Problems, MDPs).

DAAP represents a subset of problems represented by the generic stochastic game model [2]. In DAAP, the transition and reward functions for an agent are dependent only on the aggregate distributions of other agent states, whereas in more general models like stochastic games the transition and reward function for an agent are dependent on specific state and action of every other agent. It should be noted that DAAP represents problems with selfish agents and hence is different to cooperative models such as DEC-POMDP.

DAAP is the tuple:

$$\left\langle \Gamma, \{\mathcal{P}_\tau\}_{\tau\in\Gamma}, \mathcal{S}, \mathcal{A}, \{\phi_\tau\}_{\tau\in\Gamma}, \{\mathcal{R}_\tau\}_{\tau\in\Gamma}, d^0, \{\mathcal{V}_\tau\}_{\tau\in\Gamma} \right\rangle,$$

where Γ represents the different agent types and $\mathcal{P}_\tau$ is the set of agents of type τ. Two agents belong to different types if they have (a) different transition (ϕ) or reward model ($\mathcal{R}$); or (b) different notion of rationality ($\mathcal{V}$). $\mathcal{S}$ corresponds to the set of states encountered by each agent. $\mathcal{A}$ is the set of actions executed by each agent. We define the set of state distributions, $D = \{d | d = \langle d_1, d_2, \cdots, d_{|\mathcal{S}|} \rangle, \sum_{s\in\mathcal{S}} d_s = |\mathcal{P}|\}$, where $\mathcal{P}$ is the set of all agents and d_s represents the number of agents in state s. d^0 represents the starting distribution of agent states. ϕ_τ models the involuntary movements of every agent of type τ and more specifically, $\phi^t_{\tau,d}(s,a,s')$ represents the probability that an agent of type τ in state $s(\in \mathcal{S})$ after taking action $a(\in \mathcal{A})$ would transition to state s', when the distribution is d

and time is t. $\mathcal{R}^t_{\tau,d}(s,a,s')$ is the reward obtained by an agent of type τ when in state s, taking action a and moving to state s' when the distribution of agents is d at time t.

$\mathcal{V}_\tau$ characterizes the rationality concept of interest to agents of type τ and represents the value function definition for agents of type τ. This is provided to capture the notion of bounded rationality (particularly for humans in the population). In this paper, we refer to the two extreme cases for bounded rationality, which are:
(a) Perfect rationality: agents compute exact best response to opponent policies;
(b) Local rationality: agents assume no other agent is present ($d^t = \mathbf{0}, \forall t$) in the system.
Further details on various rationality concepts are provided in Section 4.

The objective in solving a DAAP is to compute a policy,π_i for each agent i of type τ, such that there is no incentive for any agent to deviate from its policy, in terms of the value (i.e. $\mathcal{V}_\tau(\pi_i, \{\pi_k\}_{k\in\mathcal{P}_\tau, k\neq i})$).

3.1 Taxi Fleet Optimization as DAAP

The number of types in this problem is primarily due to varying degrees of human rationality. We will describe this in detail in Section 7.1. In representing the taxi fleet optimization as a DAAP, the key is using the transition function to represent the involuntary movement of taxis[2]. $\mathcal{P}$ is the set of taxis in the fleet, $\mathcal{S}$ is all the zones in which a taxi could be present. $\mathcal{A}$ is the set of zones to which a taxi driver wishes to move. The transition function, ϕ is computed based on the customer flow, fl between various zones. D^0 is the distribution of taxis at the starting time. Equation 1 provides the expression for computing the transition probability between states.

Intuitively, if there are fewer taxis than customers in a zone, then all taxis are hired and their transition probability to a specific zone is dependent on flows to different zones from the current zone (represented by condition **C1**). If there are more taxis in a zone than customers, then transition is dependent on whether the action (intended zone) is same as the destination zone (captured by conditions **C2** and **C3**).
C1: if $\sum_{\hat{s}} fl^t(s,\hat{s}) \geq d_s$
C2: if $a \neq s'$, $\sum_{\hat{s}} fl^t(s,\hat{s}) < d_s$
C3: if $a = s'$, $\sum_{\hat{s}} fl^t(s,\hat{s}) < d_s$

$$\phi^t_d(s,a,s') = \left\{ \begin{array}{ll} \frac{fl^t(s,s')}{\sum_{\hat{s}} fl^t(s,\hat{s})} & \mathbf{C1} \\ \frac{fl^t(s,s')}{d_s} & \mathbf{C2} \\ 1 - \frac{\sum_{\hat{s}\neq s'} fl^t(s,\hat{s})}{d_s} & \mathbf{C3} \end{array} \right\} \quad (1)$$

[2]Since the movement of taxis and their revenues (standardized meters) are identical, we do not index transition and reward model of DAAP with type.

Similar to the transition, $\mathcal{R}$ is the reward obtained based on the three conditions.

In solving the taxi optimization DAAP, our goal is to maximize expected revenue for the individual taxi drivers while reducing starvation. Since the transition function depends on the number of taxis in the zone, maximizing expected revenue implies minimizing starvation as well. In this problem domain, both the welfare metrics (revenue and starvation) are optimized at once, however, in other domains there could be multiple objectives that are not in alignment and multi-objective reasoning might be required.

4 Characterizing Rationality

A major contribution of the paper is the introduction of multiple levels of rationality to the uncertain congestion game that mimics bounded rationality of human decision makers. In the context of DAAP, this implies the computed policy will be followed differently: perfectly rational agents adopt the policy completely, while other agents only adopt the policy partially (or even not at all). To concretely represent bounded rationality, we extend on the cognitive hierarchy model developed by Camerer *et al.* [1]. Therefore, in DAAP, $\mathcal{V}_\tau$ will represent the rationality notion of an agent in terms of the cognitive hierarchy model.

Cognitive hierarchy model assumes there is a bound on the levels of reasoning agents can do and that agents differ in their levels of reasoning. *Level-0* agents are assumed to be non-strategic and either play randomly or use a heuristic. *Level-1* agents compute best response strategy by assuming all other agents in the population are *level-0*. In general, *level-L* agents compute best response strategy by assuming a distribution f across levels $\{0, 1, \ldots, L-1\}$. Camerer *et al* present a single-parameter model, called *Poisson cognitive hierarchy* model, where f is assumed to follow Poisson distribution.

We extend the cognitive hierarchy model in three ways. Firstly, the opponent distribution f is obtained from iterative Bayesian inference (details provided in Section 7.1). Secondly, the best response is computed using the *quantal best response* strategy. The quantal best response for player i to strategy s_{-i} is given by:

$$s_i(a_i) = \frac{e^{\lambda \cdot u_i(a_i, s_{-i})}}{\sum_{a'_i} e^{\lambda \cdot u_i(a'_i, s_{-i})}}.$$

For $\lambda = 0$, each action is assigned equal probability (which can be thought as completely irrational); as λ increases, quantal best response approaches standard best response (which is completely rational). Finally, we introduce the consideration of different decision horizons (referred to as T). With the above extensions, an agent's behavior can be quantified by a

three-tuple CH_QR(λ, L, T). To illustrate the flexibility of our model, we provide two examples below::

Example 1 *CH_QR*$(1, 0, 1)$ *corresponds to an algorithm that generates a quantal best response strategy with* $\lambda = 1$ *that maximizes one-step payoff without considering other agents.* *CH_QR*(∞, ∞, H) *corresponds to an algorithm that generates the exact best response considering full horizon, assuming all levels of reasoning from other agents.*

This parametric model enables us to evaluate the SoFA technique on various sets of assorted populations and demonstrate the utility of having perfectly rational agents within a human population. The Poisson cognitive hierarchy model has been shown to provide a good fit for empirical data involving human subjects. We demonstrate similar findings with the extended model.

5 Soft-max based Flow update for Assorted populations (SoFA)

In this section, we describe our algorithm for solving DAAPs. We provide an algorithm (Algorithm 1) that is general and is applicable to a problem where there are (a) multiple types of agents with each type having a different transition and reward model and (b) multiple notions of rationality. However, for expository purposes (and to focus on the theme of the paper), we consider the case where all agents have the same model (transition and reward functions) and the difference is only in their rationality concept.

Algorithm 1 SoFA($daap, \Gamma, \mathcal{P}$)

for all $\tau \in \Gamma$ **do**
  $\pi_\tau \leftarrow$ GetInitialPolicy(τ)
end for
$\tilde{\pi} \leftarrow \phi$
$iter \leftarrow 0$
while $\tilde{\pi} \neq \pi$ **do**
  $\tilde{\pi} \leftarrow \pi$
  for all $\tau \in \Gamma$ **do**
    $\langle d^1..d^H \rangle \leftarrow$ AgentDist(π_{-i}, d^0, ϕ), $i \in \mathcal{P}_\tau$
    $\{\tilde{x}^t_{s,a}\} \leftarrow$ BestRspnse($\tau, daap, \langle d^1 \cdots d^H \rangle$)
    $x^t_\tau(s,a) \leftarrow \frac{(iter \cdot x^t_\tau(s,a) + \tilde{x}^t_{s,a})}{iter+1}, \forall s, a, t$
    $\pi^t_\tau(s,a) \leftarrow \frac{x^t_\tau(s,a)}{\sum_a x^t_\tau(s,a)}, \forall s, a, t$
    $iter \leftarrow iter + 1$
  end for
end while

Our goal is to compute the policy for perfectly rational agents (policy computed using Algorithm 1) in a population that contains other types of agents. It should be noted that a round-robin best response algorithm does not converge even if we have just one type of perfectly rational agents. Therefore, we introduce our new algorithm, SoFA that is based on the well known Fictitious Play algorithm and has interesting theoretical properties.

Intuitively, the key idea is that at each iteration, an agent computes best response against aggregate policies (over iterations of policy computation) of other agents. Since, in DAAP, the interactions between agents are due to the implicit competition for resources, an agent only has to determine best response to distribution of agent states (and not individual agent states). This observation improves the scalability considerably due to two reasons:
(1) Agents have to plan for state distributions and not against individual states and actions of other agents;
(2) We can assume same policy for all agents of the same type (either due to having same model or same notion of rationality). This allows us to reduce best response computations. Intuitively, it is a reasonable assumption when there is a large number of agents, because such a mixed policy is equivalent to aggregating (differing) individual agent policies.

It should be noted that the best response computation varies from agent to agent depending on their rationality criterion as mentioned in Section 4. Furthermore, best response computation (as mentioned earlier) requires state distribution of other agents. State distributions of other agents can be computed given the starting distribution, type of agents and their policies. Formally, we let (a) $p^t_\tau(s)$ be the probability that an agent of type τ will be in s at time t; (b) d^t_s denote the number of agents in state s at time t; (c) $\mathcal{P}_\tau$ denotes the set of players of type τ.

$$p^0_\tau(s) = \frac{d^0_s}{|\mathcal{P}|},$$

$$d^t_s = \sum_{\tau \in \Gamma} p^t_\tau(s) \cdot |\mathcal{P}_\tau|, \ \forall t \tag{2}$$

$$p^{t+1}_\tau(s) = \sum_{a^t_\tau} \pi^t_\tau(s, a^t_\tau) \sum_{s' \in \mathcal{S}} p^t_\tau(s') \phi^t_{d^t}(s', a^t_\tau, s) \tag{3}$$

For a perfectly rational agent, the best response at each iteration is computed by solving an MDP where the ϕ and $\mathcal{R}$ are fixed for the state and action sets because the state distribution of other agents $(d^t, \forall t)$ can be computed for each decision epoch using Equation 3. Solving this best response MDP using traditional LP based methods or dynamic programming methods yields a deterministic policy (i.e. one action for each state). Due to this determinism, the aggregation of policies in fictitious play can take many iterations to converge. Therefore, we propose the use of Soft-Max operator for solving MDPs similar to the Soft-Max Value Iteration [15]. This implies that the "max" operator in standard value iteration is replaced by a "soft-max" operator. The soft-max operator makes the strategy equivalent to the quantal

response strategy indicated earlier.

We provide the intuition using the following example.

Example 2 *Consider an MDP with two actions, a_1 and a_2, each of which can be executed from a state s. $\mathcal{V}(s,a_1)$ and $\mathcal{V}(s,a_2)$ are the expected values for executing actions a_1 and a_2 respectively from state s. The policy at s with a standard MDP solver (policy iteration, value iteration etc.), π^1 and SoFA, π^2 would be:*

$$\pi^1(s,a_1) = \begin{cases} 1, & \text{if } a_1 = \arg\max_{\{a_i \in \{a1,a2\}\}} \mathcal{V}(s,a_i) \\ 0, & \text{otherwise} \end{cases}$$

$$\pi^2(s,a_1) = \frac{e^{\mathcal{V}(s,a_1)}}{e^{\mathcal{V}(s,a_1)} + e^{\mathcal{V}(s,a_2)}}$$

Algorithm 1 provides the pseudo code for the SoFA algorithm. Lines 6-13 provide the core of the algorithm. At each iteration of the algorithm, for each agent there are three key steps: (a) Firstly we compute the state distribution of all other agents (using Equation 3) on line 9; (b) Then, we compute the best response (depends on the agent's rationality criterion) corresponding to the aggregated policies (across iterations) of other agents on line 10; and (c) Finally, we calculate the aggregate state action flows and consequently the aggregate policy for the current agent on lines 11-12.

6 Theoretical Results

We provide key theoretical properties of our SoFA algorithm. Firstly, we will show that SoFA converges to equilibrium policies in cases where there are only locally rational agents ($L = 0$) in the mix. Secondly, we show that if SoFA converges on problems with multiple agent types (different transition/reward model or different notions of rationality), then perfectly rational agents will converge to equilibrium policies.

Proposition 1 *In an assorted population of perfectly rational agents and locally rational agents ($L = 0$), if there exists one type (w.r.t transition and reward model) of perfectly rational agents, then irrespective of the proportion of locally rational agents, SoFA algorithm converges to equilibrium policies for the perfectly rational agents. This is assuming Equation 3 can be used to compute update of probability.*

Proof. Let the population, $\mathcal{P}$ be divided into two sets $\mathcal{P}_r$ (perfectly rational agents) and $\mathcal{P}_l$ (locally rational agents), i.e. $\mathcal{P} = \mathcal{P}_l \cup \mathcal{P}_r$. If there exists a potential function for this DAAP, then SoFA would converge to Nash equilibrium. We now define a function $\phi^t()$ and then show that it is a potential function for the DAAP problem. (In the interest of space, we use Π_{ik} to represent the set of $\{\pi_i\}_{i\neq k, i\in\mathcal{P}}$ throughout the proof.)

$$\phi^t(\{\pi_i\}_{i\in\mathcal{P}}) = \sum_k \mathcal{V}_i^t(\pi_k, \Pi_{ik}) \tag{4}$$

To show that this is a potential function for a given DAAP, we need to show that for any arbitrary agent k and two of its policies, $\pi_{(k,1)}$ and $\pi_{(k,2)}$:

$$\begin{aligned} &\phi^t(\Pi_{ik} \cup \pi_{(k,1)}) - \phi^t(\Pi_{ik} \cup \pi_{(k,2)}) \\ &= \mathcal{V}_k^t(\pi_{(k,1)}, \Pi_{ik}) - \mathcal{V}_k^t(\pi_{(k,2)}, \Pi_{ik}) \end{aligned} \tag{5}$$

We prove this proposition using mathematical induction over the time horizon t.

Base case for t = 0
For $k \in \mathcal{P}_r$, the value function for an agent is given by:

$$\mathcal{V}_k^0(\pi_{(k,1)}, \Pi_{ik}) = \sum_{s,a} p_k^0(s) \cdot \pi_{(k,1)}(s,a) \cdot \mathcal{R}_k(s,a,d^0)$$

For $k \in \mathcal{P}_l$, the only difference in value function from above would be $d^0 = \mathbf{0}$. It is easy to see that for all agents except k (irrespective of whether $k \in \mathcal{P}_l$ or $k \in \mathcal{P}_r$), the value remains the same if policy does not change for t = 0. Due to this Equation 5 holds (the terms for all other agents will cancel out).

Therefore, let us assume that the ϕ is a potential function for horizon t = m, i.e.,

$$\begin{aligned} &\phi^m(\Pi_{ik} \cup \pi_{(k,1)}) - \phi^m(\Pi_{ik} \cup \pi_{(k,2)}) \\ &= \mathcal{V}_k^m(\pi_{(k,1)}, \Pi_{ik}) - \mathcal{V}_k^m(\pi_{(k,2)}, \Pi_{ik}) \end{aligned} \tag{6}$$

Now, we will prove that it holds for $t = m + 1$

Let us consider the case where $k \in \mathcal{P}_r$,

$$\begin{aligned} \mathcal{V}_k^{m+1}(\pi_{(k,1)}, \Pi_{ik}) &= \sum_{s,a} p_k^0(s) \cdot \pi_{(k,1)}(s,a) \cdot \mathcal{R}_k(s,a,d^0) \\ &+ \sum_{s,a,s'} p_k^0(s) \cdot \pi_{(k,1)}(s,a) \cdot \phi_k(s,a,s',d^0) \cdot \\ &\quad \mathcal{V}_k^m(s', \pi_{(k,1)}, \Pi_{ik}) \end{aligned}$$

From Equation 3

$$\begin{aligned} &= \sum_{s,a} p_k^0(s) \cdot \pi_{(k,1)}(s,a) \cdot \mathcal{R}_k(s,a,d^0) + \\ &\quad \sum_{s'} p_k^1(s') \cdot \mathcal{V}_k^m(s', \pi_{(k,1)}, \Pi_{ik}) \\ &= \sum_{s,a} p_k^0(s) \cdot \pi_{(k,1)}(s,a) \cdot \mathcal{R}_k(s,a,d^0) + \mathcal{V}_k^m(\pi_{(k,1)}, \Pi_{ik}) \end{aligned} \tag{7}$$

Using Equation 7 and from assumption of Equation 6, we have

$$\begin{aligned}
&\mathcal{V}_k^{m+1}(\pi_{(k,1)}, \Pi_{ik}) - \mathcal{V}_k^{m+1}(\pi_{(k,2)}, \Pi_{ik}) \\
&= \sum_{s,a} p_k^0(s) \cdot \pi_{(k,1)}(s,a) \cdot \mathcal{R}_k(s,a,d^0) \\
&\quad - \sum_{s,a} p_k^0(s) \cdot \pi_{(k,2)}(s,a) \cdot \mathcal{R}_k(s,a,d^0) \\
&\quad + \sum_{\{i \neq k, i \in \mathcal{P}\}, s, a} p_i^0(s) \cdot \pi_i(s,a) \cdot \mathcal{R}_i(s,a,d^0) \\
&\quad - \sum_{\{i \neq k, i \in \mathcal{P}\}, s, a} p_i^0(s) \cdot \pi_i(s,a) \cdot \mathcal{R}_i(s,a,d^0) \\
&\quad + \phi^m(\Pi_{ik} \cup \pi_{(k,1)}) - \phi^m(\Pi_{ik} \cup \pi_{(k,2)})
\end{aligned}$$

Combining terms using the definition of potential function (Equation 4), we have

$$= \phi^{m+1}(\Pi_{ik} \cup \pi_{(k,1)}) - \phi^{m+1}(\Pi_{ik} \cup \pi_{(k,2)})$$

Since $k \in \mathcal{P}_l$ is a sub-case of $k \in \mathcal{P}_r$, where $d^t = \mathbf{0}$, we can adopt the same steps and prove that Equation 5 holds. Hence the proof. ■

While the following proposition is straightforward, it is important because our algorithm converges on all the problems provided in the experimental results section.

Proposition 2 *For DAAP problems with multiple types of agents, if SoFA converges, then perfectly rational agents will not have any incentive (in terms of expected value) to deviate.*

Proof. If SoFA converges, it implies that the agent policy does not change (line 6) from the previous iteration after computing best response (corresponding to other agent policies). Therefore, perfectly rational agents will have converged to a policy where there is no incentive to deviate in terms of expected value. ■

7 Experimental Results

In this section, we evaluate policies generated by SoFA (perfectly rational if SoFA converges) within the context of the taxi fleet optimization problem, where only a proportion of the taxi drivers follow SoFA strategies and others follow strategies obtained from the cognitive hierarchy model described in Section 4. Firstly, we will perform the behavioral analysis of the real-world taxi data set to obtain the composition of the assorted human-agent population. Once the distributions for assorted human-agent populations are obtained, we will show the performance of SoFA on various synthetic and real-world problem sets.

7.1 Human Behavioral Analysis

We now introduce iterative Bayesian inference to compute the distribution over levels of reasoning for taxi drivers. Let $Pr_i^t(L)$ denote the probability of player i using *level-L* reasoning in iteration t. Let $f^t(L)$ be the mean probability of *level-L* reasoning in the population, and $f^t(L) = \frac{\sum_i Pr_i^t(L)}{N}$. The expected levels of reasoning after t iterations can be computed by: $mean^t = \sum_L L \cdot f^t(L)$. Let $g_L^t(h)$ denote the perceived proportion of $level-h$ agents by $level-L$ agents in iteration t, which can be computed by:

$$g_L^t(h) = \frac{f^t(L)}{\sum_{l=0}^{L-1} f^t(l)}, \ \forall h < L.$$

Given λ, T, and $\{g_L^t(h)\}$, we can compute the behavioral strategy of *level-L* agent in iteration t by considering actual distribution of taxis. The procedure is based on Bayesian update and its pseudo-code is listed in Algorithm 2. In Algorithm 2, the prior distribution is initialized to be uniform. The movements of free taxis provide necessary observations to continuously update the prior distribution.

The data set we use for computing average levels of reasoning consists of GPS traces from close to 8000 active taxis over 30 working days. Each record in the data set provides a timestamp, taxi coordinate, and the status of the taxi (hired or free). When the taxi is free and moves from one location to another, we consider it as a valid observation. We set the length of time period to be 5 minutes and only consider the morning peak hours from 6 to 9 AM. The data set is sliced into 10 3-day chunks, and the algorithm is executed one chunk at a time. On average, this gives us $90,000$ valid observations across all taxis for each run. We also set the look ahead $T = 1$ for behavioral analysis[3].

L^{max}	$\lambda = 1$	$\lambda = 3$
4	1.21 ± 0.04	1.50 ± 0.05
5	1.35 ± 0.04	1.58 ± 0.05

Table 1: Expected levels of reasoning.

Table 1 gives expected value of levels of reasoning for two different but uniform initial distributions and for $\lambda = \{1, 3\}$. $L^{max} = 4$ implies maximum levels of reasoning is set to 4 and each level $\{0 \cdots 4\}$ is assigned initial probability of 0.2. It should be pointed out that although the actual distribution we obtain is not Poisson, the mean values we obtained are well within the

[3]As a sanity check for the Bayesian inference, we compared the average revenue of taxis from the data set and using the cognitive hierarchy model. The values were within 1% of each other ($283 and $280 per day respectively).

Algorithm 2 Iterative Bayesian Inference(λ, T)

$Pr_i^0(L) = \text{UniformDistribution}(L^{\max})$
$it \leftarrow 0$
while $it \leq \text{MaxIterations}$ **do**
  $f^{it}(L) = \frac{\sum_i Pr_i^{it}(L)}{N}, \forall L$
  $g_L^{it}(h) = \frac{f^{it}(L)}{\sum_{l=0}^{L-1} f^t(l)}, \forall h < L, \forall L$
  for all DataSet **do**
    $\langle d^1 \cdots d^T \rangle \leftarrow \text{GetAgentDist}(\text{DataSet})$
    for all $L \leq L^{max}$ **do**
      $\pi^{it}(L) = \text{Behavior}(\lambda, T, g_L^{it}, \langle d^1 \cdots d^T \rangle)$
    end for
    $s_i = \text{ObservedStartZone}(\text{DataSet})$
    $a_i = \text{ObservedAction}(\text{DataSet})$
    $Pr_i^{it}(L|a) = \frac{\pi^{it}(L,s_i,a_i) \cdot Pr_i^{it}(L)}{\sum_l \pi^{it}(l,s_i,a_i) \cdot Pr_i^{it}(l)}, \forall L$
  end for
  $it \leftarrow it + 1$
end while

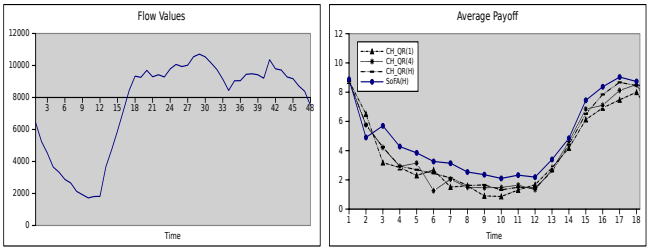
range of $[1, 2]$ which, as noted by Camerer *et al.*, can explain behavior in close to 100 games. The actual distribution we obtained are skewed towards lower levels of reasoning. A more rigorous study is needed to cross-validate this across larger data set and study the effect of look-ahead, T, on the final distribution.

7.2 Experimental Setup

To demonstrate that perfectly rational policies are useful in an assorted taxi driver population of different rationalities, we experiment on two different data sets. One is a synthetic data set that simulates random scenarios of demand distributions and another a real-world data set of a taxi company in Singapore. The synthetic data set was created with an inspiration from the real-world taxi data set and so we differentiate between peak hour and normal hours, multiple types of zones, time dependent flow-in and flow-out from a zone etc.

The main parameters of interest in the synthetic data set are (a) the map (defines how zones are connected); (b) total number of zones; (c) number of neighbors for each zone; (d) demand for all zones (represents people going into a zone) characterized by whether it is a residential zone, entertainment zone, office zone or a hospital; (e) total number of taxis; (f) total demand for taxis in a day; (f) length of peak hours. We generate problem instances considering a range of values for all these parameters.

Creating such a detailed simulation data set has helped us in evaluating our algorithm across multiple parameters and investigate the specific parameters affecting the final performance. For each map we compute policies by employing various algorithms and simulate them multiple times to obtain values for the comparison metrics.



(a) (b)

Figure 1: Customer flow at various points and the average payoff obtained by various approaches.

The real-world data set consists of flows of customers, revenues and costs for taxis between various zones across Singapore over a period of six months. We consider a time unit to be 30 minutes long and hence the time horizon for an entire day is $H = 48$. Figure 1 (a) depicts the aggregate demand for taxis throughout the day in this data set. The peak and non-peak hours are easily distinguishable. There are around 8000 taxis and 79 zones.

For all the graphs with aggregated results, we computed standard deviation as well. However, standard deviation is shown in the graphs only if it is greater than 0.1%. We evaluate the algorithms across the social welfare parameters highlighted in Section 2: (1) Average payoff of all taxis. (2) Minimum payoff across all taxis and (3) Starvation, which represents the lack of availability of taxis. The first two metrics provide an intuition for how the payoff distribution for taxi drivers changes. The third metric ensures an improvement from the system management (or taxi fleet owners) perspective.

We first present the results for the case when all agents adopt one type of rationality and then we will present the results for an assorted population. Although, we do not have theoretical guarantees on convergence when there are agents following behavioral strategies along with perfectly rational agents, empirically, we achieved convergence on all the problem instances. Our algorithm converged on all the problems in the synthetic and real-world data set within 2 hours[4].

7.3 Homogeneous Population

We compare the performance of SoFA algorithm against the strategies generated by the extended cognitive hierarchy model described in Section 7.1. While we experimented with a varied set of λ values, we only show results for $\lambda = \{1, 3\}$ because they yield the most

[4]The most amount of time was taken by the real-world problem set, where there are 8000 taxis and 79 zones.

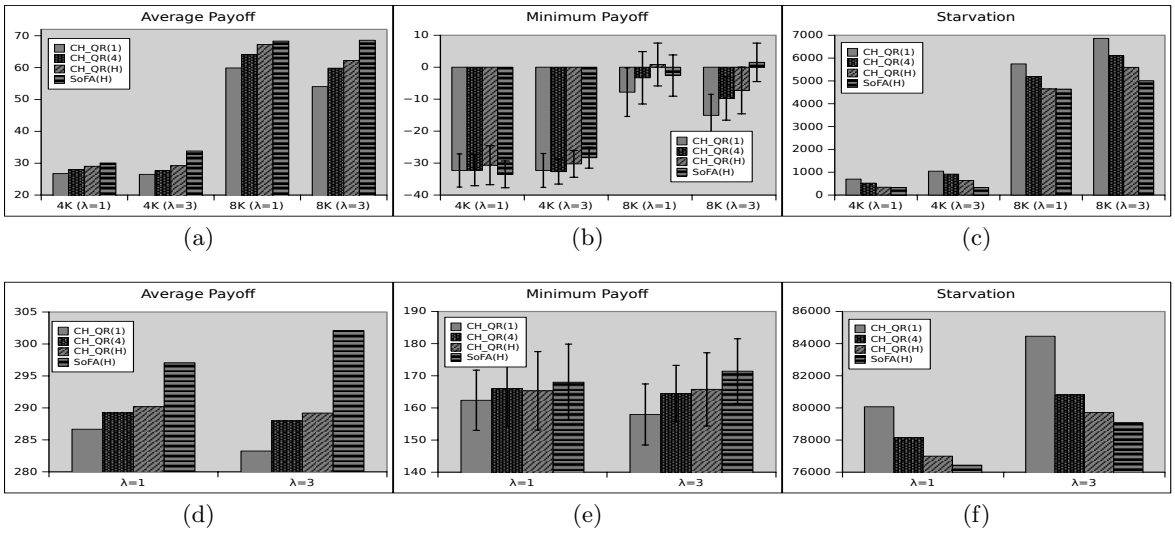


Figure 2: Performance of SoFA and behavioral strategies with $T = \{1, 4, H\}$ on synthetic and real-world datasets.

diverse set of observations. As for the time horizon, we considered $T = \{1, 4, H\}$. As for the levels of reasoning, L, we use the distributions obtained from iterative Bayesian inference corresponding to $\lambda = \{1, 3\}$. We also set $L^{max} = 4$. Thus, in our result graphs, $CH\_QR(5)$ with $\lambda = 3$ would imply the population consists of {17%, 39%, 26%, 13%, 5%} of levels $\{0 \cdots 4\}$ agents respectively with *look-ahead=5*. For $\lambda = 1$, it would be {20%, 52%, 17%, 5%, 6%}.

Figure 2 provides the results on the synthetic and real-world data sets. Graphs (a)-(c) are results for 40 zone problems and (d)-(f) are for the the taxi data set. Here are some of the key observations with respect to average and minimum payoff, starvation:

(i) For both synthetic and real-world problems, SoFA outperforms all the benchmark algorithms, due to the Nash equilibrium policies employed by SoFA agents.
(ii) Behavioral strategies with look ahead equal to horizon outperformed strategies with shorter look ahead.
(iii) We observe that for behavioral strategies, $\lambda = 1$ performs better than $\lambda = 3$. This could be because for $\lambda = 3$, the proportion of higher level reasoning agents increases and hence overall performance goes down. On the other hand, for SoFA, due to the randomized policies, performance increases from $\lambda = 1$ to $\lambda = 3$.
(iv) For the synthetic data set, when population size is $4k$, the behavioral strategy with $\lambda = 1$ and look ahead equals to *horizon* obtains marginally higher minimum payoff than all other algorithms. But when the population size is increased to $8k$ and with $\lambda = 3$, minimum payoff is guaranteed to be orders of magnitude higher under SoFA.
(v) For synthetic and real-world data sets, SoFA outperforms others for $\lambda = 3$ w.r.t. minimum payoff.
(vi) With respect to starvation (lower values are better), SoFA outperforms others on all data sets.

Figure 1(b) gives the average payoff for each algorithm at each time step on the real-world data set. We observe that SoFA performs better than other algorithms. Furthermore, it should be noted that there is high degree of similarity in the performance of SoFA over time and the flow values of Figure 1(a). This implies that SoFA is able to adapt quickly to the changes in demand over time.

7.4 Assorted Population

We now evaluate SoFA with an assorted population of agents. It should be noted that for a fixed λ and T, there are more than one type of agents in the population. We use X to denote the ratio of agents that are following SoFA policies. For instance, X = 20% (along X-axis in Figure 3) implies 20% agents are perfectly rational and the rest are following behavioral strategies for a particular λ.

Figure 3 provides the performance of SoFA within a varying population of agents: (a)-(c) and (d)-(f) are results for the synthetic and real-world data sets respectively. We make the following observations regarding average payoff, minimum payoff, and starvation:

(i) The performance of SoFA improves as the population of agents following behavioral strategies increases from 20% to 80%. The performance of boundedly rational agents at the same time goes down. It indicates that irrespective of the population of boundedly rational agents, it is always useful to employ a SoFA policy. SoFA outperforms behavioral strategies even when the proportion of agents following SoFA increases to 80%.
(ii) On the synthetic data set, SoFA outperforms other algorithms w.r.t. minimum revenue when the propor-

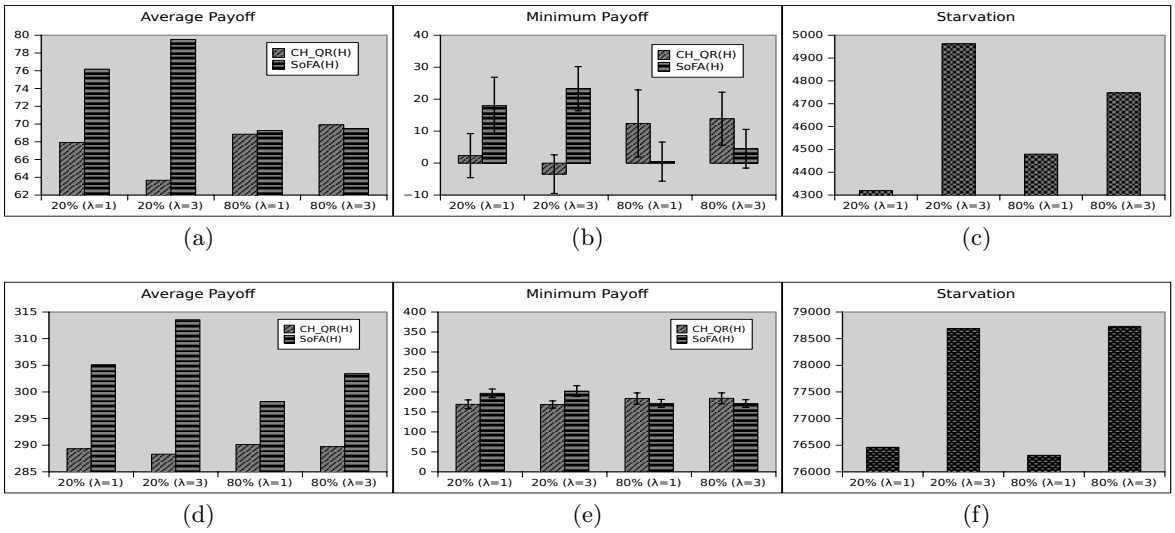


Figure 3: SoFA vs behavioral strategies in assorted population with $T = H$ on synthetic and real-world datasets.

tion of agents is 20%.
(iii) On real-world data set the minimum payoffs are almost equal across both strategy models.
(iv) Starvation values decrease with increase in the number of perfectly rational agents.

From all these experiments on both synthetic and real-world problem instances, we conclude that with respect to all welfare metrics (average revenue, minimum revenue and starvation), for any agent, it is useful to pursue the policy computed for perfectly rational agents in problems such as taxi fleet optimization.

8 Related Work and Conclusions

In this section, we briefly describe research related to the contributions made in this paper. The first thread of related research is in the field of transportation. User equilibrium (UE) is a classical and powerful equilibrium concept in transportation explaining individual route choices in face of competition for road usages from other users. Originally proposed in static setting [10], it was later expanded to dynamic cases (where temporal choices are also important) [3]. In either format, static or dynamic, the concept of UE provides a way to infer and to predict the behaviors of individual drivers; such ability helps not just individual drivers to identify better routes, but it can also help policy makers to properly design road network in anticipation of driver's responses. While the solution concept of UE is relevant, it does not account for the presence of involuntary movements for agents.

The second thread of related research is from *behavioral game theory* where different models have been proposed to accommodate experimental results (involving human subjects) and theoretical predictions. These include cognitive hierarchy model [1] and quantal response equilibrium [4]. More recently, variants of these models have been evaluated across different games [13]. As opposed to evaluation on traditional small benchmark problems, we have studied these models in the context of a very large scale real-world setting in this paper.

The next thread of relevant research is due to Weintraub *et al.* [12, 11]. This research introduces the concept of oblivious equilibrium for large scale dynamic games. They provide a mean field approximation to solve problems where there is stochasticity in state transitions. While, the problem is similar to DAAPs, the assumption of mean field (or a stationary distribution of taxis in our case) is not applicable in the context of taxi problems. In fact, there is a huge variance in the set of possible distributions at each decision epoch and hence oblivious equilibrium is not directly applicable in our context.

DAAP model represents a subset of problems represented by the generic Partially Observable Stochastic Games (POSG) model. However, all the approaches [6, 9, 8] provided in the literature are for solving identical payoff stochastic games (also referred to as Decentralized POMDPs or DEC-POMDPs) and not generic POSGs. Furthermore, they do not scale to problems with 8000 agents.

Acknowledgement

The research described in this paper was funded in part by the Singapore National Research Foundation (NRF) through the Singapore-MIT Alliance for Research and Technology (SMART) Center for Future Mobility (FM).

References

[1] C. F. Camerer, T.-H. Ho, and J.-K. Chong. A cognitive hierarchy model of games. *Quarterly Journal of Economics*, 119(3):861–898, 2004.

[2] J. Filar and K. Vrieze. *Competitive Markov decision processes*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[3] T. L. Friesz, D. Bernstein, T. E. Smith, R. L. Tobin, and B. W. Wie. A variational inequality formulation of the dynamic network user equilibrium problem. *Operations Research*, 41(1):179–191, 1993.

[4] R. D. McKelvey and T. R. Palfrey. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 2:6–38, 1995.

[5] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[6] S. Seuken and S. Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *UAI*, 2007.

[7] P. Varakantham, S.-F. Cheng, and T. D. Nguyen. Decentralized decision support for an agent population in dynamic and uncertain domains. In *10th International Conference on Autonomous Agents and Multiagent Systems*, pages 1147–1148, 2011.

[8] P. Varakantham, J. Y. Kwak, M. Taylor, J. Marecki, P. Scerri, and M. Tambe. Exploiting coordination locales in distributed POMDPs via social model shaping. In *ICAPS*, 2009.

[9] P. Velagapudi, P. Varakantham, K. Sycara, and P. Scerri. Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In *10th International Conference on Autonomous Agents and Multiagent Systems*, pages 955–962, 2011.

[10] J. G. Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of the Institute of Civil Engineers - Part II*, volume 1, pages 325–378, 1952.

[11] G. Weintraub, C. Benkard, and B. V. Roy. Markov perfect industry dynamics with many firms. *Econometrica*, 76(6):1375–1411, 2008.

[12] G. Y. Weintraub, L. Benkard, and B. V. Roy. Oblivious equilibrium: A mean field approximation for large-scale dynamic games. In *NIPS*, 2006.

[13] J. R. Wright and K. Leyton-Brown. Beyond equilibrium: Predicting human behavior in normal-form games. In *Twenty-Fourth Conference of the Association for the Advancement of Artificial Intelligence (AAAI-10)*, pages 901–907, 2010.

[14] H. Yang and S. C. Wong. A network model of urban taxi services. *Transportation Research Part B: Methodological*, 32(4):235–246, 1998.

[15] B. D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010.

Budget Optimization for Sponsored Search: Censored Learning in MDPs

Kareem Amin **Michael Kearns**
Computer and Information Science
University of Pennsylvania
{akareem,mkearns}@cis.upenn.edu

Peter Key **Anton Schwaighofer**
Microsoft Research
Cambridge, United Kingdom
{peter.key,antonsc}@microsoft.com

Abstract

We consider the budget optimization problem faced by an advertiser participating in repeated sponsored search auctions, seeking to maximize the number of clicks attained under that budget. We cast the budget optimization problem as a Markov Decision Process (MDP) with *censored observations*, and propose a learning algorithm based on the well-known *Kaplan-Meier* or *product-limit* estimator. We validate the performance of this algorithm by comparing it to several others on a large set of search auction data from Microsoft adCenter, demonstrating fast convergence to optimal performance.

1 Introduction

In this paper we study algorithms for optimized budget expenditure in sponsored search. Given an advertiser's budget, the goal of such algorithms is to maximize the number of clicks obtained during each budgeting period. We consider a single-slot model in which our algorithm's competing bid — representing the rest of the "market" for clicks — is drawn from a fixed and unknown probability distribution. While in reality, advertisers (or their proxies) may often bid strategically and not stochastically, we view this assumption as analogous to classical models in finance, where despite strategic behavior of traders at the individual level, models of macroscopic price evolution that are stochastic (such as Brownian motion models) have been quite effective in developing both models and algorithms. Our empirical results will demonstrate that algorithms designed for these stochastic assumptions also perform quite well on the non-stochastic sequence of bids actually occurring in real search auctions.

The assumption of stochastic bids by the competing market leads to a Markov Decision Process (MDP) formulation of the optimal policy, where the states of the MDP specify the remaining time in the period and the remaining budget. However, the second-price nature of sponsored search introduces the challenge of *censored* observations: only if we win the click do we observe the actual competing price; otherwise, we only know our bid was too low.

Our main contributions are the introduction of efficient algorithms that combine the MDP formulation with the classical Kaplan-Meier [5] or product-limit estimator for censored observations, and a large-scale empirical demonstration that these algorithms are extremely effective in practice — even when our underlying distributional assumptions are badly violated. Our source of data is auction-level observations on hundreds of high-volume key-phrases from Microsoft adCenter. We show that our algorithms rapidly learn to compete with the strongest possible benchmark — the performance of an offline-optimal algorithm that knows the future competing bids, and always selects the cheapest clicks in each period.

2 Related Work

There is some prior work directly concerning the problem of optimizing an advertiser's budget [6, 12], as well as work concerned with characterizing the dynamics or equilibria of a market in which advertisers play from a family of optimizing strategies [2, 3]. All these works attempt to model the strategic behavior of agents participating in a sponsored search auction. We will depart from this, modeling the auction market stochastically, as is more common in finance.

The sponsored search budget optimization problem has also been formulated as an instance of online knapsack [12]. For the online knapsack problem, it is known that no online algorithm can converge to the optimum in the worst case [8]. The stochastic knapsack problem has also been studied, and there is an algorithm with near-optimal average-case performance [7]. One of our

proposed algorithms is a censored learning version of such an algorithm. Our main proposed algorithm is closely modeled on an algorithm from a financial optimization problem [4], which similarly integrates a censored estimation step with greedy optimization.

Finally, we apply classical techniques from reinforcement learning in a finite state MDP, including Q-learning (c.f. [11]), as well as classical techniques from the study of censored observations [5, 9].

3 Preliminaries

The optimization problem we consider occurs over a series of *periods*. At the beginning of each period, the budget optimization algorithm is allocated a fresh budget B. This assumption is meant to reflect the manner in which advertisers actually specify their budgets in real sponsored search markets. Broadly speaking, the algorithm's goal is to maximize the number of clicks purchased, in each period, using the budget B.

Each period consists of a number of *auctions*, or an opportunity to earn a click. We consider the *single-slot* setting, wherein the search engine displays a single advertisement for each auction [1]. The algorithm places a *bid* in each auction. Before the bid is placed, a *price* is fixed by the auction mechanism. If the algorithm's bid exceeds this price, the algorithm wins an *impression*. For simplicity, we begin by assuming that winning an impression automatically guarantees that the algorithm also wins a click; we will describe later how to relax this assumption.

Once the algorithm wins a click, it is charged the price from its budget. Otherwise, the algorithm maintains its budget, and the next auction occurs. In actual sponsored search markets, the price is determined by the bids of arbitrary agents competing for the click in a modified second-price auction (see e.g. [10]).

The major assumption of this work is to instead model the prices as i.i.d. draws from an unknown distribution; we will refer to this price as the *market price*, since it represents the aggregate behavior of our algorithm's competitors. As in other financial applications, it is often analytically intractable to model the individual agents in a market strategically, so we instead consider the market stochastically as a whole. We will demonstrate that our algorithm outperforms other methods in practice on actual auction data from Microsoft adCenter, even when the i.i.d. assumption is badly violated.

We will consider some fixed, unknown, distribution $\mathcal{P}$ supported on $\mathbb{Z}^+$ with mass function $p(\cdot)$. On each auction, the market price is an independent random variable distributed as $\mathcal{P}$. We think of the budget B and market prices as being expressed in terms of the smallest unit of currency that can be bid by the algorithm. Modeling the problem in this manner motivates a natural algorithm.

It is important to note that the market prices are not observed directly by the algorithm. Rather, the algorithm is only privy to the consequences of its bid (whether a click or impression is received), and changes to its budget.

Succinctly, we consider the following protocol:

for period $u = 1, 2, ...$ **do**
  $B_{u,T} = B$
  for auctions remaining $t = T, T-1, ..., 1$ **do**
    Algorithm bids $b_{u,t} \leq B_{u,t}$.
    Nature draws price $x_{u,t} \sim \mathcal{P}$.
    if $b_{u,t} \geq x_{u,t}$ **then**
      $c_{u,t} \leftarrow 1$
      $B_{u,t-1} \leftarrow B_{u,t} - x_{u,t}$
    else
      $c_{u,t} \leftarrow 0$
      $B_{u,t-1} \leftarrow B_{u,t}$
    end if
    Algorithm observes $c_{u,t}$, $B_{u,t-1}$
  end for
end for

When an algorithm places a large enough a bid, winning the click, it also observes the true market price, since its budget it reduced by that amount. However, should the algorithm fail to win the click, it only knows that the market bid was higher than the bid placed. Thus, the algorithm receives what is known in the statistical literature as partially *right-censored observations* of the market prices $\{x_{u,t}\}$.

Informally, we always assume that an algorithm has available to it any information it would have in a real sponsored search auction (although not always at the same granularity), and no more. So it may be informed of whether it received a click or impression on an auction-by-auction basis, but not information regarding prices if it did not win the click.

Finally, we assume there is only a single keyword which the advertiser is bidding on. Our methods generalize to the setting where there are multiple keywords with multiple click-through rates and valuations for a click. However, for simplicity, we do not consider these extensions in this work.

[1] We suspect our methods can be adapted to the multi-slot case, but leave it to future work.

3.1 Notation

Given a distribution $\mathcal{P}$, supported on $\mathbb{Z}^+$, with mass function p, we let the tail function $T_p(b) = \sum_{b'=b+1}^{\infty} p(b')$ denote the mass to the right of b.

We use $[N]$ to mean the set $\{1, ..., N\}$, and $[N]_0 = [N] \cup \{0\}$.

4 MDP Formulation

An algorithm for the optimization problem introduced in the previous section can be described as an agent in a Markov Decision Process (MDP). An MDP $\mathcal{M}$ can be written as $\mathcal{M} = (\mathcal{S}, \{A_s\}_{s\in\mathcal{S}}, \mu, r)$ where $\mathcal{S}$ is a set of states, and A_s are the set of actions available to the agent in each state s, and $\mathcal{A} = \cup_{s\in\mathcal{S}} A_s$. For $a \in A_s$, $\mu(a, s, s')$ is the probability of transitioning from state s to state s' when taking action a in state s. $r(a, s, s')$ is the expected reward received after taking action a in state s and transitioning to state s'. The goal of the agent is to maximize the expected reward received while transitioning through the MDP.

In our case, the state space is given by $\mathcal{S} = [B]_0 \times [T]_0$. With t auctions remaining in period u, the algorithm is in state $(B_{u,t}, t) \in \mathcal{S}$. Furthermore, the actions available to any algorithm in such a state are the set of bids that are at most $B_{u,t}$. So for any $(b, t) \in \mathcal{S}$ we let $\mathcal{A}_{(b,t)} = [b]_0$.

When $t \geq 1$, two types of transitions are possible. The agent can transition from (b, t) to $(b, t-1)$ or from (b, t) to $(b', t-1)$ where $0 \leq b' < b$. In the former case, the agent must place a bid lower than the market price. Therefore, we have that $\mu(a, (b, t), (b, t-1)) = T_p(a)$. Furthermore, the agent does not win a click in this case, and $r(a, (b, t), (b, t-1)) = 0$. In the latter case, let $\delta = b - b'$. The agent must bid at least δ, and the market price must be exactly δ on auction t of the period in question. Therefore, we have that $\mu(a, (b, t), (b', t-1)) = p(\delta)$ and $r(a, (b, t), (b', t-1)) = 1$ so long as $a \geq \delta$.

When $t = 0$, for any action, the agent simply transitions to (B, T) with probability 1, with no reward. The agent's budget is refreshed, and the next period begins.

All other choices for $(a, s, s') \in \mathcal{A}_s \times \mathcal{S} \times \mathcal{S}$ represent invalid moves, and hence $\mu(a, s, s') = r(a, s, s') = 0$.

Finally, conditioned on an agent's choice of action a and current state s, its next state s is independent of all previous actions and states, since the market prices $\{x_{u,t}\}$ are independent. The Markov property is satisfied, and we indeed have an MDP.

We call this the *Sponsored Search MDP* (SS-MDP). If π is a fixed mapping from $\mathcal{S}$ to $\mathcal{A}$ satisfying $\pi(s) \in \mathcal{A}_s$, we say that π is a *policy* for the MDP.

Note that an agent in the SS-MDP, started in an arbitrary state (b, t), arrives at the state (B, T) after exactly $t + 1$ actions. Therefore, we can define the random variable $C_\pi(b, t)$ to be the total reward (i.e. number of clicks) attained by policy π before returning to (B, T). We say that π is *an optimal policy for the SS-MDP* if an agent started at (B, T), playing $\mu(s)$ in each state s encountered, maximizes the expected number of clicks rewarded before returning to (B, T). In other words:

Definition 1. *A policy π^* is an optimal policy for the SS-MDP if $\pi^* \in \arg\max_\pi E[C_\pi(B, T)]$.*

The SS-MDP is determined by the choice of budget B, time T and distribution p. We will want to make the optimal policy's dependence on p explicit, and consequently we will write it a π_p^*.

5 The Value Function

MDPs lend themselves to dynamic programming. Indeed, we can characterize exactly the optimal policy for the SS-MDP when the probability mass function p is known.

For a distribution p, let $V_p(b, t)$, the *value function for* p, denote the expected number of clicks received by an optimal policy started at state (b, t). That is, if π_p^* is an optimal policy, then $V_p(b, t) = E[C_{\pi^*}(b, t)]$.

First note that when $T = 0$, $V_p(B, T) \equiv 0$. Consider the policy $\pi^*_{a,b,t}$ that takes action a in state (b, t), and plays optimally thereafter. Let $V_p(a, b, t) = E[C_{\pi^*_{a,b,t}}(b, t)]$ be the clicks received by such a policy from state (b, t). Observe that $V_p(a, b, t)$ can be written in terms of $V_p(\cdot, t-1)$:

$$V_p(a, b, t) = \sum_{\delta=1}^{a} p(\delta)[1 + V_p(b - \delta, t-1)] + T_p(a)V_p(b, t-1).$$

In other words, if the market price is $\delta \leq a$, which occurs with probability $p(\delta)$, the agent will win a click at the price of δ and transition to state $(b - \delta, t - 1)$. At this point it behaves optimally, earning $V_p(b - \delta, t - 1)$ clicks in expectation. If the market price is greater than a, which occurs with probability $T_p(a)$, the agent will retain its budget, transitioning to the state $(b, t-1)$, earning $V_p(b, t-1)$ clicks in expectation. Furthermore, we know that:

$$V_p(b, t) = \arg\max_{a \leq b} V_p(a, b, t).$$

Therefore, if p and $V_p(\cdot, T-1)$ are known then we can compute $V_p(a, b, t)$ in $O(B)$ operations, and so

compute $V_p(b,t)$ in $O(B)$ operations. Recalling that $V_p(b,0) \equiv 0$, we can compute $V_p(b,t)$ for all $(b,t) \in [B]_0 \times [T]_0$ in $O(B^2T)$ operations.

6 Censored Data

In the previous sections, we described how to compute the optimal policy π_p^* when p is known. A natural algorithm for budget optimization is therefore to maintain an estimate $\hat{p}$ of p, and bid greedily according to $\pi_{\hat{p}}^*$. Before describing such an algorithm, we will discuss the problem of estimating p.

As introduced in Section 3, the observations received by a budget-optimization algorithm are partially right-censored data. We begin with a general discussion of censoring.

Suppose that $\mathcal{P}$ is a distribution with mass function p and $(z_1, ..., z_n)$ are i.i.d., $\mathcal{P}$-distributed random variables. Fix n integers $k_1, ..., k_n$, and define $o_i = \min(z_i, k_i)$. We say that the sample $\{o_i\}$ is partially right-censored data.

If $o_i < k_i$, we say that o_i is a direct observation. In other words, $o_i = z_i$ and we have observed the true value of z_i. Otherwise, $o_i = k_i$ and we say that o_i is a censored observation. We know only that $z_i \geq k_i$.

Given such partially right-censored data, the Product-Limit estimator [5] is the non-parametric maximum-likelihood estimator for p.

Definition 2. *Let $\mathcal{P}$ be a discrete distribution with mass function p, and let $\{z_i\}$ be i.i.d. $\mathcal{P}$-distributed random variables. Given integers $K = (k_1, ..., k_n)$ and observations $O = (o_1, ..., o_n)$ where $o_i = \min(z_i, k_i)$, let $PL(K, O)$ be the Product-Limit estimator for p.*

Specifically, given integers K, and a set of observations O generated by a distribution $\mathcal{P}$, let $D(s) = |\{o_i \in O \mid s = o_i < k_i\}|$ be the number of direct observations of value s, and $N(s) = |\{o_i \in O \mid s \leq o_i, s < k_i\}|$. Now let $S(t) = \prod_{s=1}^{t-1} 1 - \frac{D(s)}{N(s)}$. The CDF of $PL(K, O)$ is given by $1 - S(t)$.

In our setting, we are receiving censored observation of the random variables $\{x_{u,t}\}$ where the censoring set K is given by $\{b_{u,t}+1\}$, and $o_{u,t} = \min\{b_{u,t}+1, x_{u,t}\}$. When a click is received (i.e. $x_{u,t} < b_{u,t} + 1$), we observe $x_{u,t}$ directly since $x_{u,t} = B_{u,t} - B_{u,t-1}$, the amount which the algorithm is charged for winning the click. Otherwise, the algorithm is charged nothing, and we only know that $x_{u,t} \geq b_{u,t} + 1$.

Finally, we will eventually consider the setting in which winning an impression does not necessarily guarantee winning a click. In such a setting, we will have both left-censored and right-censored observations of $x_{u,t}$, what is known as *doubly-censored data*.

If the algorithm bids $b_{u,t}$ and does not win the impression, we know that $x_{u,t} \geq b_{u,t} + 1$. Similarly, if the algorithm wins both the impression and the click, it gets to observe $x_{u,t}$ directly. However, should the algorithm win the impression but not the click, it is only informed that it placed a large-enough bid (that it won the impression), or $x_{u,t} < b_{u,t} + 1$, without observing $x_{u,t}$ directly. We give a more detailed discussion of this setting in Section 9. For doubly-censored data, there is algorithm giving the non-parametric MLE [9].

7 Greedy Product-Limit Algorithm

The algorithm we propose maintains an estimate $\hat{p}$ of p. With budget b remaining, and t auctions remaining, the algorithm will greedily use its current estimate of p, and bid $\pi_{\hat{p}}(b,t)$. The pseudo-code for *Greedy Product-Limit* contains a detailed description.

Algorithm 1 Greedy Product-Limit

Input: Budget B

Initialize distribution $\hat{p}$ uniform on $[B]$
Initialize $K = []$; Initialize $O = []$
for period $u = 1, 2, ...$ **do**
  Set $B_{u,T} := B$
  for auctions remaining t=T,T-1,...,1 **do**
    Bid $\pi_{\hat{p}}^*(B_{u,t}, t)$
    Set $k_{u,t} \leftarrow \pi_{\hat{p}}^*(B_{u,t}, t) + 1$
    $K \leftarrow [K, k_{u,t}]$
    if Click won at price $x_{u,t}$ **then**
      $O \leftarrow [O, x_{u,t}]$
    else
      $O \leftarrow [O, k_{u,t}]$
    end if
    Update $\hat{p}$ to $PL(K, O)$
  end for
end for

8 Competing Algorithms

In this section we will describe a few alternative strategies against which we compare *Greedy Product-Limit*. The first relies on an observation that, given an *arbitrary* sequence of market prices, there is a simple bidding strategy that has a constant competitive ratio to the offline optimal.

8.1 Offline Optimality

So far we have focused our attention on the notion of optimality introduced in Section 4. Namely, an algorithm is optimal if it achieves $V_p(B,T)$ clicks,

in expectation, in every period. However, given an arbitrary (non-stochastic) vector of T market prices $\mathbf{x} = (x_1, ..., x_T)$, we can define $C^*(\mathbf{x}, B)$ to be the maximum number of clicks that could be attained by any sequence of bids, knowing $\mathbf{x}$ *a priori*. In other words, if $\mathbf{b} \in \{0,1\}^T$, and $\|\mathbf{b}\|_0 = |\{b_i \mid b_i = 1\}|$, then we define

Definition 3.

$$C^*(\mathbf{x}, B) \triangleq \max_{\mathbf{b} \in \{0,1\}^T} \|\mathbf{b}\|_0 \textit{ subject to } \mathbf{x} \cdot \mathbf{b} \leq B.$$

We call a sequence of bids for $\mathbf{x}$ that attains $C(\mathbf{x}, B)$ clicks an *optimal offline policy*. Notice that one attains the optimal offline policy by greedily selecting to win the clicks with the cheapest prices, until the budget B is saturated.

8.2 Fixed Price

Competing against the notion of optimality introduced in Section 8.1 may seem onerous in the online setting. Indeed, competing against an arbitrary sequence of prices is a special case of the online knapsack problem, which is known to be hard. However, we will now show that for any sequence of prices $\mathbf{x}$, there always exists a simplistic bidding policy which would have attained a constant factor of the bids of the optimal offline policy.

Let $Fixed(b)$ be the policy that bids b on every auction that it has budget to do so, and define $C(\mathbf{x}, b, B)$ to be the number of clicks attained by $Fixed(b)$ against $\mathbf{x}$ with budget B.

Theorem 1. *For any sequence of prices* $\mathbf{x}$*, and budget* B*, there exists a bid* b *such that* $C(\mathbf{x}, b, B) \geq \frac{1}{2} C^*(\mathbf{x}, B)$.

Proof. Let b^* be the value of the price for the most expensive click that the optimal offline policy selects to win. Suppose that the optimal offline policy wins $M+N$ clicks, where M clicks were won with a price of exactly b^* and the remaining N clicks were won with a price of $b^* - 1$ or less.

If $N \geq \frac{1}{2} C^*(\mathbf{x}, B)$, then $Fixed(b^* - 1)$ would win all N clicks, giving the desired result.

Otherwise, we know that $M \geq \frac{1}{2} C^*(\mathbf{x}, B)$. Consider the policy $Fixed(b^*)$. In the worst case, the policy will win only clicks with price b^* before saturating its budget. However, we know that $Mb^* \leq B$, and so $C(\mathbf{x}, b^*, B) \geq M \geq \frac{1}{2} C^*(\mathbf{x}, B)$, as desired.

□

8.3 Fixed-Price Search

This motivates a simple algorithm which attempts to find the best fixed-price, *Fixed-Price Search*.

Algorithm 2 Fixed-Price Search

```
Select b_1 arbitrarily.
for period u = 1, 2, ... do
    C_u := 0
    for auctions remaining t = T, T-1, ..., 1 do
        Bid b_u
        if Click won then
            C_u ← C_u + 1
        end if
    end for
    b_{u+1} ← UpdateBid({b_{u'}, C_{u'}}_{u'=1}^{u})
end for
```

The algorithm plays a fixed-price strategy each period. At the end of the period it uses a subroutine *UpdateBid* to select a new fixed-price according to how many clicks it has received. There are many reasonable ways to specify the *UpdateBid* subroutine, including using additive or multiplicative updates (e.g. treating each price as an expert and running a bandit algorithm such as Exp3 [1]). But general, the performance of any such approach cannot overcome the fixed-price "gap" of $E_{\mathbf{x}}[C^*(\mathbf{x}, B) - \max_b C(\mathbf{x}, b, B)]$.

8.4 Q-learning

Given the MDP formulation of the problem in Section 4, we may hope to solve the problem using techniques from reinforcement learning. Q-learning with exploration is one of the simplest algorithms for reinforcement learning, giving good results in a number of applications.

In Q-learning, the agent begins with an estimate $Q(a, b, t)$ of the function $V_p(a, b, t)$, called the Q-value, for each state (b, t) and action $a \in [b]$. In a state (b, t), the agent greedily performs the best action a^* for that state using the current Q-values receiving some reward $\hat{r}$ and arriving at a new state $(b', t-1)$. The Q-values for state $(b', t-1)$ and the observed reward $\hat{r}$ are then used to update $Q(a^*, b, t)$. This is often combined with forced exploration. Notice that Q-learning will necessarily ignore the special assumptions placed on the underlying MDP. In particular, from our discussion in Section 4, we have that $\pi(a, (b_1, t_1), (b'_1, t_1 - 1)) = \pi(a, (b_2, t_2), (b'_2, t_2 - 1))$ when $b_1 - b'_1 = b_2 - b'_2$.

8.5 Knapsack Approaches

As referenced in Section 2, the problem of budget optimization in sponsored search is very related to the

online knapsack problem. In the online knapsack problem, an optimizer is presented with a sequence of items with values and weights. At each time step, the optimizer makes an irrevocable decision to take the item (subtracting its weight from the optimizer's budget, and gaining its value). In the worst case, Marchetti-Spaccamela et al. demonstrate that a constant-factor competitive ratio with the offline is not possible [8]. Nevertheless, there are many results from the online knapsack literature that are applicable to our setting.

While in the worst case the online knapsack problem is hard, Lueker gives an average-case analysis for an algorithm for the *Stochastic Knapsack Problem*, which is related to our setting [7]. In the Stochastic Knapsack Problem, items $\{(r_i, x_i)\}$ are i.i.d. draws from some fixed, *known*, distribution. r_i is the profit or reward earned by taking the item, and x_i is the price of the item. In our setting, all clicks are considered indistinguishable for a fixed keyword, and so $r_i = 1$.

Note that the protocol differs from ours in a few ways. Firstly, there is no learning. The underlying distribution is assumed to be known. Secondly, there is no censoring of data, or any notion of an auction. The optimizer is presented with each item up-front, at which point it must make a decision before moving on to the next. Thirdly, in the language of our setting, there is only a single period.

Under these assumptions, the algorithm of Lueker gives a simple algorithm which differs from the true optimum by an average of $\Theta(T)$, where T is the length of the period, assuming that the budget available scales with T [7].

Nevertheless, the same ideas behind Greedy Product-Limit give us a natural adaptation of this algorithm to our setting. Suppose that the prices are presented up-front and that $\mathcal{P}$ is known. With budget B remaining and time T remaining in a period, the algorithm computes:

$$v(B/T) \triangleq \max_v \{v \mid \sum_{a=1}^{v} a \cdot p(a) \leq B/T\}$$

and takes the click iff its market price x satisfies $x \leq v(B/T)$. Thus, when the prices are not presented up-front, it is equivalent to simply bidding $v(B/T)$.

Note that bidding $v(B/T)$ is natural; it is the bid that, in expectation, costs B/T, or smooths the remaining budget over the time remaining. We can now combine this with an estimation step, using the product-limit estimator, as we did for *Greedy Product-Limit*, simply replacing line 7 with the assignment $k_{u,t} := v(B/T)$. We refer to this strategy as *LuekerLearn*.

Notice, however, that once $\hat{p}$ has converged to the true p, we should not expect this algorithm to outperform *Greedy Product-Limit*, which would bid optimally. For a fixed choice of distribution p, budget B, and number of auctions T, let $L_p(B,T)$ be the expected number of clicks earned by running the algorithm of Lueker, knowing p. $L_p(B,T) \leq V_p(B,T)$, by definition of $V_p(B,T)$. We will comment (as established by Lueker) that the gap $V_p(B,T) - L_p(B,T)$ is exacerbated by distributions with large variance relative to B and T, an issue we will return to in Section 9; see also Figure 1.

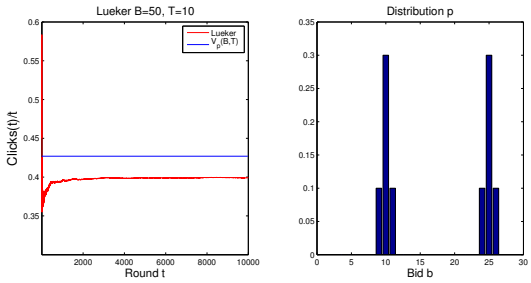


Figure 1: A distribution for which $L_p(B,T)$ is bounded away from $V_p(B,T)$. The red curve plots the performance (averaged over auctions) of the algorithm of Lueker's algorithm when the distribution p is known. The distribution is displayed on the right.

8.6 Budget Smoothing

There is also literature in which other budget-smoothing approaches are considered. Zhou et al. consider the budget optimization problem in sponsored search as a online knapsack problem directly [12]. In our setting, their algorithm guarantees a $\ln(B) + 1$ competitive ration with the offline optimum. Their algorithm smooths its budget over time, and operates by bidding: $1/(1+\exp(z(t)-1))$ where $z(t)$ is the fraction of budget remaining at time t.

9 Experimental Results

In this section we will describe experimental results for the previously described algorithms. We use bids placed through Microsoft's adCenter in two sets of experiments. In the first, we assume that our modeling assumptions from Section 3 are correct, and construct a distribution $\mathcal{P}$ from the empirical data for use in simulation. In the second set of experiments, we run the methods on the historical data directly, taken as an individual sequence and thus violating our stochastic assumptions. We will see that in both cases, our suggested algorithm outperforms the other methods discussed. First, however, we discuss an important generalization to the setting that we have considered

so far.

9.1 Impressions and Clicks

Until now we have assumed that all ad impressions result in a click (i.e. winning an auction results in an automatic click). We will now relax this assumption. Instead, when an advertiser wins an impression, we will suppose that whether a click occurs is an independent Bernoulli random variable with mean r. We call r the *click-through rate*. If a click does indeed occur, the advertiser is charged the market price. Otherwise, the advertiser is informed that an impression has occurred, but maintains its budget.

All the methods described generalize to this setting in a straightforward manner. Nevertheless, it is worth being explicit about how *Greedy Product-Limit* must be modified. First note that the MDP formulation for the problem differs in the definition of the transition probability μ. In particular, we now have $\mu(a,(b,t),(b-\delta,t-1)) = rp(\delta)$ (when $\delta \leq a$), and $\mu(a,(b,t),(b,t-1)) = (1 - r\sum_{\delta=1}^{a} p(\delta))$. π_p^* can still be computed using dynamic programming, where:

$V_p(a,B,T) = (1 - r\sum_{\delta=1}^{a} p(\delta))V_p(B,T-1) + \sum_{\delta=1}^{a} rp(\delta)[1 + V_p(B-\delta,T-1)]$

and $\pi_p^*(B,T) = \arg\max_{a \leq B} V_p(a,B,T)$

Furthermore, as discussed in Section 6, rather than using the Product-Limit estimator, this setting requires that the new algorithm treat doubly-censored data, for which techniques exist [9].

9.2 Data

The data used for these experiments were generated by collecting the auction history from advertisers placing bids through Microsoft's adCenter over a six month period. For a given keyword, we let the number of times that keyword generated an auction be its *search volume*, $\texttt{Vol}_k$, and take keyword k to be the keyword with the k-th largest volume.

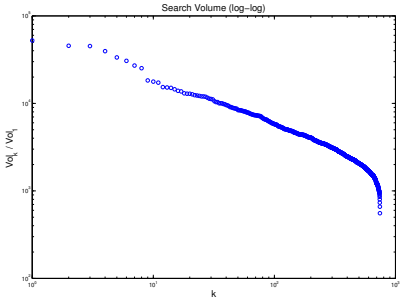


Figure 2: Log-log plot of $\texttt{Vol}_k/\texttt{Vol}_1$ for each k

The distribution of search volume is clearly heavy-tailed (see Figure 2) and is well approximated by a power law over several orders of magnitude.

We ran our experiments on the 100 keywords with largest search volume. As we will discuss further in the next section, the bidding behavior is quite varied among the different keywords in the data set.

In actual sponsored search-auctions, each bidder is given a *quality-score* for each keyword. Bidders' actual bids are multiplied by these quality scores to determine who wins the auction. For each keyword k, and auction t, let $x_{k,t}$ be the *quality-score-adjusted bid* for the advertiser that historically won the top slot for that auction, and $c_{k,t}$ indicate whether a click occurred.

We take the perspective of a new advertiser with unit quality score. $x_{k,t}$ is the amount that such an advertiser would have needed to bid to have instead taken the top slot (and the amount the advertiser would be charged should they also receive a click).

Finally, we make the single-slot assumption throughout, so even if multiple ads were indeed shown historically, we assume that an algorithm wins an impression if and only if it wins the top slot. In principle, *Greedy Product-Limit* can be modified to operate when multiple ad slots are available. However, we avoid this complication in this work.

9.3 Distributional Simulations

The first set of simulations use the historical data $\{x_{k,t}\}$, to construct an empirical distribution p_k on market prices for each keyword k. We also set a fixed click-through rate r_k, for each keyword using the background click-through rate for that keyword (the empirical average of $\{c_{k,t}\}$). While our main results in the next section eliminate the distribution p_k and use the sequence $\{x_{k,t}\}$ directly, we first simulate and investigate the case where our modeling assumptions hold.

Figure 3 demonstrates that the types of distributions generated in this manner are quite varied.

The budget B_k allocated to the advertiser for keyword k is selected so that, optimally, a constant fraction of clicks are available. In other words, the simulations were run with $B_k(T)$ satisfying $V_{p_k}(B_k(T),T) = fT$, where $f = 10\%$. Each experiment was run for 10 periods containing $T = 100$ auctions each, and 20 experiments were run for each keyword.

A major observation is that *Greedy Product-Limit* converges to the optimum policy on the time-scale of auctions, not periods. This is significant since certain methods considered are doomed to converge on the time-scale of periods instead. For example, each state

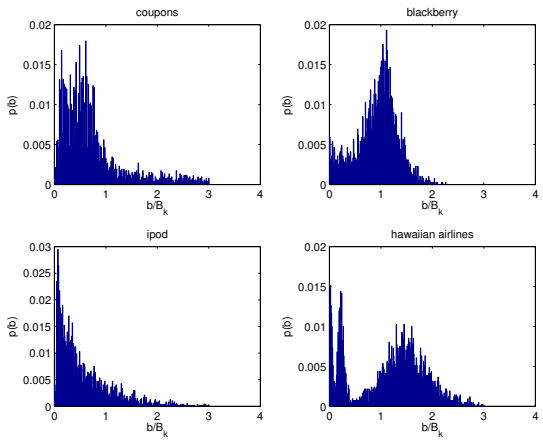


Figure 3: Each plot represents the empirical distribution p_k for a different keyword k. The x-axis represents the bid as a fraction of the total budget B_k allocated to the advertiser for keyword k.

can be visited at most once by Q-learning in a single period. Furthermore, for a particular time t, the only state corresponding to that t visited is $(B_{u,t}, t)$, (i.e. the state corresponding to the budget held by the algorithm at that time). Similarly, the Fixed-Price algorithm adjusts its bid at the end of every period. Table 1 shows that after just 2 periods, *Greedy Product-Limit* (GPL) has come within five percent of optimal across all keywords. For each algorithm, the results are the averages of 20 different simulations, and are statistically significant. An unpaired 2-sample t-test between the results for GPL and those of any other algorithm yields p-values that are less than 10^{-20}.

Table 1: Average Competitive Ratio with $V_{p_k}(B,T)$, across all keywords and experiments, after two periods.

Algorithm Name	Competitive Ratio	Std
Greedy Product-Limit	0.9573	0.1704
LuekerLearn	0.8448	0.1842
Fixed-Price Search	0.8352	0.1733
Q-learn	0.7484	0.1786
Budget Smoothing	0.1597	0.2418

9.4 Sequential Experiments

The main result of our work comes from experiments run on the real sequential data $\mathbf{d}_k = \{x_{k,t}, c_{k,t}\}_t$. Rather than taking $\{x_{k,t}, c_{k,t}\}$, and constructing the distribution p_k and a click-through rate, as in the previous section, we can use the sequence directly. Each of the previous methods are well-defined if the prices and clicks are generated in this manner, as opposed to being generated by the stochastic assumptions that motivated *Greedy Product-Limit*. We break the sequence into 10 periods of length $T = 100$. In reality, the number of auctions in a period might vary. However, this is minor, as an advertiser can attain good estimates of period-length. We allocate to each algorithm the same budget B_k used in the stochastic experiments.

Recall the notion of offline optimality defined in section 8.1. For each keyword k, we can compute the exact auctions that one should win knowing the sequence $\{x_{k,t}, c_{k,t}\}$ *a priori*. We will demonstrate that *Greedy Product-Limit* is competitive with even this strong notion.

We first look at the nature and time-scale of the convergence of *Greedy Product-Limit*.

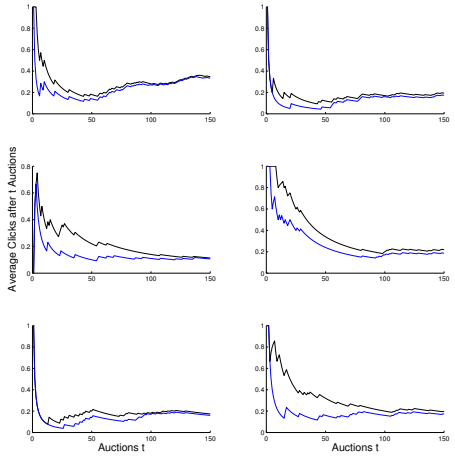


Figure 4: Convergence rates for 6 different keywords. The x-axis denotes auctions t, and the y-axis plots, in black, the number of clicks attained by the offline optimal after t auctions, normalized by t. The blue plot shows the same for *Greedy Product-Limit*. *Greedy Product-Limit* converges in the auction time-scale, not the period time-scale.

We will shortly see that *LuekerLearn*, the modification of *Greedy Product-Limit* attains similar performance. Like *Greedy Product-Limit*, it converges in the time-scale of auctions. However, recalling Figure 1, *LuekerLearn* will sometimes converge to something suboptimal, especially on keywords where there is a lot of variance in the bids. Figure 5 demonstrates this behavior on the sequential data. In fact, let A_k denote the cumulative number of clicks attained by *Greedy Product-Limit* and L_k denote the cumulative number of clicks attained by *LuekerLearn* after 10 periods for keyword k, defining $Z_k = A_k/L_k$ and $S_k = std(\{x_{k,t}\}_t)$. The observations $\{Z_k\}$ are positively correlated with $\{S_k\}$,

with a correlation coefficient of 0.2103 that is significant with a p-value of 0.0357.

Figure 6 demonstrates that, ignoring variance, *Greedy Product-Limit* has indeed converged to optimal across all keywords after just a single period. Let O_k be the offline optimal number of clicks that can be attained for keyword k after 10 periods (the entire data set). Let $A_{k,p}$ denote the cumulative number of clicks attained by an algorithm on keyword k after p periods. For an algorithm that is optimal after a single period, we should expect $A_{k,p}/O_k$ to be roughly $p/10$ for each period p. Indeed, while there are certain keywords for which this doesn't happen, we see that this is true on average.

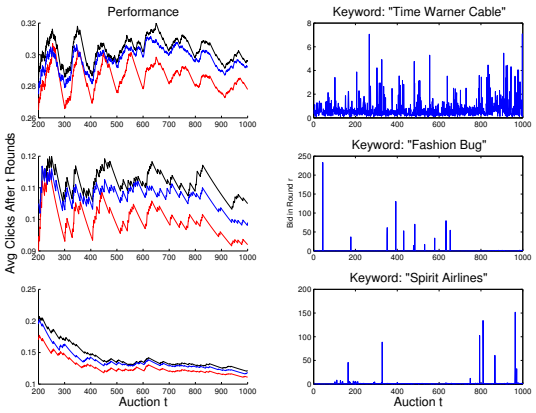


Figure 5: On the left the y-axis plots, in black, the number of clicks attained by the offline policy after t auctions, normalized by t. The blue plot shows the same for *Greedy Product-Limit* and the red for *LuekerLearn*. In all three, *LuekerLearn* is bounded away from the offline optimal. The right side displays $x_{k,t}/\bar{x}_k$ where $\bar{x}_k$ is the average over $x_{k,t}$ for the corresponding keyword. Note the "bursty" nature of the market price, with auctions occurring that set a market price hundreds of times greater than the average.

Figure 7 and Table 2 summarize the performance of the competing methods.

Table 2: Average Competitive Ratio with O_K, across all keywords.

Algorithm Name	Competitive Ratio	Std
Greedy Product-Limit	0.9062	0.1166
LuekerLearn	0.8962	0.1152
Fixed-Price Search	0.6253	0.1395
Q-learn	0.5879	0.1558
Budget Smoothing	0.3105	0.3252

Notice that besides *Greedy Product-Limit*, the only other algorithm that competes with the offline optimal is our modification to the stochastic knapsack algorithm, *LuekerLearn*. As previously discussed, the convergence of *Q-learn* and *Fixed-Price Search* happens on the time-scale or periods, not auctions. We suspect that with more data, both would converge (as they do in the stochastic setting), albeit *Fixed-Price Search* would converge only to the best fixed-price in hindsight. Finally, as demonstrated in Figure 5, when there is large variation in bidder behavior, *LuekerLearn* might stay bounded away from the offline optimal.

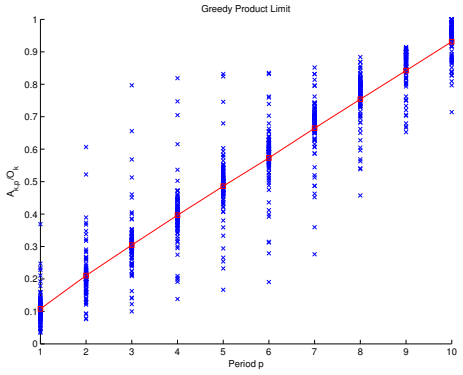


Figure 6: Each scatter point represents $A_{k,p}/O_k$ for a different keyword k, at the end of period p displayed on the x-axis. The line is the mean of $A_{k,p}/O_k$ across all k for a fixed period p.

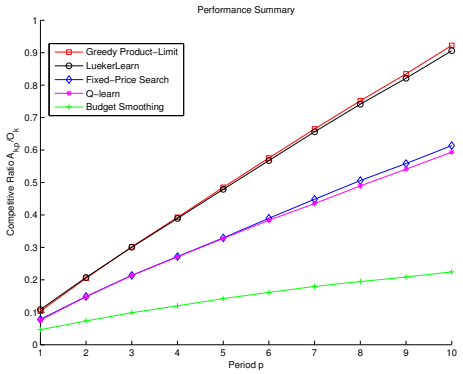


Figure 7: Average of $A_{k,p}/O_k$ across all k for each algorithm.

10 Future Work

We conjecture that *Greedy Product-Limit* always converges (rapidly) to the optimal policy in the stochastic setting, and hope to prove so in the future.

References

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2003.

[2] C. Borgs, J. Chayes, N. Immorlica, K. Jain, O. Etesami, and M. Mahdian. Dynamics of bid optimization in online advertisement auctions. In *Proceedings of the 16th international Conference on World Wide Web*, pages 531–540. ACM, 2007.

[3] M. Cary, A. Das, B. Edelman, I. Giotis, K. Heimerl, A.R. Karlin, C. Mathieu, and M. Schwarz. Greedy bidding strategies for keyword auctions. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 262–271. ACM, 2007.

[4] K. Ganchev, M. Kearns, Y. Nevmyvaka, and J. Wortman. Censored exploration and the dark pool problem. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

[5] E.L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, pages 457–481, 1958.

[6] B. Kitts and B. Leblanc. Optimal bidding on keyword auctions. *Electronic Markets*, 14(3):186–201, 2004.

[7] G.S. Lueker. Average-case analysis of off-line and on-line knapsack problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 179–188. Society for Industrial and Applied Mathematics, 1995.

[8] A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68(1):73–104, 1995.

[9] B.W. Turnbull. Nonparametric estimation of a survivorship function with doubly censored data. *Journal of the American Statistical Association*, pages 169–173, 1974.

[10] H Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

[11] C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[12] Y. Zhou, D. Chakrabarty, and R. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. *Internet and Network Economics*, pages 566–576, 2008.

Variational Dual-Tree Framework for Large-Scale Transition Matrix Approximation

Saeed Amizadeh
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15213

Bo Thiesson
Microsoft Research
Redmond, WA 98052

Milos Hauskrecht
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15213

Abstract

In recent years, non-parametric methods utilizing random walks on graphs have been used to solve a wide range of machine learning problems, but in their simplest form they do not scale well due to the quadratic complexity. In this paper, a new dual-tree based variational approach for approximating the transition matrix and efficiently performing the random walk is proposed. The approach exploits a connection between kernel density estimation, mixture modeling, and random walk on graphs in an optimization of the transition matrix for the data graph that ties together edge transitions probabilities that are similar. Compared to the de facto standard approximation method based on k-nearest-neighbors, we demonstrate order of magnitudes speedup without sacrificing accuracy for Label Propagation tasks on benchmark data sets in semi-supervised learning.

1 Introduction

Non-parametric methods utilizing random-walks on graphs have become very popular in Machine Learning during the last decade. These methods have been applied to solve Machine Learning problems as diverse as clustering (von Luxburg, 2007), dimensionality reduction (Lafon and Lee, 2006), semi-supervised learning (SSL) (Zhu, 2005), supervised learning (Yu et al., 2005), link analysis (Ng et al., 2001), and others. Unfortunately, these methods suffer from one fundamental and recurring problem: the quadratic dependency on the number of examples, which subsequently affects their scalability and applicability to large-scale data sets. Many different approximation frameworks have been proposed in the literature to tackle the above problem. Typically, these methods work by *sparsifying* the underlying graph representation by either reducing the number of nodes representing data points or reducing the number of edges representing data points similarities.

The first class of techniques aims to reduce the number of nodes in the graph from N to M ($M \ll N$). The reduction is carried out in various ways: random sampling (Kumar et al., 2009; Amizadeh et al., 2011), mixture modeling (Zhu and Lafferty, 2005), non-negative matrix factorization (Yu et al., 2005), sparse griding (Garcke and Griebel, 2005), Latent Markov Analysis (Lin, 2003), etc. A common problem with these techniques is that it is not clear how fast one should increase M when N increases. Subsequently, although the memory and the matrix-vector multiplication complexities are reduced to $O(M^2)$, we go back to $O(N^2)$ in effect if M changes with the same rate as N does.

The second class of techniques tries to sparsify the edges in the graph. k-nearest-neighbor graphs (Zhu, 2005) and b-matching (Jebara et al., 2009) are among the most famous methods in this group. The main concern with these methods, however, is the computational complexity of building these sparse graphs. Even with the smart speed-up techniques, such as k-nearest-neighbor graphs (Liaw et al., 2010; Moore, 1991), the actual time for building the full sparse graph can vary in practice.

A third class of methods goes around the problem of finding a sparse graph representation by trying to approximate the quantity of interest in the problem directly. Manifold regularization techniques (Belkin et al., 2006; Tsang and Kwok, 2006) directly find a smooth function on the graph. Fergus et al. (2009); Nadler et al. (2006); Amizadeh et al. (2012) try to directly estimate the eigen decomposition of the underlying transition matrix in the limit assuming factorized underlying distribution. Although these methods can be very effective, they are also task specific and do not give us an explicit approximation of the graph similarity (transition) structure.

The framework proposed in this paper directly approximates the transition matrix of the data graph. Our

method is similar to the second group of approximation techniques in the sense that it does not reduce the graph nodes. However, instead of zeroing out the edges (like in the second group), our framework groups the edges together to *share* edge transition probabilities that are similar. To do so, we use a recent advancement from Thiesson and Kim (2012), which for a very different task (fast mode-seeking) developed a fast dual-tree based (Gray and Moore, 2000) variational framework that exploits a mixture modeling view on kernel density estimation. Mixture modeling and the random walk on graphs are also closely related (Yu et al., 2005). Based on this connection between kernel density estimation, mixture modeling, and random walk on graphs, we extend the framework in Thiesson and Kim (2012) to a fast and scalable framework for 1) transition matrix approximation and 2) inference by the random walk.

By further exploiting the connection between mixture modeling and random walk, we also propose a lower-bound log-likelihood optimization technique to find the optimal bandwidth for the Gaussian similarity kernel (which is a very popular kernel for constructing transition matrices on data graphs). Moreover, using our framework, one can adjust the trade-off between accuracy and efficiency by *refining* the model. As we show in the paper, at its coarsest level of refinement, our framework achieves complexity order of $O(N^{1.5}\log N)$ for construction, $O(N)$ inference via matrix-vector multiplication, and memory consumption of $O(N)$, which improves the performance of graph sparsification methods reviewed earlier. To demonstrate the speed-up in practice, we experimentally show that without compromising much in terms of accuracy, our framework can build, represent and operate on random-walk transition matrices orders of magnitude faster than the baseline methods.

2 Background

Kernel density estimation plays an important conceptual role in this paper. Let $\mathcal{M} = \{m_1, m_2, \ldots, m_N\}$ denote a set of kernel centers corresponding to observed data $\mathcal{D}$ in $\mathbb{R}^d$. The kernel density estimate for any data point $x \in \mathbb{R}^d$ is then defined by the following weighted sum or mixture model:

$$p(x) = \sum_j p(m_j)p(x|m_j) \tag{1}$$

where $p(m_j) = \frac{1}{N}$, $p(x|m_j) = \frac{1}{Z_\sigma}k(x, m_j; \sigma)$, and k is a kernel profile with bandwidth σ and normalization constant Z_σ. For the commonly used Gaussian kernel, $k(x, m_j; \sigma) = \exp(-\|x - m_j\|^2/2\sigma^2)$ and $Z_\sigma = (2\pi\sigma^2)^{\frac{d}{2}}$.

Now let us assume we want kernel density estimates for data points $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ in $\mathbb{R}^d$ that also define the kernel centers. That is, $\mathcal{D} = \mathcal{M}$, and we use the $\mathcal{D}$ and $\mathcal{M}$ notation to emphasize the two different roles every example takes - one as a data point, the other as a kernel center. Let us exclude the kernel centered at $m_i = x_i$ from the kernel density estimate at that data point. In this case, the estimate at $x_i \in \mathcal{D}$ can be defined as the $N-1$ component mixture model:

$$p(x_i) = \sum_{j\neq i} p(m_j)p(x_i|m_j) = \sum_{j\neq i} \frac{1}{N-1}\frac{1}{Z_\sigma}k(x_i, m_j; \sigma), \tag{2}$$

where $p(m_j) = 1/(N-1)$.

In the mixture model interpretation for the kernel density estimate, the posterior kernel membership probability for a data point can be expressed as

$$p(m_j|x_i) = \frac{p(m_j)p(x_i|m_j)}{p(x_i)} = \frac{k(x_i, m_j; \sigma)}{\sum_{l=1}^{N} k(x_i, m_l; \sigma)} \tag{3}$$

These posteriors form a matrix $P = [p_{ij}]_{N\times N}$ where $p_{ij} = p(m_j|x_i)$ for $i \neq j$ are the true posteriors, and $p_{ij} = 0$ for $i = j$ are neutral elements added for later notational convenience. Note that in this representation, each row in P holds a posterior distribution $p_{i\cdot} \triangleq \{p_{ij}|j = 1, \ldots, N\}$.

The complexity of computing and representing P is $O(N^2)$, which is costly for large-scale data sets with large N. In this work, we seek ways of alleviating the problem by taking the advantage of the cluster structure in data (and kernels) that would let us *share* posteriors among groups of data points and kernels instead of computing them for each individual pair. More precisely, suppose that the data set $\mathcal{D}$ consists of two well-separated clusters in $\mathbb{R}^d$, denoted by C_1 and C_2. In this case, it is likely that the posteriors $p(m_j|x_i)$ are roughly similar for all $x_i \in C_1$ and $m_j \in C_2$ so that one can approximate all posteriors for the pairs in between these two clusters with one value. Notice how this approximation directly targets the posteriors and not the expensive intermediate kernel summation in the denominator of (3). This simple idea can be generalized recursively if we have nested cluster structure in the form of a *cluster hierarchy*. In this case, the computational savings can be dramatic.

3 Dual-tree Based Variational P

Whilst the approximation idea for P outlined above may appear simple, it needs careful handcrafting in order to yield a practical and computationally efficient framework. We start building our framework by borrowing the ideas from the dual-tree-based variational approach proposed by Thiesson and Kim (2012).

3.1 Dual-tree Block Partitioning

The dual-tree-based variational approach in Thiesson and Kim (2012) maintains two tree structures; the *data partition tree* and the *kernel partition tree*, that hierarchically partition data points (kernels) into disjoint subsets, such that an intermediate node in a tree represents a subset of data points (or kernels) and leaves correspond into singleton sets. In this work we assume the structure of the two trees is identical, leading to the exactly same subsets of data points and kernels represented by the tree.

The main reason for introducing the partition tree is to define relations and permit inferences for groups of related data points and kernels without the need to treat them individually. More specifically, we use the partition tree to induce a *block partition* of the matrix P, where all posteriors within the block are forced to be equal. We also refer to this partition as *block constraints* on P. Formally, a *valid* block partition $\mathcal{B}$ defines a mutually exclusive and exhaustive partition of P into blocks (or sub-matrices) $(A, B) \in \mathcal{B}$, where A and B are two *non-overlapping* subtrees in the partition tree. That is, $A \cap B = \emptyset$, in the sense that data-leaves in the subtree under A and kernel-leaves in the subtree under B do not overlap. (To maintain the convenient matrix representation of P, the singleton blocks representing the neutral diagonal elements in P are added to this partition.) Figure 1a) shows a small example with a valid block partition for a partition tree built for six data points (kernels). This block partition will, for example, enforce the block constraint $p_{13} = p_{14} = p_{23} = p_{24}$ for the block $(A, B) = (1\text{–}2, 3\text{–}4)$ (where $a-b$ denotes a through b).

To represent the block-constrained P matrix more compactly, we utilize the so-called *marked partition tree (MPT)* that annotates the partition tree by explicitly linking data groups to kernel groups in the block partition: for each block $(A, B) \in \mathcal{B}$, the data-node A is marked with the matching kernel-node B. Each node A in the MPT will therefore contain a, possibly empty, list of marked B nodes. We will denote this list of marks as $A_{mkd} \triangleq \{B|(A, B) \in \mathcal{B}\}$. Figure 1b) shows the MPT corresponding to the partition in Figure 1a). For example, the mark of kernels $B = 3{-}4$ at the node representing the data $A = 1{-}2$ corresponds to the same block constraint $(A, B) = (1-2, 3-4)$, as mentioned above. It is the only mark for this data node, and therefore $A_{mkd} = 1{-}2_{mkd} = \{3-4\}$. As another example, the list of marks for node $A = 5$ has two elements; $A_{mkd} = 5_{mkd} = \{5, 6\}$. An important technical observation, that we will use in the next section, is that each path from a leaf to the root in the MPT corresponds to the row indexed by that leaf in the block partition matrix for P. By storing the posterior probabilities at the marks in the MPT, we can therefore extract the entire posterior distribution $p_{i\cdot}$ by starting at the leaf that represents the data point x_i and follow the path to the root.

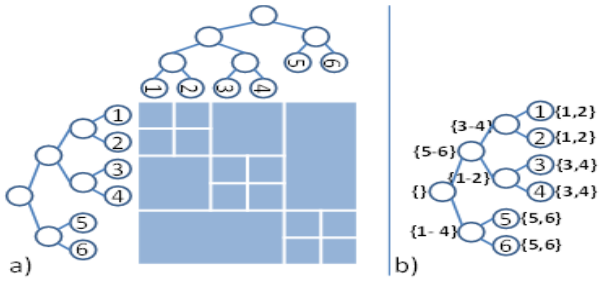


Figure 1: a) A block partition (table) for data partition tree (left) and identical kernel partition tree (top). b) MPT representation of the block partition.

Clearly, there are more than one valid partitions of P; in fact, any further *refinement* of a valid partition results in another valid partition with increased number of blocks. We postpone the discussion of how to choose an initial valid partition and how to refine it to Section 4.4. For now, let us assume that we are given a valid partition $\mathcal{B}$ of P with $|\mathcal{B}|$ number of blocks. By insisting on the block constraint of equality for all posteriors in a given block, the number of parameters in P is effectively reduced from N^2 to $|\mathcal{B}|$.

3.2 Optimization of Block Partitioning

In general, the block partition approach links groups of data points and kernels together such that any specific link is no longer able to distinguish its components. The key question now is whether we can cast the block partitioning problem into an optimization framework that can approximate well the true unconstrained P. We address the question by using a variational approximation approach. In this case, each block $(A, B) \in \mathcal{B}$ is assigned a single variational parameter q_{AB} that approximates the posteriors p_{ij} for all $x_i \in A$ and $m_j \in B$ in that block. That is, for $Q = [q_{ij}]_{N\times N}$

$$q_{ij} = q_{AB} \text{ s.t. } (A, B) \in \mathcal{B}, x_i \in A, m_j \in B \qquad (4)$$

Recall that in our setup, $q_{ij} = p_{ij} = 0$ for $i = j$.

The trick to solve the problem rests on expressing a variational lower bound for the log-likelihood of data:

$$\log p(\mathcal{D}) = \sum_i \log p(x_i) = \sum_i \log \sum_{j\neq i} p(m_j)p(x_i \mid m_j)$$
$$= \sum_i \log \sum_{j\neq i} \frac{q_{ij}}{q_{ij}} p(m_j)p(x_i \mid m_j)$$
$$\geq \sum_i \sum_{j\neq i} q_{ij} \log \frac{p(m_j)p(x_i \mid m_j)}{q_{ij}} \qquad (5)$$

$$= \log p(\mathcal{D}) - \sum_i D_{KL}\big(q_{i\cdot}\|p_{i\cdot}\big) \triangleq \ell(\mathcal{D}), \qquad (6)$$

where $D_{KL}(\cdot||\cdot)$ is the KL-divergence between two distributions. Notice that optimizing the lower bound $\ell(\mathcal{D})$ with respect to Q corresponds to minimizing the KL-divergence between the two distributions $q_{i\cdot}$ and $p_{i\cdot}$ for all i, since this is the only term in (6) that depends on Q. The block-constrained Q matrix is therefore a good approximation for the unconstrained P.

Let us now explicitly insert the block constraints from (4) into the expression for $\ell(\mathcal{D})$ in (5). With $p(m_i) = 1/(N-1)$ and the Gaussian (normalized) kernel for $p(x_i|m_j)$, the optimization reduces to finding the variational parameters q_{AB} that maximizes

$$\begin{aligned}\ell(\mathcal{D}) &= c - \frac{1}{2\sigma^2}\sum_{(A,B)\in\mathcal{B}} q_{AB}\cdot D^2_{AB} \\ &\quad - \sum_{(A,B)\in\mathcal{B}} |A||B|\cdot q_{AB}\log q_{AB}, \quad (7)\end{aligned}$$

where

$$\begin{aligned}c &= -N\log\big((2\pi)^{d/2}\sigma^d(N-1)\big) \\ D^2_{AB} &= \sum_{x_i\in A}\sum_{m_j\in B}\|x_i - m_j\|^2. \qquad (8)\end{aligned}$$

Thiesson and Kim (2012)[Algorithm 3] has developed a recursive algorithm that solves this optimization for all q_{AB}, $(A,B)\in\mathcal{B}$ in $O(|\mathcal{B}|)$ time. Specifically, their solution avoids the quadratic complexity that a direct computation of the Euclidean distances in (8) would demand by factorizing D^2_{AB} into data-specific and kernel-specific statistics as follows

$$D^2_{AB} = |A|S_2(B) + |B|S_2(A) - 2S_1(A)^T S_1(B), \quad (9)$$

where $S_1(A) = \sum_{x\in A} x$ and $S_2(A) = \sum_{x\in A} x^T x$ are the statistics of subtree A. These statistics can be incrementally computed and stored while the shared partition tree is being built; an $O(N)$ computation. Using these statistics, D^2_{AB} is computed in $O(1)$.

Now, the question is how to efficiently construct the shared partition tree from data. Well-known, efficient partition-tree construction methods include *anchor* tree (Moore, 2000), *kd*-tree (Moore, 1991) and *cover* tree (Ram et al., 2009; Beygelzimer et al., 2006) construction. We have used the anchor tree. With a relatively balanced tree, it takes $O(N^{1.5}\log N + |\mathcal{B}|)$ time and $O(|\mathcal{B}|)$ memory in total to build and store the variational approximation for the posterior matrix P from data. See the supplementary Appendix for more details on the anchor tree construction complexity.

4 Application to Random Walk

In this section, we show how the variational mixture modeling framework can be utilized to tackle large-scale random walk problems. The random walk here is defined on the graph $\mathcal{G}$ which is an undirected similarity graph whose nodes are the data points in $\mathcal{D}$ and the edge between nodes x_i and x_j is assigned the similarity weight $s_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$.

4.1 Variational Random Walk

As we saw in Section 2, $p(m_j|x_i)$ models the posterior membership of data point x_i to the Gaussian kernel m_j. However, $p(m_j|x_i)$ can be interpreted from a completely different view as the probability of jumping from data point x_i to data point m_j in a random walk on $\mathcal{G}$. In fact, (3) is the same formula that is used to compute the transition probabilities on data graphs, when the Gaussian similarity is used (Lafon and Lee, 2006). As a result, the matrix P from the previous section can be seen as the transition probability matrix for a random walk on $\mathcal{G}$. Subsequently, the corresponding variational approximation of Q is in fact the approximation of the transition probability matrix using only $|\mathcal{B}|$ number of parameters (blocks); in other words, q_{AB} approximates the probability of jumping from a data point in A to a data point in B. This is, in particular, important because it enables us to compute and store the transition probability matrix in $O(N^{1.5}\log N + |\mathcal{B}|)$ time and $O(|\mathcal{B}|)$ memory for large-scale problems.

In the light of this random walk view on Q, the lower bound log-likelihood $\ell(\mathcal{D})$ in (7) will also have a new interpretation. More precisely, the second term in (7) can be reformulated as:

$$-\frac{1}{2\sigma^2}\sum_{(A,B)\in\mathcal{B}} q_{AB}\cdot D^2_{AB} = -\frac{1}{2\sigma^2}\sum_{i,j} q_{ij}\cdot\bar{d}_{ij} \quad (10)$$

where, q_{ij} is defined as in (4) and $\bar{d}_{ij} = D^2_{AB}/|A||B|$ is the *block-average distance* such that $(A,B)\in\mathcal{B}$, $x_i\in A$ and $m_j\in B$. This is, in fact, a common optimization term in similarity-graph learning from mutual distances (Jebara et al., 2009) and can be more compactly represented as $-\frac{1}{2\sigma}tr(Q\bar{D})$, where $Q = [q_{ij}]_{N\times N}$ and $\bar{D} = [\bar{d}_{ij}]_{N\times N}$ are the similarity and distance matrices, respectively. However, there is one problem with a maximization of this term: it will make each point connect to its closest neighbor with $q_{ij} = 1$ (and $q_{ij} = 0$ for the rest). In other words, the similarity graph will be highly disconnected. This is where the third term in (7) benefits the new interpretation. One can rewrite this term as:

$$\begin{aligned}-\sum_{(A,B)\in\mathcal{B}} |A||B|\cdot q_{AB}\log q_{AB} &= -\sum_{i,j} q_{ij}\log q_{ij} \\ &= \sum_{i=1}^{N} H(q_{i\cdot}) \qquad (11)\end{aligned}$$

where $H(q_{i\cdot})$ is the entropy of the transition probability distribution from data point x_i. As opposed to (10), the term in (11) is maximized by a uniform distribution over the outgoing probabilities at each data point x_i; that is, a fully connected graph with equal transition probabilities. The third term in (7) therefore acts as the *regularizer* in the learning of the similarity-graph, trying to keep it connected. The trade-off between the second and the third terms is adjusted by the coefficient $1/2\sigma^2$: increasing σ will leave the graph more connected.

4.2 Learning σ

Finding the transition probabilities for a random walk by the variational optimization of (7) has a side advantage too: (7) is a *quasi-concave* function of the bandwidth σ, which means that, given q_{AB}'s fixed, one can find the optimal bandwidth that maximizes the log-likelihood lower bound. By taking the derivative and solving for σ, the closed form solution is:

$$\sigma^* = \sqrt{\frac{\sum_{(A,B)\in\mathcal{B}} q_{AB} \cdot D_{AB}^2}{Nd}} \tag{12}$$

In the special case where each element of matrix P is a singleton block (i.e. the most refined case), one can find σ^* independent of q_{AB}'s values. To do so, we first form the following log-likelihood lower bound using the Jensen inequality:

$$\log p(\mathcal{D}) \geq \sum_i \sum_{j\neq i} p(m_j) \log p(x_i \mid m_j) \tag{13}$$

By maximizing the right-hand side w.r.t. σ, we get:

$$\sigma^* = \frac{1}{N}\sqrt{\frac{\sum_i \sum_{j\neq i} \|x_i - x_j\|^2}{d}} \tag{14}$$

In general, due to the dependence of σ^* to q_{AB}'s, we alternate the optimization of q_{AB}'s and σ in our framework. In practice, we have observed that the convergence of this alternate optimization is fast and not sensitive to the initial value of σ.

4.3 Fast Inference

In the previous subsections, we saw how the variational dual-tree based framework can be used to efficiently build and store a variational transition matrix Q of a random walk. We will now demonstrate that the block structure of this transition matrix can be very useful for further inference in similarity-graph based learning algorithms. In particular, using the MPT representation of Q, we can efficiently compute the multiplication with an arbitrary vector $Y = (y_1, y_2, \ldots, y_n)^T$ of observations to achieve $\hat{Y} = QY \simeq PY$ in $O(|\mathcal{B}|)$ rather than $O(N^2)$ computations.

Algorithm 1 Calculate $\hat{Y} = QY$

Input: MPT with $\{q_{AB} : B \in A_{mkd}\}$ on each node A, and (x_i, y_i) on each leaf.
Output: $\hat{y}_i$ on each leaf.

```
A = Root(MPT)
CollectUp(A)
DistributeDown(A, 0)

function CollectUp(A)
  if IsLeaf(A) then
    T_A = y_A                    //null-vector if y_A is not observed
  else
    CollectUp(A_l)
    CollectUp(A_r)
    T_A = T_{A_l} + T_{A_r}
  end if
end function

function DistributeDown(A, py)
  for all marks B ∈ A_mkd do
    py += |B| q_AB T_A
  end for
  if IsLeaf(A) then
    ŷ_A = py
  else
    DistributeDown(A_l, py)
    DistributeDown(A_r, py)
  end if
end function
```

Algorithm 1 describes the $O(|\mathcal{B}|)$ computation of $\hat{Y}$. The algorithm assumes that each y_i is stored at the corresponding leaf x_i in the MPT–e.g., by association prior to constructing the MPT. Alternatively, an index to the leaves can be constructed in $O(N)$ time. The algorithm starts with a CollectUp phase that traverses the MPT bottom-up and stores incrementally computed sum-statistics at each node A, as

$$T_A = \sum_{x_i \in A} y_i = T_{A_l} + T_{A_r},$$

where A_l, A_r are the left and right children of A. This is an $O(N)$ computation. Let $\mathcal{B}(x_i) \triangleq \{(A,B) \in \mathcal{B} \mid x_i \in A\}$ denote the set of marked blocks that can be experienced on the path in the MPT from leaf x_i to the root. For example, in Figure 1, $\mathcal{B}(3) = \{(3,3), (3,4), (3{-}4, 1{-}2), (1{-}4, 5{-}6)\}$. With each q_{AB} stored at the node A marked with B in the MPT, a DistributeDown phase now conceptually computes

$$\hat{y}_i = \sum_{(A,B)\in\mathcal{B}(x_i)} |B| q_{AB} T_A \simeq \sum_j p_{ij} y_j,$$

by following the path from each leaf to the root in the MPT. The more efficient implementation in Algorithm 1 traverses the MPT top-down, avoiding recalculations of the shared terms for the updates by propagating the value of the shared terms through the variable py. This traversal has complexity $O(|\mathcal{B}|)$. On completion of the algorithm the leaves in the MPT will contain $\hat{Y} = \{y_i\}_{i=1}^N$. The fast matrix-vector multiplication is a significant achievement because it allows us to significantly speed up any algorithm with this computational bottleneck. Two such algorithms

are Label Propagation (LP) (Zhou et al., 2003) used for Semi-supervised Learning and Link Analysis (see, e.g., Ng et al. (2001)), and Arnoldi Iteration (Saad, 1992) used for spectral decomposition. More specifically, given a similarity graph over the data points x_i, the LP algorithm iteratively updates the label matrix $Y = [y_{ij}]_{N\times C}$ (where C is the number of label classes) at time $t+1$ by propagating the labels at time t one step forward according to the transition matrix P:

$$Y^{(t+1)} \leftarrow \alpha P Y^{(t)} + (1-\alpha)Y^0 \tag{15}$$

Here, the matrix Y^0 encodes the initial labeling of data such that $y^0_{ij} = 1$ for label$(x_i) = y_i = j$, and $y^0_{ij} = 0$ otherwise. The coefficient $\alpha \in (0,1)$ determines how fast the label values are updated. The repeated matrix multiplication in (15) definitely poses a bottleneck for large-scale problems, and this is where our fast framework comes into play.

4.4 Partitioning and Refinement

So far, we have assumed that a valid partition $\mathcal{B}$ of P with $|\mathcal{B}|$ number of blocks is given. In this section, we illustrate how to construct an initial valid partition of P and how to further refine it to increase accuracy of the model. We should note that the methods for partitioning and refinement in this section are different from the ones in Thiesson and Kim (2012).

Let $\mathcal{B}_{diag}$ denote the N neutral singleton blocks that appear on the diagonal of P. The coarsest (with the smallest number of blocks) valid partition $\mathcal{B}_c$ for P is achieved when for every block $(A,B) \in \mathcal{B}_c = \mathcal{B}\setminus\mathcal{B}_{diag}$, we have that A and B are sibling subtrees in the partition tree. (Recall that data and kernels are partitioned by the same tree.) Any other partition will either not be a valid partition conforming with the partition tree, or it will have a larger number of blocks. The number of blocks in $\mathcal{B}_c$, therefore equals twice the number of inner nodes in the anchor tree; i.e. $|\mathcal{B}_c| = 2(N-1)$. Figure 1 is an example of a coarsest valid block partition. On the other hand, the most refined partition is achieved when $\mathcal{B}_r = \mathcal{B}\setminus\mathcal{B}_{diag}$ contains $N^2 - N$ singleton blocks. Hence, the number of blocks in a valid partition $|\mathcal{B}|$ can vary between $O(N)$ and $O(N^2)$. As we saw in the previous sections, $|\mathcal{B}|$ plays a crucial role in the computational performance of the whole framework and we therefore want to keep it as small as possible. On the other hand, keeping $|\mathcal{B}|$ too small may excessively compromise the accuracy of the model. Therefore, the rational approach would be to start with the coarsest partition $\mathcal{B}_c$ and split the blocks in $\mathcal{B}_c$ into smaller ones only if needed. This process is called *refinement*. As we refine more blocks, the accuracy of the model effectively increases while its computational performance degrades. Note that a block (A,B) can be refined in two ways: either *vertically* into $\{(A_l,B),(A_r,B)\}$ or *horizontally* into $\{(A,B_l),(A,B_r)\}$. Here the subscript r (l) denotes the right (left) child node.

After any refinement, we can re-optimize (7) to find the new variational approximation Q for P. Importantly, any refinement loosens the constraints imposed on Q, implying that the KL-divergence from P cannot increase. From (6) we can therefore easily see that a refinement is likely to increase the log-likelihood lower bound $\ell(\mathcal{D})$, and can never decrease it. Intuitively, we want to refine those blocks which increase $\ell(\mathcal{D})$ the most. To find such blocks, one need to refine each block in each direction (i.e. horizonal and vertical) one at a time, re-optimize Q, find the difference between the new $\ell(\mathcal{D})$ and the old one (aka *log-likelihood gain*), and finally pick the refinements with maximum difference. However, this is an expensive process since we need to perform re-optimization per each possible refinement. Now the question is, whether we can obtain an estimate of the log-likelihood gain for each possible refinement without performing re-optimization. The answer is positive for horizontal refinements, and rests on the fact that each row in Q defines a (posterior) probability distribution and therefore must sum to one. In particular, this sum-to-one constraint can for $\mathcal{B}(x_i) \triangleq \{(A,B) \in \mathcal{B} \mid x_i \in A\}$ be expressed as

$$\textstyle\sum_{(A,B)\in\mathcal{B}(x_i)} |B| \cdot q_{AB} = 1 \text{ for all } x_i \in \mathcal{D}. \tag{16}$$

Consider the horizontal refinement of (A,B) into $\{(A,B_l),(A,B_r)\}$. By this refinement, in essence, we allow a random walk, where the probability of jumping from points in A to the ones in B_l (i.e. q_{AB_l}) is different from that of jumping from A to B_r (i.e. q_{AB_r}). If we keep the q values for other blocks fixed, we can still *locally* change q_{AB_l} and q_{AB_r} to increase $\ell(\mathcal{D})$. Since the sum of the outgoing probabilities from each point is 1 (see (16)) and the other q's are unchanged, the sum of the outgoing probabilities from A to B_l and B_r must be equal to that of the old one from A to B:

$$|B_l| q_{AB_l} + |B_r| q_{AB_r} = |B| q_{AB}. \tag{17}$$

Under this local constraint, we can find q_{AB_l} and q_{AB_r} in closed form such that $\ell(\mathcal{D})$ is maximized:

$$q_{AB_c} = \frac{|B|\exp(G_{AB_c}) q_{AB}}{\sum_{t\in\{l,r\}} |B_t|\exp(G_{AB_t})},\ c \in \{l,r\} \tag{18}$$

where $G_{AB} = -D^2_{AB}/(2\sigma^2|A||B|)$. By inserting (18) in (7), we can compute the maximum log-likelihood gain for the horizontal refinement of (A,B) as

$$\begin{aligned}\Delta^h_{AB} &= \ell'(\mathcal{D}) - \ell(\mathcal{D}) \\ &= |A||B| q_{AB} \cdot \log\left(\frac{\sum_{t\in\{l,r\}} |B_t|\exp(G_{AB_t})}{|B|\exp(G_{AB})}\right),\end{aligned} \tag{19}$$

where $\ell'(\mathcal{D})$ denotes the log-likelihood lower bound after the refinement. Note that the actual gain can be

greater than Δ^h_{AB} because after a refinement, all q values are re-optimized. In other words, Δ^h_{AB} is a lower bound for the actual gain and is only used to pick blocks for refinements. Unfortunately, for vertical refinements such a bound is not easily obtainable. The reason is that the constraints in (16) will not allow $q_{A_l B}$ and $q_{A_r B}$ to change if we fix the remaining q values. The local optimization that we applied for the horizontal refinement can therefore not be applied to estimate a vertical refinement. Whenever we pick a block (A, B) for vertical refinement, we therefore incorporate vertical refinements by applying the horizontal refinement to its symmetric counterpart (B, A) if it also belongs to $\mathcal{B}$. We call this *symmetric refinement*.

Now by computing Δ^h_{AB} for all blocks, we greedily pick the block with the maximum gain and apply symmetric refinement. The newly created blocks are then added to the pool of blocks and the process is repeated until the number of blocks reaches the maximum allowable block number $|\mathcal{B}|_{max}$, which is an input argument to the algorithm. This greedy algorithm can be efficiently implemented using a priority queue.

5 Experiments

In this section, we present the experimental evaluation of our framework. In particular, we have evaluated how well our method performs for semi-supervised learning (SSL) using Label Propagation (LP) (Zhou et al., 2003). The LP algorithm starts with a partial vector of labels and iteratively *propagates* those labels over the underlying similarity graph to unlabeled nodes (see (15)). After T iterations of propagation, the inferred labels at unlabeled nodes are compared against the true labels (which were held out) to compute Correct Classification Rate (CCR). We also measure the time (in ms) taken to build each model, propagate labels and refine each model. In our experiments, we set $T = 500$ and $\alpha = 0.01$. It should be noted that, here the goal is not to achieve a state-of-the-art SSL algorithm, but to relatively compare our framework to baselines in terms of efficiency and accuracy under the *same* conditions. This means that we have not tuned the SSL parameters to improve SSL; however, we use the same parameters for all competitor methods.

5.1 The Baselines

We have compared our method, *VariationalDT*, with two other methods for building and representing P. The first method is the straightforward computation of P using (3). We refer to this representation as the *exact model*. In terms of computational complexity, it takes $O(N^2)$ to build, store, and multiply a vector to P using the exact method. The second method is the *k-nearest-neighbor (kNN)* algorithm where each data point is connected only to its k closest neighbors. In other words, the rows of matrix P will contain only k non-zero entries such that P can be represented as a sparse matrix for small k's. In that way, kNN zeros out many of these parameters to make P sparse, as opposed to our VariationalDT method that groups and shares parameters in P. Note that we still assign weights to the k edges for each data point using (3). This means that as we increase k toward N, the model converges to the exact model. Therefore, k acts as a tuning parameter to trade off between computational efficiency and accuracy (the same role that $|\mathcal{B}|$ plays in VariationalDT). We call k and $|\mathcal{B}|$ the trade-off parameters.

Using the kNN representation, P can be stored and multiplied by an arbitrary vector in $O(kN)$. For constructing a kNN graph directly, the computational complexity is $O(rN^2)$ where $r = \min\{k, \log N\}$. A smarter approach is to use a metric tree in order to avoid unnecessary distance computations. Moore (1991) proposed a speedup of the kNN graph construction that utilizes a *kd-tree*. In our implementation of kNN, we have used the same algorithm with the *kd*-tree replaced by the anchor tree, introduced in Section 3. We call this algorithm *fast kNN*. The computational analysis of fast kNN greatly depends on the distribution of data points in the space. In the best case, it takes $O\big(N(N^{0.5}\log N + k\log k)\big)$ to build the kNN graph using fast kNN. However, in the worst case, the computational order is $O\big(N(N^{0.5}\log N + N\log k)\big)$. Table 1 summarizes the computational complexity orders for the models compared in the experiments. Note that for $|\mathcal{B}| = kN$ for some integer k, the VariationalDT and kNN will have the same memory and multiplication complexity. To increase k (or equivalently $|\mathcal{B}|$), one needs to refine the respective model. The last column in Table 1 contains the complexity of refining k to $k+1$ (or equivalently $|\mathcal{B}|$ from kN to $(k+1)N$) for kNN (or VariationalDT).

5.2 Experimental Setup

We have performed three experiments to evaluate the computational efficiency and accuracy of the models described above. It should be noted that for all methods, we have used the log-likelihood lower-bound technique in Section 4.2 to tune the bandwidth σ.

In the first experiment, we have run the LP algorithm for semi-supervised learning on the SecStr data set, one of the benchmark data sets for SSL (Chapelle et al., 2006). The data set consists of 83,679 different amino acids each of which is represented by 315 binary features. The task is to predict the secondary structure of the amino acids (2 classes). The goal in this experiment, as we increase the problem size N, is to study (a) the time needed to build the exact model, the

Models	Construction	Memory	Multiplication	Refinement
Exact	$O(N^2)$	$O(N^2)$	$O(N^2)$	N/A
Fast kNN	$O\big(N(N^{0.5}\log N + h\log k)\big)^*$	$O(kN)$	$O(kN)$	$O\big(N(\log N + N\log k)\big)$
Variational DT	$O(N^{1.5}\log N + \lvert\mathcal{B}\rvert)$	$O(\lvert\mathcal{B}\rvert)$	$O(\lvert\mathcal{B}\rvert)$	$O(\lvert\mathcal{B}\rvert\log\lvert\mathcal{B}\rvert)$

Table 1: Theoretical complexity analysis results. (*) h is equal to k in the best case and N in the worst case.

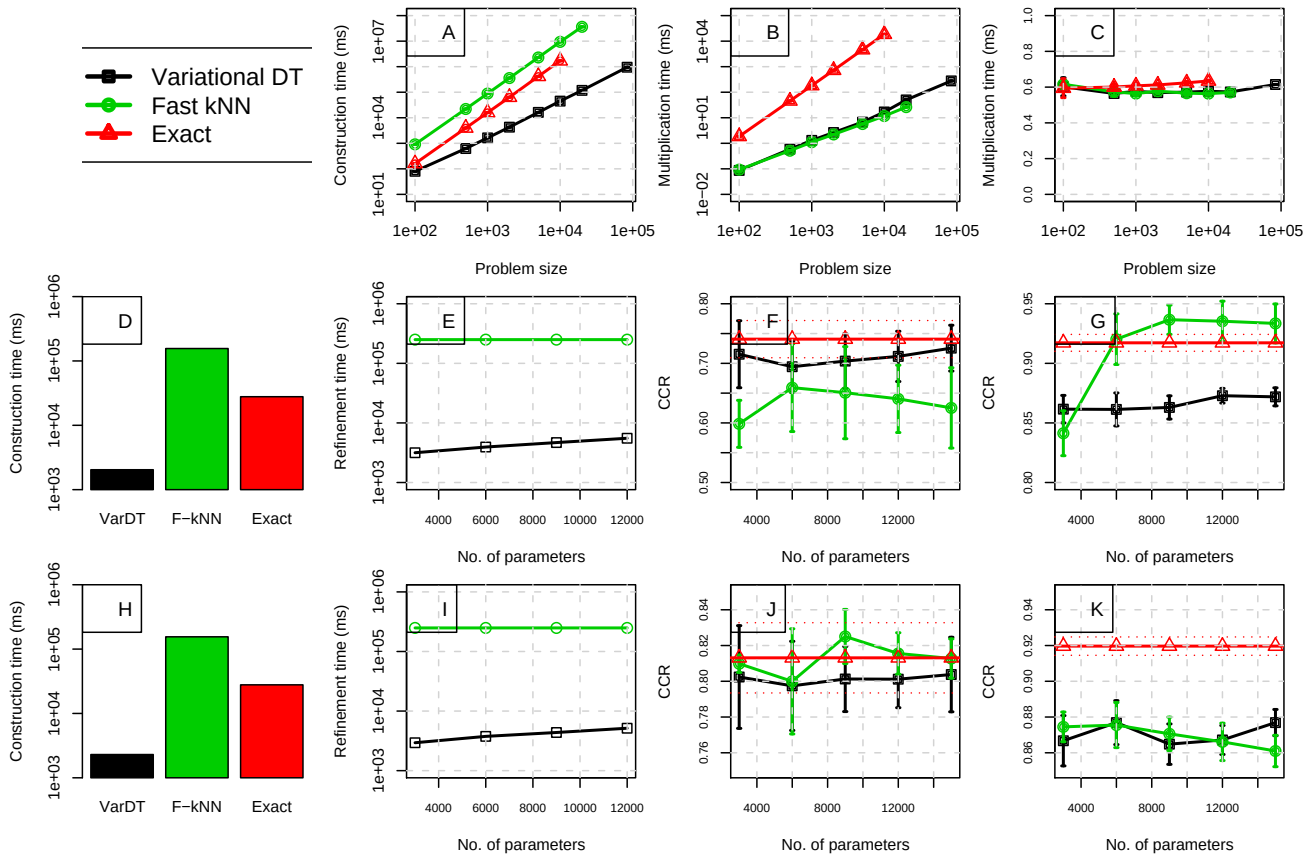


Figure 2: The experimental results for Variational DT, kNN and exact methods

coarsest Variational DT model (i.e. $|\mathcal{B}| = 2(N-1)$), and the coarsest kNN model (i.e. $k = 2$), (b) the time needed for multiplication in these models, and (c) CCR after label propagation given 10% labeled data. In particular, we draw samples of size s from the data set and use each sample to construct a model. Once a model is built, we choose 10% of the sample randomly to be fed to the LP algorithm as the labeled partition. We repeat this process 5 times for each problem size s and report the average result. Figure 2A) shows the construction time (in ms) for the three models as the problem size increases. As the plot shows, our method is orders of magnitude faster than the other two baselines (note that it is the log-log scale). In terms of the time needed for one multiplication, Figure 2B) shows that our method and kNN have similar complexity while both are order of magnitudes faster than the exact method. Note that Figure 2B) also shows the proportional memory usage for these methods because according to Table 1 multiplication and memory usage have the same complexity in all three methods. Finally, Figure 2C) depicts the CCR for the three models with different problem sizes. Although the exact model as expected is slightly more accurate than kNN and our method, the difference is not that big. In other words, by using VariationalDT we save orders of magnitudes in memory and CPU (i.e. both the construction and multiplication times) while compromising a little on accuracy.

In the second experiment, we study the efficiency and the effectiveness of the refinement process for the kNN and variationalDT models. To this end, first we build the coarsest kNN ($k = 2$) and VariationalDT ($|\mathcal{B}| = 2(N-1)$) models and then refine each model to

higher levels of refinement. At each refinement level, we make sure both methods have the same number of parameters; that is, $|\mathcal{B}| = kN$. We stop refining both models when the number of parameters reaches $O(N \log N)$. The data sets used for this experiment are Digit1 and USPS from the set of SLL benchmark data sets (Chapelle et al., 2006). Both data sets consist of digit images; while Digit1 is an artificially generated data set, USPS contains images of handwritten digits. Each data set has 1500 examples, 241 features and 2 classes. The second (third) row in Figure 2 shows the results for Digit1 (USPS) data set. Figures 2D,H) show the construction times for initial coarse models. Again our method is much faster than the baselines in terms of construction time. In Figures 2E,I), we have measured the time needed to refine each model to the next level of refinement. As the figure illustrates, VariationalDT needs an order of magnitude less time for refinement than kNN. Moreover, at each refinement level, we measure the CCR for LP when the size of the labeled set is 10 (Figure 2F,J) and 100 (Figure 2G,K). The red flat line in both plots depicts the CCR for the exact model. As the plots show, kNN and VariationalDT behave differently for different data sets and different sizes of labeled data. While refinement improves the kNN's performance significantly in Figure 2G, it degrades the kNN's performance in Figure 2F,K. We attribute this abrupt behavior to the *uniform* refinement of kNN graph. More precisely, when the kNN graph is refined, the degrees of all nodes are uniformly increased by 1 regardless of how much the log-likelihood lower bound is improved. Our method, on the other hand, explicitly aims to increase the log-likelihood lower bound resulting in a non-uniform refinement of P as well as more consistent behavior in terms of accuracy.

In the third experiment, we explore the applicability and the scalability of the proposed framework for very large data sets. The two data sets used in this experiment are taken from the Pascal Large-scale Learning Challenge.[1] The first data set, *alpha*, consists of 500,000 records with 500 dimensions. The second data set, *ocr*, is even larger with 3,500,000 records of 1156 features. Both data sets consist of 2 balanced classes. Table 2 shows the construction and propagations times as well as the number of model parameters when the Variational DT algorithm is applied. We could not apply other baseline methods for these data sets due to their infeasible construction times. Nevertheless, by extrapolating the graphs in Figure 2A, we guess the baselines would be roughly 3-4 orders of magnitude ($10^3 - 10^4$ times) slower. It should also be stressed that this experiment is specifically designed for showing the applicability of our framework for gigantic data sets and not necessarily showing its accuracy. However, even though we have not tuned the SSL parameters (like the labeling threshold), we still got $CCR = 0.56 \pm 0.01$ which is better than the random classifier. In conclusion, as a serial algorithm, our framework takes a reasonable time to construct and operate on these gigantic data sets. Furthermore, due to its tree-based structure, the Variational DT framework has the great potential to be parallelized which will make the algorithm even faster.

[1] http://largescale.ml.tu-berlin.de/

Data set	N	**Param#**	**Const.**	**Prop.**
alpha	0.5 M	1 M	4.5 hrs	11.7 min
ocr	3.5 M	7 M	46.2 hrs	93.3 min

Table 2: Very large-scale results

6 Conclusions

In this paper, we proposed a very efficient approximation framework based on a variational dual-tree method to estimate, store and perform inference with the transition matrix for random walk on large-scale data graphs. We also developed an unsupervised optimization technique to find the bandwidth for the Gaussian similarity kernel used in building the transition matrix. Algorithmically, we extended the variational dual-tree framework with a fast multiplication algorithm used for random walk inference with applications in large-scale label propagation and eigendecomposition. We also provided a complexity comparison between our framework and the popular k-nearest-neighbor method designed to tackle the same large-scale problems. In experiments, we demonstrated that while both the kNN based method and our method do not compromise much in terms of accuracy compared to the exact method, our method outperforms the kNN based method in terms of construction time with order of magnitudes difference. We also showed that our framework scales to gigantic data sets with millions of records.

Our future directions for this work include adding an inductive feature to our framework to deal with new examples as well as exploring the theoretical aspects of the approximation made by the algorithm.

Acknowledgements

This research work was supported by grants 1R01LM010019-01A1 and 1R01GM088224-01 from the NIH and by the Predoctoral Andrew Mellon Fellowship awarded to Saeed Amizadeh for the school year 2011-2012. The content of this paper is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

References

S. Amizadeh, S. Wang, and M. Hauskrecht. An efficient framework for constructing generalized locally-induced text metrics. In *IJCAI:, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1159–1164, 2011.

S. Amizadeh, H. Valizadegan, and M. Hauskrecht. Factorized diffusion map approximation. *Journal of Machine Learning Research - Proceedings Track*, 22: 37–46, 2012.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 97–104, New York, NY, USA, 2006. ACM.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collection. In *NIPS*, 2009.

J. Garcke and M. Griebel. Semi-supervised learning with sparse grids. In *Proc. of the 22nd ICML Workshop on Learning with Partially Classified Training Data*, 2005.

A. G. Gray and A. W. Moore. 'N-body' problems in statistical learning. In *NIPS*, pages 521–527. MIT Press, 2000.

T. Jebara, J. Wang, and S. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382, page 56. ACM, 2009. ISBN 978-1-60558-516-1.

S. Kumar, M. Mohri, and A. Talwalkar. Sampling techniques for the nystrom method. *Journal of Machine Learning Research - Proceedings Track*, 5:304–311, 2009.

S. Lafon and A. B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, September 2006.

Y. Liaw, M. Leou, and C. Wu. Fast exact k nearest neighbors search using an orthogonal search tree. *Pattern Recognition*, 43(6):2351–2358, 2010.

J. Lin. Reduced rank approximations of transition matrices. In *Proceedings of the Sixth International Conference on Artificial Intelligence and Statistics*, pages 3–6, 2003.

A. W. Moore. An introductory tutorial on kd-trees. Technical Report No. 209, Computer Laboratory, University of Cambridge, 1991.

A. W. Moore. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 397–405, 2000.

B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis. Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. In *Applied and Computational Harmonic Analysis: Special issue on Diffusion Maps and Wavelets*, 2006.

A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. *IJCAI*, pages 903–910, 2001.

P. Ram, D. Lee, W. B. March, and Alexander G. Gray. Linear-time algorithms for pairwise statistical problems. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009, Vancouver, British Columbia, Canada*, pages 1527–1535, 2009.

Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, Manchester, UK, 1992.

B. Thiesson and J. Kim. Fast variational mode-seeking. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics 2012, JMLR 22: W&CP 22*. Journal of Machine Learning Research, 2012.

I. W. Tsang and J. T. Kwok. Large-scale sparsified manifold regularization. In *NIPS*, pages 1401–1408. MIT Press, 2006.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

K. Yu, S. Yu, and V. Tresp. Blockwise supervised inference on large graphs. In *Proc. of the 22nd ICML Workshop on Learning*, 2005.

D. Zhou, O. Bousquet, T. Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*. MIT Press, 2003.

X. Zhu. *Semi-supervised learning with graphs*. PhD thesis, Pittsburgh, PA, USA, 2005.

X. Zhu and J. D. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *ICML: Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 1052–1059. ACM, 2005.

Exploiting Uniform Assignments in First-Order MPE

Udi Apsel and **Ronen I. Brafman**
Computer Science Dept.
Ben-Gurion University of The Negev
Beer-Sheva, Israel 84105
apsel,brafman@cs.bgu.ac.il

Abstract

The *MPE* (Most Probable Explanation) query plays an important role in probabilistic inference. MPE solution algorithms for probabilistic relational models essentially adapt existing belief assessment method, replacing summation with maximization. But the rich structure and symmetries captured by relational models together with the properties of the maximization operator offer an opportunity for additional simplification with potentially significant computational ramifications. Specifically, these models often have groups of variables that define symmetric distributions over some population of formulas. The maximizing choice for different elements of this group is the *same*. If we can realize this ahead of time, we can significantly reduce the size of the model by eliminating a potentially significant portion of random variables. This paper defines the notion of *uniformly assigned* and *partially uniformly assigned* sets of variables, shows how one can recognize these sets efficiently, and how the model can be greatly simplified once we recognize them, with little computational effort. We demonstrate the effectiveness of these ideas empirically on a number of models.

1 Introduction

Probabilistic relational models (PRM) [5] such as the *Markov Logic Network* (MLN) [13] and the *Parfactor model* [11] encapsulate a large number of random variables and potential functions using first-order predicate logic. Using exact lifted inference methods [2; 9; 12; 6; 4], one can perform inference tasks directly on the relational model, solving many tasks which were previously considered intractable.

To date, the first-order MPE query is perceived as a derivative of these methods, since its computational structure is similar to computing the partition function of a first-order model, with the exception of maxing-out random variables instead of summing-out, as laid out by Braz et al. [3].

In this paper we introduce a method that simplifies many first-order MPE tasks by utilizing properties specific to the maximization operator, allowing us to solve models in which computing the partition function is complicated, or even intractable.

The MPE property which we capitalize on is called a *uniform assignment* (UA). Briefly, it is the existence of a group of random variables that are assigned identically in some MPE solution. In relational models, these groups correspond to sets of ground atoms derived from the same atomic formula. Once a group is recognized as uniformly assigned, it can be replaced by a single representative, substantially simplifying the model. In relational models, this replacement corresponds to an arity reduction – the removal of some logical variable(s) from the relevant atomic formula. With suitable care, the MPE probability in the new model is the same as that of the original model, and the MPE for the original model is easily derived from the MPE of the new model.

The key contribution of this paper is a computationally efficient procedure for finding a set of uniform assignments in a given model. For this purpose, we define a set of purely symbolic operators (*anchoring*, *model alignment*, and *symbolic fusion*), which are all based on standard lifted inference tools, *fusion* [2] and *propositionalization* [9]. When applied repeatedly, the three symbolic operators serve as a structure analysis tool, and result in a compact representation of the UA property, expressing which logical variables can be removed from which atomic formulas.

Once uniform assignments are detected, we proceed to simplify our model by applying a reduction procedure called *uniform assignment reduction* (UAR). The procedure reduces the arity of relevant atomic formulas, and suitably modifies table entries in the model, ensuring that the MPE probability of the modified model remains equal to that of the original.

Both steps, UA detection and UA reduction, are preprocessing steps that often result in a much smaller model. Both incur low computational overhead that is independent of domain sizes, and are totally agnostic to the MPE algorithm later applied. As we demonstrate, the effort of UAR reduction is small w.r.t. to the complexity of the underlying inference task, and incurs virtually no overhead even when no simplification is achieved. Finally, the smaller, simplified, model is passed to your favorite MPE engine, and from the computed MPE, the MPE of the original model is easily extracted.

Our UA detection method is general, and can be applied both on MLNs and parfactor models. However, UA detection is most naturally formulated on parfactors, since it utilizes the (symbolic) fusion operator, which is not commonly applied to MLNs. We therefore start with a background of the parfactor model, before introducing the concepts of our method in a Walk-through section. Formal definitions and complete algorithm formulation are presented next, followed by an extension of UA to recursive conditioning, a presentation of the experimental results, and a conclusion section.

2 Background

We review the properties of the parfactor model, as introduced by Poole [11], and later extended by Braz et al. [2] and Milch et al. [9]. Readers familiar with this model can safely skip to the following Walk-through section.

2.1 Model Representation

The parfactor model is a first-order representation of a factor network, such as a Bayesian [10] or a Markov [13] network. Random variables in the model correspond to ground atomic formulas (aka *ground atoms*) of the form $p(c_1, \ldots, c_n)$, where p is a predicate with an assignment range $range(p)$, and $c_1, \ldots, c_n$ are constant symbols. Under a set of assignments v to random variables, the notation $\alpha_{|v}$ is used to depict the values assigned to α, whether α is a ground atom or a set of ground atoms. A *factor* f is a pair (A, ϕ), consisting of a set of ground atoms $A = \bigcup_{i=1}^{m}\{\alpha_i\}$ and a potential function $\phi : \prod_{i=1}^{m} range(\alpha_i) \to \mathbb{R}_0^+$. Under the set of assignments v, the weight of factor f is $w_f(v) = \phi(\alpha_{1|v}, \ldots, \alpha_{m|v})$. The abbreviation $\phi : \alpha_i, \ldots, \alpha_m$ is used to denote a factor with a potential table ϕ and a respective set of ground atoms.

Example 2.1. Let factor f consist of the following, $\phi : smokes(Alice), drinks(Bob), friends(Alice, Bob)$. Hence, the table entries in ϕ represent the various chances of $Alice$ and Bob being friends, depending on whether $Alice$ and Bob are drinkers/smokers.

Whereas factor networks are modeled by a set of factors, the core representation unit of our model is the parameterized factor (aka *parfactor*), a first-order extension of a factor. Before reviewing its properties, we define the notions of *atoms*, *logical variables*, *substitutions* and *constraints*.

An atomic formula (aka *atom*) is a formula of the form $p(t_1, \ldots, t_n)$, where t_i is a constant or a *logical variable*. Each logical variable X has a domain $dom(X)$ with cardinality $|X|$. For instance, $smokes(X)$ denotes an atomic formula over logical variable X, where, in our running example, the domain of X contains $Alice$ and Bob. $LV(\alpha)$ is the set of logical variables referred by α, where α is a formula or a set of formulas.

A *substitution* θ over a set of logical variables L maps each variable in L to a constant symbol or some other logical variable, and $\alpha\theta$ is the result of applying θ to α. A *ground substitution* is the substitution of all logical variables in α with constants. For instance, a substitution Y/X applied to $friends(X, Y)$ results in $friends(X, X)$, and a ground substitution $X/Alice$ applied to atom $smokes(X)$ results in the ground atom $smokes(Alice)$. The range of atom α, denoted by $range(\alpha)$, is defined as the range of any ground atom $\alpha\theta$ obtained by the ground substitution θ.

A *constraint* C is a set of formulas over a set of logical variables L, defining the set of legal substitutions applicable to L. For instance, the constraint $X \neq Y$ defines any substitution under which the substitutions of X and Y are identical, as illegal. We use $rv(\alpha : C)$ to depict the set of random variables that can be obtained by any legal substitution on α, as defined by C.

$gr(L : C)$ is the set of legal ground substitutions which can be applied on L under constraint C, where $|L : C|$ is used to depict the size of the set. We use $C_L^{\downarrow}$ to depict the projection of a set of logical variables L on a constraint C. Similarly to [9], we require the constraints in our model to be in some normal form, where for each logical variable X, $|X : C|$ has a fixed value regardless of the binding of other logical variables in C.

Lastly, a *parfactor* is a tuple (L, C, A, ϕ), comprised of a set of logical variables L, a constraint C on L, a set of atoms A and a potential function $\phi : \prod_{\alpha \in A} range(\alpha) \to \mathbb{R}_0^+$. Applying substitution θ on parfactor $g = (L, C, A, \phi)$ results in $g\theta = (L', C\theta, A\theta, \phi)$, where L' is obtained by applying a substitution on its logical variables, and dropping those who are mapped to constants. Parfactors compactly model a set of factors upon their *grounding*, namely – upon applying all legal ground substitutions, as defined by C. The operator $gr(g)$ depicts the set of all legal ground substitutions . The abbreviation $\phi : \alpha_1, \ldots, \alpha_m \mid C_\psi$ is used to describe a parfactor with the set of atoms $\bigcup_i\{\alpha_i\}$, a constraint with formula ψ, and an explicit set of logical variables.

Example 2.2. Let parfactor g consist of the following, $\phi : smokes(X), drinks(Y), friends(X, Y) \mid C_{X \neq Y}$. In a world where the domain of X and Y equals

$\{Alice, Bob\}$, the grounding of g entails two factors: $\phi: \ smokes(Alice), drinks(Bob), friends(Alice, Bob)$ $\phi: \ smokes(Bob), drinks(Alice), friends(Bob, Alice)$

The weight of g is defined as $w_g(v) = \prod_{f \in gr(g)} w_f(v)$, where v is an assignment to all the ground atoms (random variables) entailed by the the grounding of g. This set of random variables is conveniently denoted by $rv(g)$.

2.2 Shattering, Fusion and Propositionalization

Let model G be a set of parfactors, let α_i denote an atom, and C_i denote its parfactor constraint. We say that G is *completely shattered*, if for any i, j, the sets of random variables denoted by $rv(\alpha_i : C_i)$ and $rv(\alpha_j : C_j)$ are either disjoint or equal. Hence, *shattering* [2] is a procedure which takes model G, and produces a completely shattered model G', s.t. the set of all ground factors entailed by G and G' is equal. To eliminate some possible ambiguity, we refer to all atoms in a completely shattered model as each having a unique predicate symbol.

Example 2.3. Let model G consist of two parfactors, $\phi_1: \ p(X, Y), q(Y)$ and $\phi_2: \ p(X, X)$.
The shattering of G replaces the first parfactor with two new parfactors, consisting of the same potential, as follows $\phi_1: \ p(X, X), q(X)$ and $\phi_1: \ p(X, Y), q(Y) \mid C_{X \neq Y}$.

Fusion [2] is the procedure of merging two or more parfactors by unifying logical variables, while maintaining the same model weight under any given assignment. Since there are several ways to unify logical variables, inference engines apply fusion according to some strategy. For instance, consider the fusion of $\phi_1: \ p(X, Y), q(Z)$ and $\phi_2: \ p(X, Y), q(Y)$. Two possible fusions (out of many) are $\phi_{f_1}: \ p(X, Y), q(Z), q(Y)$ and $\phi_{f_2}: \ p(X, Y), p(X, Z), q(Y)$, and so the inference engine must choose which one to apply. In this paper, fusion is used only symbolically (aka *symbolic fusion*), as a means of structure analysis. As a result, table entries are never examined, and the fusion procedure never "blows up".

Propositionalization [9] is the operation of partially grounding a parfactor over a set of logical variables, thereby eliminating these variables from the resulting set. For example, the propositionalization of Z in parfactor $\phi: \ p(Y, Z), q(X, Y)$, where $dom(Z) = \{z_1, z_2\}$, results in the two parfactors, $\phi: \ p(Y, z_1), q(X, Y)$ and $\phi: \ p(Y, z_2), q(X, Y)$.

2.3 First-Order MPE

We now define the MPE task. Given a set of parfactors G, the combined weight of an assignment v is

$$w_G(v) = \prod_{g \in G} w_g(v) = \prod_{g \in G} \prod_{f \in gr(g)} w_f(v) \qquad (1)$$

The Most Probable Explanation of a first-order model is the pair $(v, \frac{1}{Z} w_G(v))$, s.t. $v = \text{argmax}_{v'} \, w_G(v')$, and Z is a normalization factor. In this paper, as in [3], we address the task of obtaining the unnormalized MPE.

3 Walk-through: Simplifying the MPE Task

3.1 Motivation

Consider the following single-parfactor model:

$$\phi: \ friends(Y, X), friends(Z, X), knows(Y, Z)$$

In this model, the chance of two people knowing each other depends on having mutual friends. Although a simple model, all existing methods fail to lift the task of computing its partition function. More specifically, inversion elimination [2] fails since no atom contains all the logical variables. Counting conversion [2] and partial inversion [3] fail since all logical variables are occupied by at least two atoms, but never by all three. Recursive conditioning (splitting on atoms) [6] fails, since all atoms contain more than a single logical variable. Finally, additional elimination and model restructuring rules [7; 1] fail to assist as well.

However, obtaining the MPE of this same model, as we will demonstrate, is polynomial in the population size. This, of course, cannot be achieved by a simple modification of the lifted methods, but rather – by exploiting a property which is unique to MPE: *uniform assignments*. To understand this property and the derived framework, we shall study three model prototypes: *single-parfactor with no recurring formulas* (namely, each atomic formula appears only once), *single-parfactor with recurring formulas* and *multiple-parfactors*.

3.1.1 Single Parfactor, No Recurring Formulas

Consider model $\boldsymbol{\phi: \ a(X), b(Y), c(Z)}$. Computing the partition function of this model is polynomial in the domain size, but finding the MPE is even easier: let entry $\phi(\rho_a, \rho_b, \rho_c)$ be the maximum entry in table ϕ. An assignment with maximal potential is obtained by assigning all grounds of atoms of $a(X), b(Y)$ and $c(Z)$ uniformly, with ρ_a, ρ_b and ρ_c, respectively. We refer to $a(X), b(Y)$ and $c(Z)$ as *uniformly assigned formulas*.

We wish to exploit the UA property in a way that may seem redundant at the moment, by applying a procedure called *uniform assignment reduction* (UAR). UAR modifies the model while preserving the MPE probability, producing a simpler MPE task. In our example $\phi: \ a(X), b(Y), c(Z)$ is modified to $\phi': \ a', b', c'$, where the arity-reduced atoms a', b', c' have the same assignment range as their original counterparts, and $\phi' = \phi^{|\{X,Y,Z\}|}$. This exponentiation compensates for the reduced number of ground factors post

modification. Once the MPE of the modified model is obtained (by a simple table lookup), the MPE assignments for $a(X), b(Y)$ and $c(Z)$ are immediately derived.

3.1.2 Single Parfactor, Recurring Formulas

We return to the friends/knows example discussed earlier, $\boldsymbol{\phi : fr(Y,X), fr(Z,X), k(Y,Z)}$ (where $fr \equiv friends$, $k \equiv knows$). Unlike the previous case, a simple table lookup cannot resolve this model's MPE, since any assignment to one ground of fr inevitably affects table entries in two table positions: $fr(Y,X)$ and $fr(Z,X)$. As a first step, we identify $\{Y,Z\}$ as the *overlap set* of the model, as Y and Z are different logical variables occupying the same position in predicate fr. Intuitively, the overlap set is the "obstruction" under which the MPE task cannot be solved by a simple table lookup. Hence, the next logical step would be to eliminate this obstacle.

Assume now that we propositionalize both Y and Z, where the domain of both is $\bigcup_i\{o_i\}$. The result of this operation is a set of parfactors, residing over the set of atoms $A = \bigcup_{i,j}\{fr(o_i,X), k(o_i,o_j)\}$. If we fuse all these parfactors together without applying a "name change" to any of the logical variables, we obtain a single parfactor over A with no recurring formulas. Thus, the previous case applies and each of $fr(o_1,X),\ldots,fr(o_n,X)$ is detected as uniformly assigned. That is, for o_1 and for *every* possible value of X, $fr(o_1,X)$ will have the same value assigned in an MPE. Similarly, for o_2 and for *every* possible value of X, $fr(o_2,X)$ will have the same value assigned in an MPE, etc. Of course, this maximizing value for $fr(o_i,X)$ and $fr(o_j,X)$ may be different, if $i \neq j$. Interestingly, we did not refer to any of the table entries in our procedure. In other words, the UA property can be detected by a set of purely symbolic operations.

We encapsulate the above sequence of operations within a new operator called *anchoring*, denoted by $\circledast$, which applies a set of symbolic propositionalizations and fusions. In our example, $\phi : fr(Y,X), fr(Z,X), k(Y,Z) \circledast \{Y,Z\} = \phi_f : fr(*,X), k(*,*)$, where $fr(*,X)$ and $k(*,*)$ are called *anchored formulas*: $fr(*,X) \equiv \bigcup_i\{fr(o_i,X)\}$ and $k(*,*) \equiv \bigcup_{i,j}\{k(o_i,o_j)\}$. We emphasize that anchoring is a symbolic operation that does not require actually carrying out explicit propositionalization and fusion but is equivalent semantically. Thus, anchoring allows us to analyze the model's structure regardless of domain size, and to simplify the depiction of uniform assignments. Namely, the anchored formula $fr(*,X)$ carries the information that $fr(o_1,X),\ldots,fr(o_n,X)$ are uniformly assigned. Essentially, this says that in the MPE, for every o_j, $fr(o_j,X)$ will have the same value for every possible substitution of X. Thus, it depends on o_j alone, and we can use $fr'(o_j)$ to denote it, instead of $fr(o_j,X)$.

To sum up thus far, we identified an overlap set, applied anchoring and obtained a set of anchored formulas. As a last step, we wish to exploit the UA property implied by the anchored formulas. We return to the UAR procedure which was introduced previously. However, now arity reductions are applied discriminately, as only non-anchored logical variables (those remaining in the anchored formulas) can be removed from the model. Hence, only X is eliminated by the UAR procedure, producing $\phi' : fr'(Y), fr'(Z), k(Y,Z)$, where $\phi' = \phi^{|X|}$. From here, the MPE is resolved by any existing inference engine.

3.1.3 Multiple Parfactors

Here, we study a model consisting of two parfactors $\boldsymbol{\phi_1 : p(X), p(Y), q(Z)}$ and $\boldsymbol{\phi_2 : p(X), r(X), q(Z)}$. As before, we wish to detect UAs and apply a UAR procedure. However, the UA detection in this case is somewhat more complicated, and involves a sequence of operations: anchoring, *alignment* and symbolic fusion. We start by identifying $\{X,Y\}$ as an overlap set in the first parfactor, followed by its anchoring to $\phi_{f_1} : p(*), q(Z)$. At this stage, the form of p is not consistent among different parfactors in the model, since the second parfactor contains a non-anchored $p(X)$. To address this inconsistent form, we must *align* the model by anchoring the X logical variable in the second parfactor. Once completed, the second parfactor is replaced with $\phi_{f_2} : p(*), r(*), q(Z)$.

The next step invokes a (symbolic) fusion of the two anchored parfactors (those consisting of ϕ_{f_1} and ϕ_{f_2}), producing $\phi_f : p(*), r(*), q(Z)$. In general, symbolic fusions may produce additional overlaps, but in this simple case no overlaps are introduced. Hence, the anchored formulas now carry the UA property. Finally, UAR is applied independently on each of the original parfactors, producing ${\phi_1}^{|Z|} : p(X), p(Y), q'$ and ${\phi_2}^{|Z|} : p(X), r(X), q'$. Having introduced the core concepts of our algorithm, we move to a formal representation.

4 Formal Definitions

4.1 Overlap Set, Anchoring and Alignment

A *position* of logical variable X in formula α is the location of X in the formula's predicate under argument list ordering. In $p(X,Y)$, the positions of X and Y are 1 and 2 respectively.

An *overlap set* L_o in parfactor g is the set of all logical variables in g which do not occupy the same positions under all instances of the same formulas.

Example 4.1. : In parfactor
$\phi : p(X,Z,Y), p(Z,U,Y), q(S,W), q(S,T), r(S)$,
the overlap set is $L_o = \{X,Z,U,W,T\}$.

The *effect scope* of a set of logical variables L_o in parfactor $g = (L,C,A,\phi)$, denoted by L_{eff}, is the set of logical

variables in g whose set of substitutions is dependent on any binding of L_o. Formally, L_{eff} is defined as the minimal set of logical variables in g under which $L_o \subseteq L_{eff}$, and $\forall \theta_1, \theta_2 \in gr(L_{eff} : C) : gr(L \setminus L_{eff} : C\theta_1) = gr(L \setminus L_{eff} : C\theta_2)$. **In constraint-less parfactors, it is always the case that** $L_{eff} = L_o$.

Example 4.2. The effect scope of $\{X, W\}$ in parfactor $\phi : p(X), q(Y, Z), r(W)$ is $\{X, W\}$.

Example 4.3. The effect scope of $\{X\}$ in parfactor $\phi : p(X), q(Y), r(Z), s(W) | C_{X \neq Y \wedge Y \neq Z}$ is $\{X, Y, Z\}$.

Anchoring of parfactor $g = (L, C, A, \phi)$ over the set of logical variables L_o, denoted by $g \circledast L_o$, is a two step procedure: propositionalization of L_{eff}, which is the effect scope of L_o, followed by a symbolic fusion of the result set of parfactors. The final result, parfactor $g' = (L', C', A', \phi')$, contains the following properties:

- $L' = L \setminus L_{eff}$
- $C' = C^{\downarrow}_{L'}$
- $A' = \{\alpha\theta \mid \alpha \in A, \theta \in gr(L_{eff} : C)\}$

And some ϕ', which we refrain from actually computing in this context. The positions of the propositionalized logical variables under anchoring are called *anchored positions*. We use the substitution $\theta = \bigcup_{X \in L_{eff}} \{X/*\}$ on the set of logical variables in g to depict the result of anchoring.

Alignment of parfactor g according to parfactor g_*, denoted by $align(g, g_*)$, is the anchoring of g over L, where L consists of all logical variables in g that occupy positions which are anchored in g_*, under predicates which are mutual to g and g_*. For instance, aligning $\phi : p(X, Y), q(Z)$ according to $\phi_* : r(*), p(W, *)$, yields $\phi' : p(X, *), q(Z)$. We define $align(G, g_*)$ as applying $align(g, g_*)$ to all $g \in G$, and then repeatedly re-aligning the new content, until all instances of each predicate are anchored consistently.

4.2 Uniform Assignments and Arity Reduction

A *uniform assignment* over formula α is the assignment of a constant ρ to every ground of α, where $\rho \in range(\alpha)$. A *partially uniform assignment* over formula α with relation to a set of logical variables L is a uniform assignment over each ground substitution of L. Namely, a partially uniform assignment over $t(X, Y, Z)$ w.r.t. $\{Y, Z\}$ implies that $\forall y \in dom(Y), \forall z \in dom(Z) : t(X, y, z)$ is uniformly assigned.

To enable the application of uniform assignments on our model, we define the *arity reduction* operator $\alpha \downarrow \alpha_*$ on atomic formulas. If α and α_* have the same predicate symbol, $\alpha \downarrow \alpha_*$ is a new predicate obtained by removing all positions in α not anchored in α_*. For instance, $p(X, Y, Z) \downarrow p(W, *, T) = p'(Y)$. If both formulas differ in the predicate symbol, $\alpha \downarrow \alpha_* = \alpha$.

4.3 Uniform Assignment Reduction

A *Uniform Assignment Reduction* (*UAR*) applied to parfactor $g = (L, C, A, \phi)$ by formula α_*, denoted by $g \downarrow \alpha_*$, is the operation of applying $\alpha \downarrow \alpha_*$ to each $\alpha \in A$, followed by exponentiating the potential by the combined domain size of the removed logical variables. More formally, the operation $g \downarrow \alpha_*$ yields $g' = (L', C', A', \phi')$ with the following properties:

- $A' = \{\alpha \downarrow \alpha_* \mid \alpha \in A\}$
- $C' = C^{\downarrow}_{L'}$
- $L' = LV(A')$
- $\phi' = \phi^{|(L \setminus L') : C\theta'|}$

Where θ' is some ground substitution over L' under constraint C.

Example 4.4. Let $g = \phi : p(X), q(Y, Z)$.
Let $\alpha_* = q(W, *)$. Hence, $g \downarrow \alpha_* = \phi^{|Y|} : p(X), q'(Z)$.

Example 4.5. Let $g = \phi : p(X), q(Y) \mid C_{X \neq Y}$.
Let $\alpha_* = p(Z)$. Hence, $g \downarrow \alpha_* = \phi^{|X|-1} : p', q(Y)$,
since $|X : C_{X \neq Y}| = |X| - 1$ under any binding of Y.

Example 4.6. When the set of the logical variables remains the same, as in $\phi : p(Y), q(Y, Z) \downarrow q(W, *) = \phi : p(Y), q'(Z)$, no exponentiation is required.

5 Simplified First-Order MPE

The model simplification algorithm is comprised of two phases – detecting uniform assignments, and applying a uniform assignment reduction, as follows.

Algorithm 1: DETECT-UNIFORM-ASSIGNMENTS

input : G_{input} – a completely shattered model
output: A set of anchored formulas

$G \leftarrow G_{input}$
while $\exists g \in G, \exists L_o \neq \emptyset$ *s.t.* L_o *is an overlap set in* g **do**
  $g_* \leftarrow g \circledast L_o$ `// anchoring`
  $G \leftarrow align(G, g_*)$ `// alignment`

if $\exists g_1 \neq g_2 \in G$ *s.t.* $rv(g_1) \cap rv(g_2) \neq \emptyset$ **then**
  $G \leftarrow G \cup$ `SymbolicFusion(`g_1, g_2`)` $\setminus \{g_1, g_2\}$
  return `DETECT-UNIFORM-ASSIGNMENTS(`G`)`
else
  return all formulas in G

5.1 Detecting Uniform Assignments

The detection procedure (Algorithm 1) is purely symbolic, hence only the set of formulas is examined, whereas table entries are completely ignored. The detection starts by identifying all overlap sets in the model, accommodated by anchoring. The result is a set of parfactors with no recurring formulas. The entire model is then aligned accord-

Algorithm 2: SIMPLIFY-FOMPE

input : G – a completely shattered model
output: Simplified Model

$A_* \leftarrow$ DETECT-UNIFORM-ASSIGNMENTS(G)
foreach $\alpha_* \in A_*$ **do**
  foreach $g \in G$ **do** $G \leftarrow G \cup \{g \downarrow \alpha_*\} \setminus \{g\}$
return G

ingly, making sure that following manipulations operate on a completely shattered model. Next, two parfactors are chosen for a symbolic fusion. This, in turn, may generate a new overlap set. The cycle of overlap set detection, anchoring, and symbolic fusion is repeated until no two parfactors which share a formula are left. The procedure returns a set of anchored formulas, depicting the set of detected UAs.

One issue which remains unsolved is how to reapply the symbolic fusion procedure in order to produce as many arity reductions as possible. We use a greedy fusion scheme, where the logical variables of formulas with maximum arity are matched first. This, of course, does not guarantee an optimal sequence of fusions, and we leave the study of better fusion selection strategies for future work.

Proposition 1. *Given a completely shattered model G, Algorithm 1 produces a set of anchored formulas A_*, where there exists an MPE solution under which all formulas $\alpha_* \in A_*$ are partially uniformly assigned w.r.t. the logical variables in the anchored positions.*

Proof outline. The algorithm applies (symbolically) standard lifted inference tools, where each fusion is followed by anchoring, namely – the elimination of overlapping logical variables. Once all parfactors are merged into a single parfactor with no overlap sets, an MPE solution can be trivially obtained by assigning the atomic formulas uniformly, according to remaining logical variables. □

5.2 UAR Algorithm

After invoking DETECT-UNIFORM-ASSIGNMENTS, which detects uniform assignments in model G, a uniform assignment reduction is applied to each parfactor in G, producing a simplified model G'.

Proposition 2. *Algorithm 2 produces a simplified model G' with the same MPE probability as G.*

Proof outline. The formal proof is found in the Appendix. The idea is to pair each original parfactor $g \in G$ with its UAR result $g' \in G'$, and show the following: (i) for each assignment v' in model G' there exists an assignment v in model G, where the probabilistic weights of all pairs g and g' are equal; and (ii) for each optimal assignment v in G, under which formulas are uniformly assigned according to A_*, there exists an assignment v' s.t. the probabilistic weights of all pairs g and g' are equal. A_* here denotes the set of anchored formulas which yielded G'. □

Original model	$\phi_1 : p(X,Y), q(Y,Z), r(Z,X), s(X,Z)$ $\phi_2 : s(X, \underbrace{V}_{overlap}), s(X, \underbrace{W}_{overlap}), p(X,Y)$
Anchoring	$\phi_1 : p(X,Y), q(Y,Z), r(Z,X), s(X,Z)$ $\phi_2' : s(X,*), p(X,Y)$
Model alignment	$\phi_1' : p(X,Y), q(Y,*), r(*,X), s(X,*)$ $\phi_2' : s(X,*), p(X,Y)$
Symbolic fusion	$\phi_f : p(X,Y), q(Y,*), r(*,X), s(X,*)$
UA reduction	${\phi_1}^{\lvert X\rvert\cdot\lvert Y\rvert} : p', q'(Z), r'(Z), s'(Z)$ ${\phi_2}^{\lvert X\rvert\cdot\lvert Y\rvert} : s'(V), s'(W), p'$

Figure 1: Example – model simplification

5.2.1 UAR Example

We now cover the example in Figure 1. Let model G consist of $\phi_1 : p(X,Y), q(Y,Z), r(Z,X), s(X,Z)$ and $\phi_2 : s(X,V), s(X,W), p(X,Y)$. We start by detecting the overlap set {V,W} in ϕ_2, followed by a subsequent anchoring of ϕ_2 to $\phi_2' : s(X,*), p(X,Y)$, and the alignment of ϕ_1 to $\phi_1' : p(X,Y), q(Y,*), r(*,X), s(X,*)$. Next, a symbolic fusion is applied to ϕ_1' and ϕ_2', resulting in $\phi_f : p(X,Y), q(Y,*), r(*,X), s(X,*)$. The remaining formulas in ϕ_f are then passed to the SIMPLIFY-FOMPE procedure, where consecutive uniform assignment reductions yield ${\phi_1}^{\lvert X\rvert\cdot\lvert Y\rvert} : p', q'(Z), r'(Z), s'(Z)$ and ${\phi_2}^{\lvert X\rvert\cdot\lvert Y\rvert} : s'(V), s'(W), p'$.

5.3 Complexity Analysis

Let t, n, a and p denote the properties of the original model G, where t is the number of table entries, n is the number of atoms, a is the maximum arity of any predicate and p is the number of parfactors. The complexity of SIMPLIFY-FOMPE is simply $O(t)$. As for DETECT-UNIFORM-ASSIGNMENTS, each position in any atom of G is aligned no more than once, thus $O(n \cdot a)$ is the upper bound for the number of anchored positions. $p - 1$ is the maximal number of symbolic fusions. Combined, the total complexity is $O(t) + O(n \cdot a) + O(p \cdot O_f)$, where O_f is the upper bound of a symbolic fusion and depends on the fusion strategy. For example, a greedy fusion strategy takes $O(n \cdot a)$ time. Other polynomial time strategies are possible. In typical models, n and t are the dominating elements, and so we get a complexity that is linear in t and polynomial in n. All in all, the complexity is independent of domain sizes.

6 UA under Recursive Conditioning

As demonstrated, the UA property is heavily dependent on the structure of the model. One interesting aspect of recur-

sive conditioning is the structure modification induced by the conditioning operator (splitting on a singleton atom). This allows non UA formulas to be split into uniformly assigned groups, a property which is referred to as a *conditional UA*. Consequently, the scope of uniform assignments is extended beyond the role of preprocessing, as presented so far, since various conditioning contexts introduce further opportunities for exploiting the UA property.

Consider model $\phi_1 : p(X), q(X)$ and $\phi_2 : p(X), q(Y)$, where p and q are boolean, and the domain size of X and Y is n. Since all possible fusions of ϕ_1 and ϕ_2 result in an overlap (e.g. $\phi_f : p(X), q(X), q(Y)$), this model contains no uniform assignments. However, when conditioning on atom $p(X)$, the ground atoms of $q(X)$ split into two uniformly assigned groups. We demonstrate this as follows: let $k \in \{0, ..., n\}$ depict the number of $p(X)$ ground atoms assigned 0. Hence, under each binding of k, the domain is split into two parts, and a set of four logical variables is introduced: X_0^k, X_1^k, Y_0^k and Y_1^k, where $|X_0^k| = k$, $|X_1^k| = n - k$ and $dom(X_i^k) = dom(Y_i^k)$. By definition, each ground of $p(X_i^k)$ is assigned i, thus all random variables of predicate p are eliminated from the expression.

Formally:

$$\max_{rv(p(X),q(X))} \prod_X \phi_1(p(X), q(X)) \cdot \prod_{X,Y} \phi_2(p(X), q(Y)) =$$

$$\max_k \max_{rv(q(X))} \prod_i \prod_{X_i^k} \underbrace{\phi_1(i, q(X_i^k))}_{\equiv \phi_i^X(q(X_i^k))} \cdot$$

$$\prod_i \prod_{Y_i^k} \underbrace{\phi_2(0, q(Y_i^k))^k \phi_2(1, q(Y_i^k))^{n-k}}_{\equiv \phi_i^Y(q(Y_i^k))}$$

The result is a set of four parfactors, as follows:
(i) $\phi_0^X : q(X_0^k)$ (ii) $\phi_1^X : q(X_1^k)$
(iii) $\phi_0^Y : q(Y_0^k)$ (iv) $\phi_1^Y : q(Y_1^k)$

In this formation, it is easy to detect that both $q(X_0^k)$ and $q(X_1^k)$ are uniformly assigned. Namely, the set of parfactors can be simplified into
(i) $(\phi_0^X)^k : q_0'$ (ii) $(\phi_1^X)^{n-k} : q_1'$
(iii) $(\phi_0^Y)^k : q_0'$ (iv) $(\phi_1^Y)^{n-k} : q_1'$

We note that in this basic example, MPE can be efficiently solved without resorting to UAR. However, the same conditional UA principle applies to any model, thus the symbolic simplification becomes quite efficient when considering that the same computation is repeated for each k. Embedding UAR in a recursive conditioning inference engine, may indeed accelerate many of the lifted MPE tasks.

7 Experimental Evaluation

We present six sets of experiments. To highlight the fact that the method is independent of the inference engine used, and effective with different engines, we used two lifted inference engines, C-FOVE [9] and WFOMC [4], with and without the uniform assignment reduction. The engines were obtained from `http://people.csail.mit.edu/milch/blog/index.html` and `dtai.cs.kuleuven.be/ml/systems/wfomc` respectively, and were both modified to support max-product queries, by replacing Sum operators with Max, and ignoring the number of counting permutations. In WFOMC, specifically, the modification corresponds to changing the code in each of the NNF node subclasses. Times are shown in log scale, and the computation time of UAR was added to the overall time results.

Since WFOMC accepts weighted first-order formulas as input, each parfactor $\phi : p_1(X_1), \ldots, p_n(X_n)$ was passed to WFOMC as a weighted first-order formula of the form $p_1(X_1) \wedge \ldots \wedge p_{n-1}(X_{n-1}) \implies p_n(X_n)$. Unlike variants of belief propagation [14; 8], where table entries affect the runtime of the algorithm, the runtime of both WFOMC and C-FOVE in models which exclusively consist of non-zero and non-one entries is agnostic to the actual numerical values. Hence, all table entries in the models presented here were arbitrarily set to non-zero, non-one.

Figures 2 & 3 show the results of two models that were previously discussed in this paper, demonstrating how UAR extends the scope of known tractable models. In these two figures, we see that both C-FOVE and WFOMC quickly fail to lift the inference task, as they resort to propositional inference. Once UAR is applied, both solve the MPE query efficiently. Figure 4 introduces a model which is solvable in polynomial time, and again, becomes much easier to solve with UAR. The $\downarrow$ symbol denotes a logical variable which was eliminated by the UAR.

In all three figures, UAR computation never exceeds 40 milliseconds. This result is, of course, consistent with the complexity analysis, demonstrating that UA detection has a negligible effect on the overall performance. Given that MPE is computationally demanding, and exponential in the worst-case, one could hardly lose from attempting to run UAR in every MPE task.

Figure 5 presents the results of running WFOMC with and without UAR, on random models with varying number of parfactors. Each parfactor, in an n-parfactor randomized model, consists of 3 atoms. Each atom is obtained randomly from a uniformly distributed pool of $2n$ unary predicates, and a set of (maximum) 3 logical variables per parfactor. This randomization guarantees a (polynomially) tractable model, that can evaluate the effectiveness of UAR. However, even with small domain sizes, solving MPE is still computationally demanding. We tested these random models under three domain sizes: 5, 10 and 50. The results are an average over 10 randomized sets.

Figure 6 shows an unsuccessful attempt by UAR to simplify the Friends & Smokers model [14]. Since UAR com-

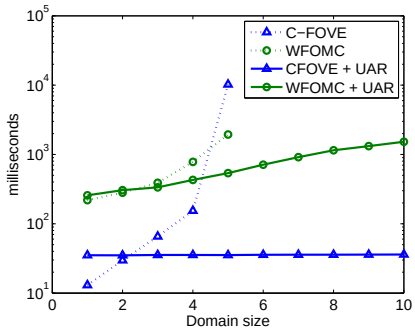


Figure 2: $\phi: friends(X_\downarrow, Y), friends(X_\downarrow, Z), knows(Y, Z)$

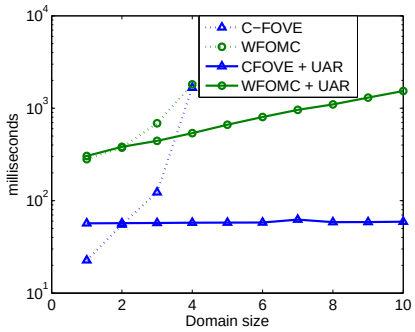


Figure 3: $\phi_1: p(X_\downarrow, Y_\downarrow), q(Y_\downarrow, Z), r(Z, X_\downarrow), s(X_\downarrow, Z)$
$\phi_2: s(X_\downarrow, V), s(X_\downarrow, W), p(X_\downarrow, Y_\downarrow)$

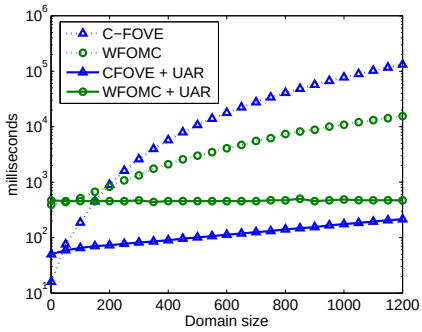


Figure 4: $\phi_1: p(X), q(X), r(Z_\downarrow)$
$\phi_2: p(X), q(Y), r(Z_\downarrow)$

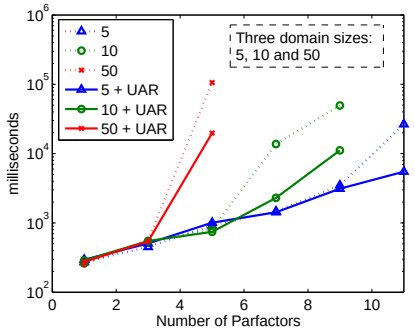


Figure 5: WFOMC on random models

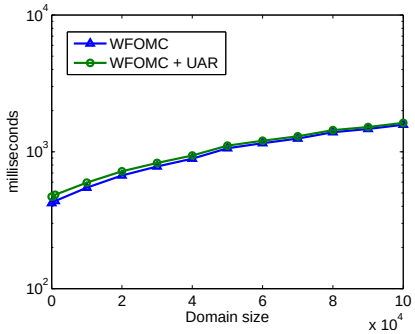


Figure 6: Friends & Smokers [14]

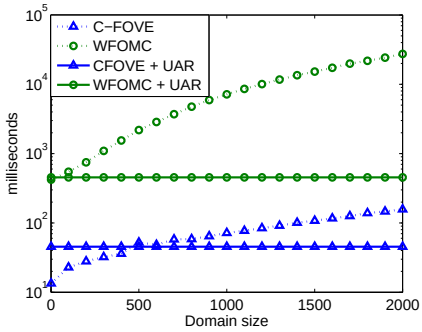


Figure 7: Students & Professors [6]

putation incurs very little overhead (48 milliseconds in this case), its effect on the result is negligible. In Figure 7, however, UAR is able to simplify the Students & Professors link prediction model [6], s.t. the computation becomes domain-size independent.

8 Conclusion

We introduced a model simplification method which narrows the assignment space of first-order MPE queries by batching together sets of random variables, while refraining from table lookups. The suggested method can be integrated into existing lifted inference engines or serve as a preprocessing algorithm, adding a low computational overhead compared to non-simplified queries. And thus, even in cases where UAR fails, the penalty for running this procedure is negligible.

We note that the UA property is not restricted to the specific detection method presented in this paper. For example, although we refrained from analyzing the content of the potential tables when detecting uniform assignments, one can leverage specific maximizing assignments to obtain farther reductions. Consider, for example, an atom p contained exclusively by parfactor $\phi : p(Y), p(Z)$. The overlap set of the two instances of p is not empty, and in general requires anchoring and alignment. However, if maximal entries in ϕ assign both instances of p the same value (e.g., they are $(0, 0)$ or $(1, 1)$, etc.) then p can be treated as uniformly assigned. Namely, the parfactor is simplified into $\phi^{|Y|\cdot|Z|} : p'$, by applying UAR. This idea can be further adapted by approximation algorithms, which may force uniform assignments in cases where they are not guaranteed. Finally, third parties may also wish to assert uniform assignments as part of user constraints. For all these purposes, the UAR serves as a valid reduction.

Acknowledgments

The authors are partially supported by ISF grant 8254320, the Paul Ivanier Center for Robotics Research and Production Management, and the Lynn and William Frankel Center for Computer Science.

Appendix – UAR Proof

Let G be a completely shattered model, and let $G' = \bigcup_{g \in G} \{g \downarrow \alpha_*\}$ be the set of post-UAR parfactors, for some anchored formula α_*. WLOG, properties of each parfactor g in G are (L, C, A, ϕ), where $A = \{\alpha_1, \ldots, \alpha_n, \beta\}$, and $\forall i : rv(\alpha_i) = rv(\alpha_*) \wedge rv(\alpha_i) \cap rv(\beta) = \emptyset$.

Properties of each $g' = g \downarrow \alpha_* = (L', C', A', \phi')$ are

- $A' = \{\alpha'_1, \ldots, \alpha'_n, \beta\}$, where $\forall i : \alpha'_i = \alpha_i \downarrow \alpha_*$
- $L' = LV(A')$
- $C' = C^{\downarrow}_{L'}$
- $\phi' = \phi^{|(L \setminus L'):C\theta'|}$

Where θ' is some ground substitution over L' under constraint C.

In the context of parfactor g, let L^*_i denote the set of logical variables in α_i that occupy positions which are anchored in α_*, and let L_β depict the set of logical variables in β. We define $L^+ = L' = L^*_1 \cup \ldots \cup L^*_n \cup L_\beta$, followed by $L^- = L \setminus L^+$ which is the set of logical variables which occupy positions that are unanchored in α_* and contained exclusively by some α_i instances. Note that $gr(L^+ : C) = gr(L' : C')$, since $L^+ = L'$ and $C' = C^{\downarrow}_{L'}$.

Weights of each g and g', under assignments v and v', are:

$$w_g(v) = \prod_{\theta^+} \prod_{\theta^-} \phi(\alpha_1\theta^+\theta^-{}_{|v}, \ \ldots\ , \alpha_n\theta^+\theta^-{}_{|v}, \beta\theta^+\theta^-{}_{|v})$$

$$w_{g'}(v') = \prod_{\theta'} \phi(\alpha'_1\theta'{}_{|v'}, \ \ldots\ , \alpha'_n\theta'{}_{|v'}, \beta\theta'{}_{|v'})^{|L \setminus L':C\theta'|}$$

Where $\prod_{\theta^+}$, $\prod_{\theta^-}$ and $\prod_{\theta'}$ are abbreviations for $\prod_{\theta^+ \in gr(L^+:C)}$, $\prod_{\theta^- \in gr(L^-:C\theta^+)}$ and $\prod_{\theta' \in gr(L':C')}$.

Proposition 3. *For each v' there exists v such that $w_G(v) = w_{G'}(v')$*

Proof. We pick one of the α_i instances in some parfactor g, and construct v out of v' as follows:

1. $\forall \theta^* \in gr(L^*_i : C),\ \forall \theta \in gr(LV(\alpha_i) \setminus L^*_i : C\theta^*) : \alpha_i\theta^*\theta_{|v} = \alpha'_i\theta^*{}_{|v'}$
2. $\psi \in rv(G) \wedge \psi \in rv(G') \implies \psi_{|v} = \psi_{|v'}$
   Meaning, the assignment to variables which are mutual to G' and G is copied from v' to v.

The weights of each g and counterpart g' are then expressed as follows

$$\begin{aligned}
w_g(v) &= \prod_{\theta^+} \prod_{\theta^-} \phi(\alpha_1\theta^+\theta^-{}_{|v}, \ \ldots\ , \alpha_n\theta^+\theta^-{}_{|v}, \beta\theta^+\theta^-{}_{|v}) \\
&= \prod_{\theta^+} \prod_{\theta^-} \phi(\alpha_1\theta^+\theta^-{}_{|v'}, \ \ldots\ , \alpha_n\theta^+\theta^-{}_{|v'}, \beta\theta^+\theta^-{}_{|v'}) \\
&= \prod_{\theta^+} \prod_{\theta^-} \phi(\alpha'_1\theta^+{}_{|v'}, \ \ldots\ , \alpha'_n\theta^+{}_{|v'}, \beta\theta^+{}_{|v'}) \\
&= \prod_{\theta'} \phi(\alpha'_1\theta'{}_{|v'}, \ \ldots\ , \alpha'_n\theta'{}_{|v'}, \beta\theta'{}_{|v'})^{|L \setminus L':C\theta'|} \\
&= w_{g'}(v')
\end{aligned}$$

Since the full weights are a product over all parfactor weights, it follows that $w_G(v) = w_{G'}(v')$. □

Proposition 4. *Given optimal assignment $v = \text{argmax}_v\, w_G(v)$, under which α_* is uniformly assigned, there exists v' for which $w_{G'}(v') = w_G(v)$.*

Proof. First, we pick one of the α'_i instances in some parfactor g'. From the uniform assignment property depicted by α_*, it follows that $\forall \theta^* \in gr(L^*_i : C),\ \forall \theta_1, \theta_2 \in gr(LV(\alpha_i) \setminus L^*_i : C\theta^*) : \alpha_i\theta^*\theta_1(v) = \alpha_i\theta^*\theta_2(v)$. Next, we construct v' out of v as follows:

1. $\forall \theta^* \in gr(L^*_i : C),\ \forall \theta \in gr(LV(\alpha_i) \setminus L^*_i : C\theta^*) : \alpha'_i\theta^*{}_{|v'} = \alpha_i\theta^*\theta_{|v}$
   Enabled by the uniform assignment property.
2. $\psi \in rv(G) \wedge \psi \in rv(G') \implies \psi_{|v'} = \psi_{|v}$

As previously, weights of each g' and g are expressed as a function of ground substitutions, where $w_{g'}(v') = w_{g'}(v)$, and consequently $w_{G'}(v') = w_G(v) = \max_v w_G(v)$. □

From propositions 3 & 4 it follows that $max_{v'} w_{G'}(v') = max_v w_G(v)$, hence the correctness of UAR.

References

[1] U. Apsel and R. I. Brafman. Extended lifted inference with joint formulas. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pages 11–18. AUAI Press, 2011.

[2] R. de Salvo Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *IJCAI*, pages 1319–1325, 2005.

[3] R. de Salvo Braz, E. Amir, and D. Roth. Mpe and partial inversion in lifted probabilistic variable elimination. In *AAAI*. AAAI Press, 2006.

[4] G. V. den Broeck, N. Taghipour, W. Meert, J. Davis, and L. D. Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, pages 2178–2185, 2011.

[5] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[6] V. Gogate and P. Domingos. Probabilistic theorem proving. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pages 256–265. AUAI Press, 2011.

[7] A. Jha, V. Gogate, A. Meliou, and D. Suciu. Lifted inference seen from the other side : The tractable features. In *Advances in Neural Information Processing Systems 23*. 2010.

[8] K. Kersting, Y. E. Massaoudi, F. Hadiji, and B. Ahmadi. Informed lifting for message-passing. In *AAAI*, 2010.

[9] B. Milch, L. S. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling. Lifted probabilistic inference with counting formulas. In *AAAI*, pages 1062–1068, 2008.

[10] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[11] D. Poole. First-order probabilistic inference. In G. Gottlob and T. Walsh, editors, *IJCAI*, pages 985–991. Morgan Kaufmann, 2003.

[12] D. Poole, F. Bacchus, and J. Kisynski. Towards completely lifted search-based probabilistic inference. *CoRR*, abs/1107.4035, 2011.

[13] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[14] P. Singla and P. Domingos. Lifted first-order belief propagation. In *AAAI*, pages 1094–1099, 2008.

Plackett-Luce regression: A new Bayesian model for polychotomous data

Cédric Archambeau
Xerox Research Center Europe
Meylan, France
cedric.archambeau@xrce.xerox.com

François Caron
INRIA
Univ. Bordeaux, IMB, UMR 5251
Talence, France
francois.caron@inria.fr

Abstract

Multinomial logistic regression is one of the most popular models for modelling the effect of explanatory variables on a subject choice between a set of specified options. This model has found numerous applications in machine learning, psychology or economy. Bayesian inference in this model is non trivial and requires, either to resort to a Metropolis-Hastings algorithm, or rejection sampling within a Gibbs sampler. In this paper, we propose an alternative model to multinomial logistic regression. The model builds on the Plackett-Luce model, a popular model for multiple comparisons. We show that the introduction of a suitable set of auxiliary variables leads to an Expectation-Maximization algorithm to find Maximum A Posteriori estimates of the parameters. We further provide a full Bayesian treatment by deriving a Gibbs sampler, which only requires to sample from highly standard distributions. We also propose a variational approximate inference scheme. All are very simple to implement. One property of our Plackett-Luce regression model is that it learns a sparse set of feature weights. We compare our method to sparse Bayesian multinomial logistic regression and show that it is competitive, especially in presence of polychotomous data.

1 Introduction

The multinomial logistic regression (or multinomial logit) model is one of the most popular models for modelling the effect of explanatory variables $X_i = (X_{i1}, \ldots, X_{id}) \in \mathcal{X}$ on a subject choice Y_i between a set of prespecified options $\{1, \ldots, K\}$. The model, which belongs to the class of generalized linear models (McCullagh & Nelder, 1989), takes the following form:

$$\Pr(Y_i = k | X_i, \beta) = \frac{e^{X_i^\top \beta_k}}{1 + \sum_{\ell=1}^{K-1} e^{X_i^\top \beta_\ell}} \qquad (1)$$

for $k = 1, \ldots, K-1$ and $\Pr(Y_i = K | X_i) = \frac{1}{1+\sum_{\ell=1}^{K-1} \exp(X_i^\top \beta_\ell)}$. The monograph of Agresti (1990) gives details on the foundations of this model and related ones such as conditional logit (McFadden, 1973). The parameters $\beta_k = (\beta_{k1}, \ldots, \beta_{kd})$, $k = 1, \ldots, K-1$ are unknown and have to be estimated from the data. Bayesian inference in multinomial logit is complicated by the fact that no conjugate prior exists for the regression parameters. Various algorithms have been proposed in the literature, see e.g. (Dey et al., 2000). In particular, similarly to Bayesian inference in probit models (Albert & Chib, 1993; Talhouk et al., 2012; Holmes & Held, 2006) proposed an auxiliary variable model for block Gibbs sampling. Alternative Markov Chain Monte Carlo (MCMC) algorithms have been compared by Girolami & Calderhead (2011). However, although some algorithms show excellent mixing properties, they lack simplicity of implementation.

When only some of the predictors are assumed to be relevant, sparse multinomial logistic regression has been proposed using Laplace priors.Krishnapuram et al. (2005) obtained Maximum A Posteriori (MAP) estimates via Minorization-Maximization-based algorithms, while fully Bayesian inference is conducted in (Cawley et al., 2007) and (Genkin et al., 2007).

In this article, we propose an alternative model to multinomial logit for multi-class classification and discrete choice modelling. We show that the use of a carefully chosen set of latent variables leads to an Expectation Maximization (EM) algorithm to find MAP estimates, a Gibbs sampler or a variational EM algorithm to approximate the full posterior. Importantly, the Gibbs sampler only requires to sample from standard distributions, which makes it computationally highly amenable. Moreover, for some values of the hyperpa-

rameter, we show that the model induces sparsity over the parameters. In particular, the MAP estimates are exactly sparse. We also provide detailed comparaisons with Bayesian sparse multinomial logit in terms of computational efficiency and prediction performances. The Bayesian treatment of sparse multinomial logistic regression is discussed in Appendix A. The model builds on the data augmentation model proposed by Holmes & Held (2006) and the Bayesian lasso of Park & Casella (2008). To the best of our knowledge is has not been proposed before.

2 Statistical model

We propose an alternative model for categorical data analysis. The model is defined as follows:

$$\Pr(Y_i = k|X_i, \lambda) = \frac{W_i^\top \lambda_k}{\sum_{\ell=1}^K W_i^\top \lambda_\ell} \tag{2}$$

for $k = 1, \ldots, K$, where $W_i = (W_{i1}, \ldots, W_{ip})$ and $W_{ij} = \mathcal{K}_j(X)$. $\mathcal{K}_j$ is a known function from $\mathcal{X}$ to $[0, +\infty[$. In practice, there is a lot of flexibility on the choice of the transformations $\mathcal{K}_j$ of the explanatory variables. In the remaining of this paper, we will consider $\mathcal{K}_j(X) = \exp(X_{ij})$, $\mathcal{K}_{d+j}(X) = \exp(-X_{ij})$ to account for negative effects and $\mathcal{K}_{2d+1}(X) = \exp(0)$ for the offset. The non-negative parameters λ_{kj}, $k = 1, \ldots, K$, $j = 1, \ldots, p$ are unknown and have to be estimated from the data.

The model shares strong similarities with the Plackett-Luce model for multiple comparisons (Luce, 1959; Plackett, 1975), for which efficient Bayesian inference can be carried out (Guiver & Snelson, 2009; Caron & Doucet, 2012). Similarly to this model, it also admits a Thurstonian interpretation (Diaconis, 1988) that we describe now. For $k = 1, \ldots, K$ and $j = 1, \ldots, p$, let

$$V_{ikj} \sim \text{Exp}(W_{ij}\lambda_{kj}) \ ,$$

where $\text{Exp}(b)$ denotes the exponential distribution with rate parameter b. As an analogy, let consider a race between K different teams, each team having p competitors. Each individual j in a team k has an arrival time V_{ikj} for competition i. Then Y_i denotes the winning team, i.e. the team k which has the individual with the lowest arrival time. Following properties of the exponential distribution, it is straightforward to show that

$$\Pr(\arg\min_\kappa(\min_j V_{i\kappa j}) = k|X_i, \lambda) = \frac{W_i^\top \lambda_k}{\sum_{\ell=1}^K W_i^\top \lambda_\ell} \ .$$

The decision boundaries between two classes k and ℓ are given by

$$W^\top(\lambda_k - \lambda_\ell) = 0 \ ,$$

i.e. they are linear in the transformed domain W. As the multinomial logit model, the model (2) satisfies Luce's axiom of choice (Luce, 1977) a.k.a. *independence from irrelevant alternatives*.

3 MAP estimation and Bayesian Inference

3.1 Data Augmentation

The likelihood defined by (2) does not admit a conjugate prior. We can nonetheless consider a data augmentation scheme as indicated by Caron & Doucet (2012). Assume that we have observed data $\mathcal{Y} = \{Y_i\}_{i=1}^n$. We introduce auxiliary latent variables $\mathcal{C} = \{C_i\}_{i=1}^n$ and $\mathcal{Z} = \{Z_i\}_{i=1}^n$ such that, for $i = 1, \ldots, n$

$$Y_i|C_i, \lambda \sim \text{Disc}\left(\frac{\lambda_{1C_i}}{\sum_\ell \lambda_{\ell C_i}}, \ldots, \frac{\lambda_{KC_i}}{\sum_\ell \lambda_{\ell C_i}}\right), \tag{3}$$

$$C_i|\lambda \sim \text{Disc}\left(\frac{W_{i1}\sum_k \lambda_{k1}}{W_i^\top \sum_\ell \lambda_\ell}, \ldots, \frac{W_{ip}\sum_k \lambda_{kp}}{W_i^\top \sum_\ell \lambda_\ell}\right), \tag{4}$$

$$Z_i|\lambda \sim \text{Exp}\left(W_i^\top \sum_\ell \lambda_\ell\right), \tag{5}$$

where $\text{Disc}(\pi_1, \ldots, \pi_p)$ denotes the discrete distribution of parameters $(\pi_1, \ldots, \pi_p)$ where $\pi_i \geq 0$ and $\sum_i \pi_i = 1$. Pursuing the analogy with team competition introduced in the previous section, the latent variables Z_i and C_i have the following interpretation. $Z_i = \min_{k,j}(V_{ijk})$ corresponds to the arrival time of the winner of the competition. As the variables V_{ijk} are exponentially distributed and the minimum of two exponential variable of rates w_1 and w_2 is an exponential variables of rate $w_1 + w_2$, we recover (5). $C_i \in \{1, \ldots, p\}$ corresponds to the index of the individual in the winning team who arrives first.

The model (3–4) corresponds to an ad-mixture of Discrete distributions. Indeed, the class probabilities defined by the Plackett-Luce model (2) can be rewritten in the form of a mixture model by integrating out the latent indicator variables $\{C_i\}_{i=1}^n$:

$$\Pr(Y_i = k|X_i, \lambda) = \sum_j \pi_{ij} \text{ Disc}\left(\frac{\lambda_{1j}}{\sum_\ell \lambda_{\ell j}}, \ldots, \frac{\lambda_{Kj}}{\sum_\ell \lambda_{\ell j}}\right),$$

where the mixture weight $\pi_{ij} = \frac{W_{ij}\sum_k \lambda_{kj}}{W_i^\top \sum_\ell \lambda_\ell}$ depends on the data through $\{W_{ij}\}_{j=1}^p$. Note that there is one mixture component per feature. Hence, each component characterises the classes by assigning a different importance weight to their associated feature.

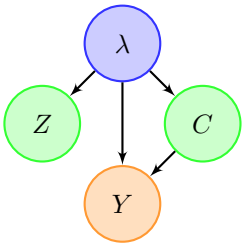


Figure 1: Graphical Model. Parameter of interest λ is in blue, latent variables Z and C in green and measurements Y in orange.

The resulting log-complete likelihood is given by

$$\ln p(\mathcal{Y}, \mathcal{C}, \mathcal{Z}|\lambda) = \sum_{k=1}^{K}\sum_{j=1}^{p}\left\{ n_{kj} \ln \lambda_{kj} - \lambda_{kj} \sum_{i=1}^{n} Z_i W_{ij} \right\} + \sum_{i=1}^{n}\sum_{j=1}^{p} \delta_{C_i j} \ln W_{ij} \ , \tag{6}$$

where $n_{kj} = \sum_i \delta_{Y_i k}\delta_{C_i j}$ and δ_{ij} is the Kronecker delta. We further assign a conjugate Gamma prior to the parameters λ:

$$\lambda \sim \prod_{k=1}^{K}\prod_{j=1}^{p} \mathrm{Gam}(\lambda_{kj}; a, b) \ . \tag{7}$$

where $a > 0$ is the shape parameter and $b > 0$ is the rate parameter. The graphical model is shown in Figure 1.

3.2 EM algorithm

The log-posterior can be maximized using the EM algorithm by iteratively maximizing a lower bound, called the negative *variational free energy* (Neal & Hinton, 1998), or a penalized version as below:

$$\begin{aligned} \ln p(\lambda|\mathcal{Y}) &\geq \langle \ln p(\mathcal{Y}, \mathcal{C}, \mathcal{Z}|\lambda) \rangle_q + \mathrm{H}\left(q(\mathcal{C}, \mathcal{Z})\right) + \ln p(\lambda) \\ &\equiv -\mathcal{F}(q, \lambda) + \ln p(\lambda) \ , \end{aligned} \tag{8}$$

where $\langle \cdot \rangle_q$ denotes the expectation with respect to $q(\mathcal{C}, \mathcal{Z})$ and $\mathrm{H}(\cdot)$ is the (differential) entropy. When the posterior $p(\mathcal{C}, \mathcal{Z}|\mathcal{Y}, \lambda)$ is analytically tractable, then we set $q(\mathcal{C}, \mathcal{Z}) = p(\mathcal{C}, \mathcal{Z}|\mathcal{Y}, \lambda)$ so that the bound is exact and we recover the EM algorithm.

Given the augmented model (3-5), the posterior factorizes: $p(\mathcal{C}, \mathcal{Z}|\mathcal{Y}, \lambda) = \prod_i p(C_i|Y_i, \lambda) p(Z_i|\mathcal{Y}, \lambda)$. The E-step consists in computing the posterior of the latent indicator variables for fixed λ.[1] This leads to

$$C_i|Y_i, \lambda \sim \mathrm{Disc}\left(\frac{W_{i1}\lambda_{Y_i 1}}{W_i^\top \lambda_{Y_i}}, \ldots, \frac{W_{ip}\lambda_{Y_i p}}{W_i^\top \lambda_{Y_i}} \right). \tag{9}$$

[1] Note that $p(Z_i|\mathcal{Y}, \lambda)$ is given by (5)

The M-step updates the parameter set λ for fixed posteriors:

$$\lambda \leftarrow \arg\max_{\lambda} \langle \ln p(\mathcal{Y}, \mathcal{C}|\lambda) \rangle_{p(\mathcal{C}|\mathcal{Y}, \lambda)} + \ln p(\lambda) \ .$$

It follows that

$$\lambda_{kj} = \begin{cases} \frac{a - 1 + \langle n_{kj} \rangle}{b + \sum_i \langle z_i \rangle W_{ij}} & \text{if } a > 1 - \langle n_{kj} \rangle \ , \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

where $\langle n_{kj} \rangle = \sum_i \frac{W_{ij}\lambda_{kj}}{W_i^\top \lambda_k}$ and $\langle z_i \rangle = \left(W_i^\top \sum_l \lambda_l \right)^{-1}$. For $a = 1$ and $b = 0$, the maximum a posteriori estimate coincides with the maximum likelihood estimate. If $a < 1$, one may obtain sparse estimates of the weights, as the numerator of (10) may become negative. By changing the value of the hyperparameter a, one may obtain different levels of sparsity. It could for example be set by cross-validation. Next, we propose a Bayesian treatment of λ. We first show that it is straightforward to derive a Gibbs sampler for (2) given our data augmentation. Subsequently, we derive a deterministic approximate Bayesian inference scheme, closely related to the Gibbs sampler.

3.3 Gibbs sampler

Using the same data augmentation, we can define a Gibbs sampler for sampling from the posterior distribution $p(\mathcal{C}, \mathcal{Z}, \lambda|\mathcal{Y})$. The conditional distribution $p(\mathcal{C}|\mathcal{Y}, \lambda)$ factorizes, such that each C_i can be updated conditioned on (Y_i, λ) using (9). Likewise, the conditional $p(\mathcal{Z}|\mathcal{Y}, \lambda)$ factorizes, such that each Z_i can be updated conditioned on λ as the posterior reverts to the prior (5). Finally, the conditional for each λ_{kj} is of the form $p(\lambda_{kj}|\mathcal{Y}, \mathcal{C}, \mathcal{Z}) \propto p(\lambda_{kj}) e^{\ln p(\mathcal{Y}, \mathcal{C}, \mathcal{Z}|\lambda)}$. Since the Gamma prior is conjugate to the complete likelihood, the conditional for λ_{kj} still follows a Gamma distribution.

To summarize, the Gibbs sampler is given by

$$C_i|Y_i, \lambda \sim \mathrm{Disc}\left(\frac{W_{i1}\lambda_{Y_i 1}}{W_i^\top \lambda_{Y_i}}, \ldots, \frac{W_{ip}\lambda_{Y_i p}}{W_i^\top \lambda_{Y_i}} \right), \tag{11}$$

$$Z_i|\lambda \sim \mathrm{Exp}\left(W_i^\top \sum_{\ell=1}^{K} \lambda_\ell \right), \tag{12}$$

$$\lambda_{kj}|\mathcal{Y}, \mathcal{C}, \mathcal{Z} \sim \mathrm{Gam}\left(a + n_{kj}, b + \sum_{i=1}^{n} Z_i W_{ij} \right). \tag{13}$$

3.4 Variational approximation

Next, we turn our attention to a deterministic approximation instead of sampling. Variational EM maximizes the variational lower bound (Neal & Hinton,

1998), which can be re-written in the form

$$\ln p(\mathcal{Y}) \geq -\mathcal{F}(q) = \langle \ln p(\mathcal{Y}, \mathcal{C}, \mathcal{Z}, \lambda) \rangle_{q(\mathcal{C})q(\mathcal{Z})q(\lambda)} + \mathrm{KL}\left[q(\mathcal{C})q(\mathcal{Z})q(\lambda) \| p(\mathcal{C}, \mathcal{Z}, \lambda|\mathcal{Y}))\right]. \quad (14)$$

The approximation assumes that the latent variables and the parameters are independent a posteriori given the data. The approximate posteriors are given by

$$q(\mathcal{C}) \propto p(\mathcal{C}) e^{\langle \ln p(\mathcal{Y}, \mathcal{Z}, \lambda|\mathcal{C}) \rangle_{q(\mathcal{Z})q(\lambda)}}, \quad (15)$$

$$q(\mathcal{Z}) \propto e^{\langle \ln p(\mathcal{Y}, \mathcal{C}, \mathcal{Z}, \lambda) \rangle_{q(\mathcal{C})q(\lambda)}}, \quad (16)$$

$$q(\lambda) \propto p(\lambda) e^{\langle \ln p(\mathcal{Y}, \mathcal{C}, \mathcal{Z}|\lambda) \rangle_{q(\mathcal{C})q(\mathcal{Z})}}, \quad (17)$$

In practice, this boils down to cycling through the following updates

$$\rho_{kji} \propto \delta_{Y_i k} W_{ij} e^{\langle \ln \lambda_{kj} \rangle}, \quad \langle z_i \rangle = \frac{1}{W_i^\top \sum_\ell \langle \lambda_\ell \rangle} , \quad (18)$$

$$a_{kj} = a + \langle n_{kj} \rangle , \qquad b_{kj} = b + \sum_i \langle z_i \rangle W_{ij} , \quad (19)$$

where $\langle \lambda_{kj} \rangle = \frac{a_{kj}}{b_{kj}}$, $\langle \ln \lambda_{kj} \rangle = \psi(a_{kj}) - \ln b_{kj}$, and $\langle n_{kj} \rangle = \sum_i \rho_{kji}$.

The similarity between the variational posteriors and the Gibbs sampler is striking. For example, $p(\lambda|\mathcal{Y}, \mathcal{C}, \mathcal{Z})$ and $q(\lambda)$ have the same form, except that the counts n_{kj} and the latent auxiliary variables Z_i are respectively replaced by their expected values, namely $\langle n_{kj} \rangle$ and $\langle Z_i \rangle$. However, unlike Gibbs sampling, the convergence of variational inference and the correctness of the algorithm are easy to check by monitoring the variational lower bound, which monotonically increases at each iteration. Hence, considering both approaches in parallel is convenient when debugging the Gibbs sampling code.

3.5 Identifiability and hyperparameters estimation

The likelihood (2) is invariant to a rescaling of the λ_{kj}'s. As a consequence, the scaling hyperparameter b does not have any effect on the prediction, It can be set to an arbitrary value, e.g. $b = 1$. Let

$$\Lambda = \sum_k \sum_j \lambda_{kj} , \qquad \overline{\lambda_{kj}} = \frac{\lambda_{kj}}{\Lambda} .$$

Because of the invariance w.r.t. rescaling, Λ is not likelihood identifiable and

$$p(\Lambda, \overline{\lambda}|\mathcal{Y}) = p(\Lambda) p(\overline{\lambda}|\mathcal{Y}) ,$$

with $p(\Lambda) = \mathrm{Gam}(\Lambda; Kp, b)$. It follows that

$$\Lambda^{\mathrm{MAP}} = \frac{Kp - 1}{b} .$$

To improve the mixing of the Markov chain, an additional sampling step can be added by sampling the total mass Λ from the prior then rescale the parameters λ adequately. While it would improve the mixing of the Markov chain, this is useless here as we are typically interested in the prediction with the normalized weights.

The parameter a can be estimated by cross-validation in the EM algorithm. For full Bayesian inference, we assume the following flat improper prior:

$$p(a) \propto \frac{1}{a}$$

and we add a Metropolis-Hastings sampling step in the Gibbs sampler. In the variational EM algorithm, a is estimated by type II Maximum Likelihood:

$$a \leftarrow \arg\max_a \left\{ -\mathcal{F}(q, \lambda) + \ln p(a) \right\}.$$

While this does not lead to a closed form update for a, the maximization can be performed by a line search.

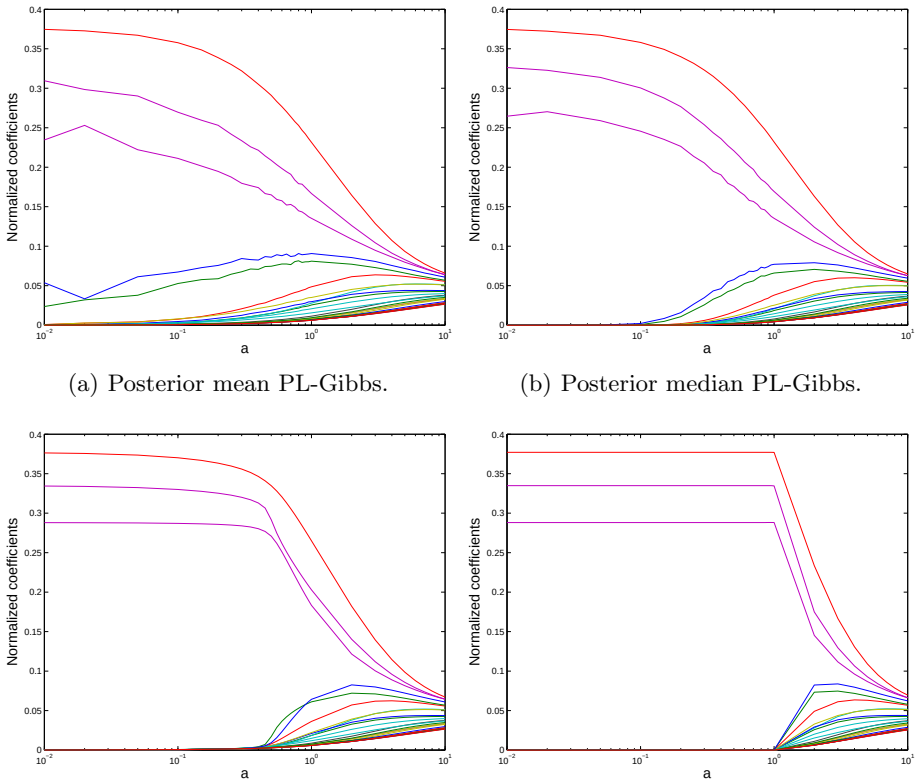
4 Experiments

In this section we investigate the performances of the Plackett-Luce regression model, which we will denote PL-EM, PL-Gibbs or PL-Var depending on the training algorithm we used (see Section 3).

4.1 Sparsity and regularization paths

First, we compare the sparsity properties of PL-EM, PL-Gibbs and PL-Var. We consider the iris dataset, which is available from the UCI repository. As with ℓ_1-penalization in generalized linear regression models, varying the value of the regularization coefficient a will enforce the amount of sparsity over the estimated λ of our model.

Figure 2 shows the regularization path for the coefficients as a function of a, using the MAP estimate obtained with EM, the posterior mean and median obtained with Gibbs sampling, and the posterior mean obtained with the variational approximation. Regularization paths report the values of the weights obtained when training the models with decreasing values of the hyperparameter a. When considering MAP, we obtain exactly sparse estimates for sufficiently small values of a. Similarly to what is observed with the Bayesian lasso (Park & Casella, 2008) the posterior mean estimates obtained by Gibbs sampling are not sparse but more concentrated around zero as the value of a decreases. The posterior median leads to sparser estimates, similarly to what is observed with the Bayesian lasso. Interestingly, the variational approximation shows similar sparsity as the posterior median, but converges faster.



(a) Posterior mean PL-Gibbs.

(b) Posterior median PL-Gibbs.

(c) Posterior mean PL-Var.

(d) MAP PL-EM.

Figure 2: Regularization paths for the iris dataset. The paths are the values of the weight parameters as a function of a. They are obtained by training the models for decreasing values of a. Each colour corresponds to a different λ_{kj}. The coefficient values are normalized to make the scales comparable.

4.2 Toy datasets

Next, we investigate the predictive and computational performances of the proposed PL models. We compare to Bayesian multinomial logistic regression (or logit). More specifically, we consider the Gibbs sampler described by Holmes & Held (2006), as the authors provide detailed pseudo-code on the algorithm, and the algorithm has no tuning parameter as in the case of our Gibbs sampler (as opposed to hybrid Monte Carlo, for example).[2] However, we will consider a sparse extension inspired by the Bayesian lasso of Park & Casella (2008). The Gibbs sampler is detailed in the Appendix.

We compare both models/algorithms in terms of computational efficiency and predictive performance on five datasets from the UCI repository: three polychotomous (Lenses, Wine and Iris) and three binary (Heart, German and Pima). Summaries of the different datasets are given in Table 1.

[2]To be precise we used the corrected version by van der Lans (2011); Holmes & Held (2011).

Name	d	K	n
Lenses	4	3	24
Wine	13	3	178
Iris	4	3	150
Heart	13	2	270
German	24	2	1000
Pima	8	2	768

Table 1: Dataset characteristics: covariates (d), categories (K), and data points (n).

As Holmes & Held (2006), we first conduct 5000 burn in iterations then the next 5000 iterations are used to collect posterior samples. Each method was implemented in Matlab on a standard computer. We compare the predictive performance by computing the average number of misclassifications over 20 replications (i.e. random splits of the data in training and test set). Predictions are based on Bayesian averaging over 5000 samples. The results are reported in Table 2. It can be observed that the proposed Plackett-Luce models are competitive with sparse Bayesian multinomial logit.

This is confirmed when computing the area under the receiver operating curve (Figure 3). Gibbs sampling is in general beneficial compared to the variational approximation. This can be explained by the fact the PL-Var tends to provide sparser solutions, hence loosing predictive power.

We used the same experimental setup to investigate the relative efficiency of PL-Gibbs and sparse Bayesian multinomial logit. The two methods are compared by calculating the effective sample size using the posterior samples for each covariate

$$ESS = \frac{N}{1 + 2\sum_k \gamma(k)} \ ,$$

where $N = 5000$ and $\sum_k \gamma(k)$ is the sum of the K monotone sample auto-correlations as estimated by the initial monotone sequence estimator(Geyer, 1992; Girolami & Calderhead, 2011). We report the minimum ESS over the set of whole set of covariates. Table 3 shows the results for the different datasets, based on 20 replications. It can be observed that ESS is better (higher) for sparse multinomial logit in the binary cases. However, the running time is also higher, leading to a relative speed (ratio of time to ESS) of 2 to 3 in favour of sparse logit. However, when the number of classes increases this trend is drastically changed with a relative speed of approximately 30 in favour of PL-Gibbs. As discussed in Section 5, we attribute this to the fact that, in multinomial logit, the coefficients of one class are sampled conditioned on the coefficients of the other classes. Unfortunately, these coefficients are strongly correlated, resulting in a poor mixing. By contrast, the PL regression models jointly sample (or update) the coefficients associated to all the classes.

4.3 Real data examples

First, we consider the colon cancer data[3]. This is a binary classification problem consisting of 62 data points and 2001 features. In general, 50 data points are used for training and 12 for testing. The average number of misclassifications is 0.36 ± 0.11 for PL-Gibbs and 0.33 ± 0.15 for sparse binary logit. Hence, both model perform similarly. However, sparse binary logit based on the sampler of Holmes & Held (2006) runs about 500 times slower.

Second, we consider the sushi data (Kamishima, 2003). Individuals were asked which out of 10 sushis they preferred. The number of training data points is 4000 and the number of test data points is 1000. The number of features is 11. We repeated the experiment 5 times. Here, the average number of misclassifications for Plackett-Luce and sparse multinomial logit were respectively 656 and 672. While the performances are poor in both cases, PL-Gibbs performs slightly better and is much faster to run. Generating 5000 samples with PL-Gibbs takes a couple of minutes on a standard quad-core laptop, while it takes approximately 6 hours with the sampler of Holmes & Held (2006).

[3] `http://perso.telecom-paristech.fr/~gfort/GLM/Programs.html`

Dataset	Sparse Logit	PL-Gibbs	PL-Var
Pima	.240 (.024)	**.238** (.015)	.239 (.017)
Iris	**.086** (0.086)	.186 (.055)	.181 (.057)
Heart	.215 (.046)	**.170** (.023)	.223 (.056)
German	.262 (.021)	**.260** (.017)	.298 (.015)
Lenses	**.700** (0.087)	.825 (.071)	.821 (.073)
Wine	.080 (0.038)	**.048** (.019)	.093 (.048)

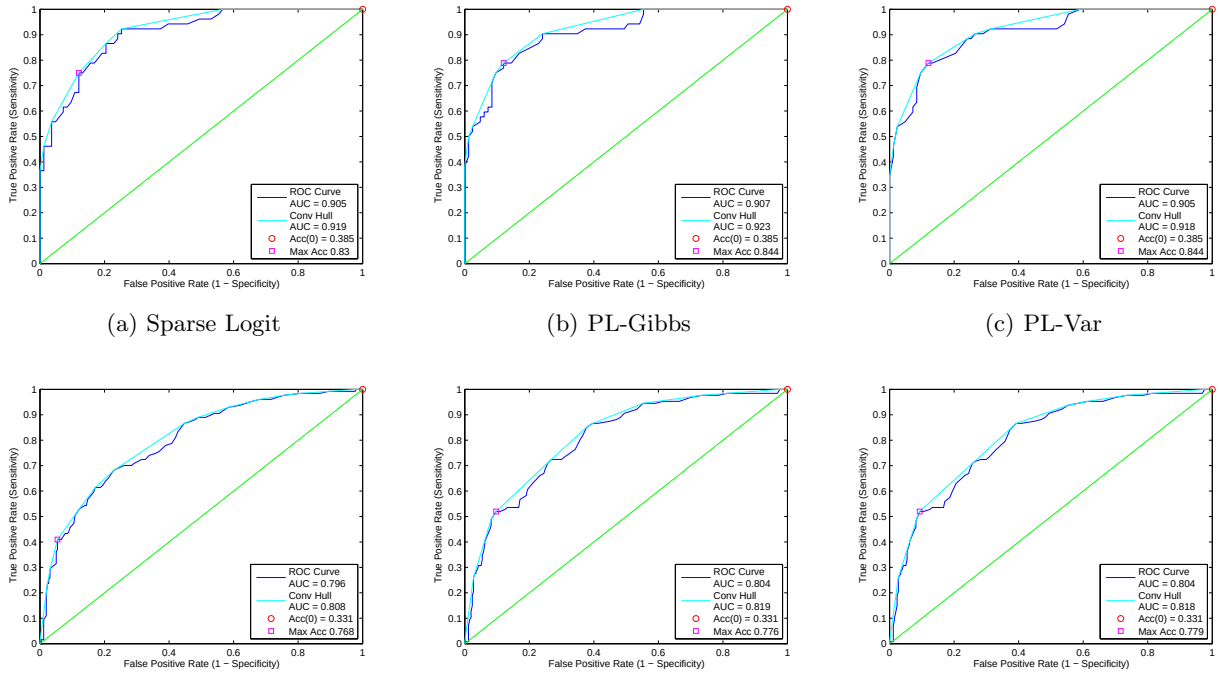
Table 2: Average number of misclassifications (and standard deviations) for the different dataset.

Dataset	Method	Time	ESS	$\frac{\text{Time}}{\text{ESS}}$	Relat. Speed
Lenses	Sp. Logit	26.0	304	0.086	1
	PL-Gibbs	8.6	2916	0.003	29
Wine	Sp. Logit	184.9	8	22.960	1
	PL-Gibbs	15.1	23	0.655	35
Iris	Sp. Logit	139.9	8	17.996	1
	PL-Gibbs	10.8	14	0.747	24
Heart	Sp. Logit	128.8	270	0.477	2
	PL-Gibbs	16.0	14	1.185	1
German	Sp. Logit	561.1	418	1.342	3
	PL-Gibbs	58.0	17	3.426	1
Pima	Sp. Logit	409.6	675	0.607	2
	PL-Gibbs	23.2	23	1.028	1

Table 3: Efficiency of Sparse multinomial logit and Plackett-Luce regression on the different datasets.

5 Discussion and extensions

The Gibbs sampler of the Plackett-Luce regression model only requires to sample from highly standard distributions (Exponential, Gamma and Discrete) for which very efficient generators exist. This is to be compared to Bayesian (multinomial) logistic regression, where it is required to sample from the truncated logistic and from other distributions without standard form, like for example the Kolmogorov-Smirnov, with rejection sampling (Holmes & Held, 2006). An important drawback of recent treatments of multinomial logistic regression like the one of Holmes & Held (2006) is that the multinomial outcomes are expressed as a sequence of binary outcomes. In other words, each vector of coefficient β_k is sampled conditional on the others.This explains why we observed experimentally



(a) Sparse Logit (b) PL-Gibbs (c) PL-Var

(d) Sparse Logit (e) PL-Gibbs (f) PL-Var

Figure 3: ROC for heart (first row) and pima (second row) datasets.

the relatively poor mixing properties of multinomial logit when the number of classes is more than 2. Inference in multinomial probit models often suffer from the same weakness (e.g., Albert & Chib, 1993). In the Gibbs sampler associated to the model we propose, the parameters λ_{kj} are updated jointly and independently given the latent variables.

Throughout the paper, we arbitrarily used an exponential transformation of the covariates. Additional flexibility could be introduced into the model by allowing other transformations, which would better characterize the classes. Indeed, one of the limitations of the proposed generative model is that for each observation X_i, it is assumed first a feature C_i is selected, and then its class Y_i is drawn given C_i. When features do not uniquely characterize the classes, it becomes difficult to distinguish them. Figure 4 shows the decision boundaries for the iris data set (when only considering the petal length and width). It could be argued that these decision boundaries are counter-intuitive, especially in contrast to the decision boundaries of the sparse multinomial logit. This is confirmed by the poor performance of PL regression on the iris data set (see Table 2). However, by introducing additional transformation, more natural boundaries can be obtained. For example, to obtain Figure 4) we considered the additional features $\exp(X_{i1} + X_{i2})$ and $\exp(-X_{i1} - X_{i2})$.

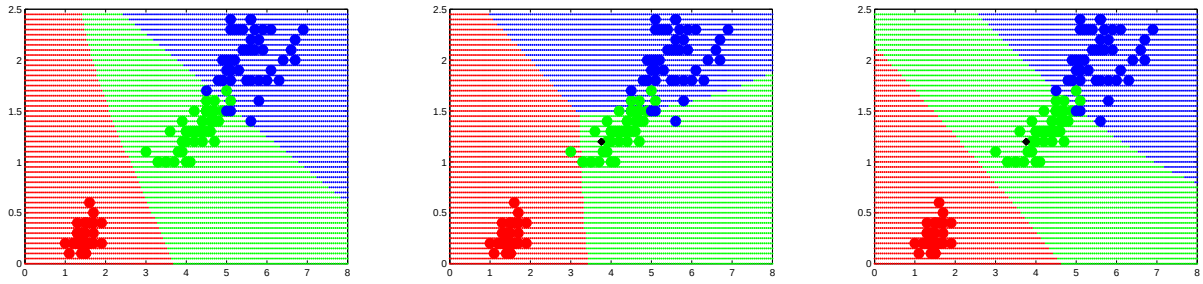
In general, devicing features of this kind would require prior knowledge of the problem at hand, which is rarely the case.

In this paper, we have focused on a Gamma prior for the weights parameters λ_{kj}. Now assume that $a = \alpha/p$. When the number of covariates p and hence $p^\star$ becomes very large, we have

$$\Lambda = \sum_{j=1}^{\infty} \lambda_{kj} \sim \text{Gam}(\alpha, b) \ .$$

Hence, the sequence of normalized ordered weights $\frac{\lambda_{k\sigma(1)}}{\Lambda} > \frac{\lambda_{k\sigma(2)}}{\Lambda} > \dots$ is drawn from the Poisson Dirichlet distribution of parameter α(Pitman, 1996), which is the distribution of the ordered weights in a draw from a Dirichlet process (Teh, 2010). The limit may give useful hints on the behavior of the model with a large number of covariates.

The model could also be directly extended by replacing the Gamma prior by the larger family of generalized inverse Gaussian distributions, which admits the Gamma distribution as a special case, and is also a conjugate prior for the complete-likelihood. Hence, the full conditional distribution of λ_{kj} follows a generalized inverse Gaussian distribution. The use of this distribution may offer more flexibility in the modelling of the tails of the prior for the parameters λ_{kj}.



(a) Sparse Logit. (b) PL-Gibbs. (c) PL-Gibbs with additional features.

Figure 4: Decision boundaries for iris dataset (feature 3 and 4).

Finally, it should be noted that the model studied in this paper has been suggested by Caron & Doucet (2012). However, no discussion of the algorithms, nor any study on the fit of this model to data were discussed there.

A Bayesian sparse multinomial logistic regression

In this Appendix, we provide a Gibbs sampler for sparse multinomial logistic regression. The likelihood is given by (1) and a sparsity-promoting prior is imposed on the regression weights β_{kj}, $k = 1, \ldots, K-1$ and $j = 1, \ldots, p$. More specifically, we consider the following hierarchical prior:

$$\beta_{kj}|\tau_{kj} \sim \mathcal{N}(0, \tau_{kj}) \ , \qquad \tau_{kj} \sim \text{Exp}\left(\frac{\theta}{2}\right).$$

This model ensures that marginally β_{kj} follows a double-exponential (a.k.a. Laplace) distribution of parameter θ. It is routinely used in Bayesian generalized linear mode to induce sparsity Genkin et al. (2007); Park & Casella (2008); Griffin & Brown (2010).We further assume a Gamma prior for the hyperparameter θ for conjugacy reasons (see below (20)):

$$\theta \sim \text{Gam}(c, d) \ .$$

As noted by Park & Casella (2008), the improper prior $p(\theta) \propto \frac{1}{\theta}$ should be avoided in this model as it leads to an improper posterior for θ.

We can define a Gibbs sampler for the above model by using auxiliary variables u_k, $k = 1, \ldots, K-1$ as defined in (Holmes & Held, 2006). We therefore can define a Gibbs sampler to sample from the full posterior distribution $p(\beta, u, \tau, \theta|y)$. The sampler is defined as follows at each iteration:

- For $k = 1, \ldots, K-1$:
  - Sample $u_k, \beta_k|\beta_{-k}, u_{-k}, \{\tau_{kj}\}$ as described in (Holmes & Held, 2006; van der Lans, 2011; Holmes & Held, 2011)
  - For $j = 1, \ldots, J$, sample

$$\frac{1}{\tau_{kj}} \sim \text{iGauss}\left(\sqrt{\frac{\theta^2}{\beta_{kj}^2}}, \theta^2\right).$$

- Sample θ as follows:

$$\theta|\{\tau_{kj}\} \sim \text{Gam}\left(Kp + c, \sum_k \sum_j \frac{\tau_{kj}}{2} + d\right). \quad (20)$$

The notation $\text{iGauss}\,(a, b)$ denotes the inverse-Gaussian distribution with density given by

$$p(x) = \sqrt{\frac{b}{2\pi}} x^{-3/2} \exp\left(-\frac{b(x-a)^2}{2a^2 x}\right).$$

References

Agresti, A. *Categorical Data Analysis*. Wiley, 1990.

Albert, J.H. and Chib, S. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, pp. 669–679, 1993.

Caron, F. and Doucet, A. Efficient Bayesian inference for generalized Bradley-Terry models. *Journal of Computational and Graphical Statistics*, 21:174–176, 2012.

Cawley, G.C., Talbot, N.L.C., and Girolami, M. Sparse multinomial logistic regression via Bayesian l1 regularisation. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, pp. 209. The MIT Press, 2007.

Dey, D., Ghosh, S.K., and Mallick, B.K. *Generalized linear models: A Bayesian perspective*. CRC Press, 2000.

Diaconis, P. *Group representations in probability and statistics, IMS Lecture Notes*, volume 11. Institute of Mathematical Statistics, 1988.

Genkin, A., Lewis, D.D., and Madigan, D. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

Geyer, C.J. Practical Markov chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.

Girolami, M. and Calderhead, B. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

Griffin, J.E. and Brown, P.J. Inference with normal-gamma prior distributions in regression problems. *Bayesian Analysis*, 5(1):171–188, 2010.

Guiver, J. and Snelson, E. Bayesian inference for Plackett-Luce ranking models. In *International Conference on Machine Learning*, 2009.

Holmes, C.C. and Held, L. Bayesian auxiliary variable models for binary and multinomial regression. *Bayesian Analysis*, 1(1):145–168, 2006.

Holmes, C.C. and Held, L. Response to van der Lans. *Bayesian Analysis*, 6(2):357–358, 2011.

Kamishima, T. Nantonac collaborative filtering: Recommendation based on order responses. *KDD*, pp. 583–588, 2003.

Krishnapuram, B., Carin, L., Figueiredo, M.A.T., and Hartemink, A.J. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 957–968, 2005.

Luce, R.D. *Individual choice behavior: A theoretical analysis*. Wiley, 1959.

Luce, R.D. The choice axiom after twenty years. *Journal of Mathematical Psychology*, 15:215–233, 1977.

McCullagh, P. and Nelder, J.A. *Generalized linear models*. Chapman & Hall/CRC, 1989.

McFadden, D. Conditional logit analysis of qualitative choice behavior. *Frontiers in Econometrics*, pp. 105–142, 1973.

Neal, R. M. and Hinton, G. E. A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I. (ed.), *Learning in Graphical Models*, pp. 355–368. MIT press, 1998.

Park, T. and Casella, G. The Bayesian lasso. *Journal of the American Statistical Association*, 103(482): 681–686, 2008.

Pitman, J. Some developments of the Blackwell-MacQueen urn scheme. *Lecture Notes-Monograph Series*, pp. 245–267, 1996.

Plackett, R. The analysis of permutations. *Applied Statistics*, 24:193–202, 1975.

Talhouk, A., Doucet, A., and Murphy, K.P. Efficient Bayesian inference for multivariate probit models with sparse inverse covariance matrices. *Journal of Computational and Graphical Statistics*, to appear, 2012.

Teh, Y.W. Dirichlet process. *Encyclopedia of machine learning*, pp. 280–287, 2010.

van der Lans, R. Bayesian estimation of the multinomial logit model: A comment on Holmes and Held. *Bayesian Analysis*, 6(2):353–356, 2011.

Deterministic MDPs with Adversarial Rewards and Bandit Feedback

Raman Arora
TTIC
6045 S. Kenwood Ave.
Chicago, IL 60637, USA

Ofer Dekel
Microsoft Research
1 Microsoft Way
Redmond, WA 98052, USA

Ambuj Tewari
Department of Statistics
University of Michigan
Ann Arbor, MI 48109, USA

Abstract

We consider a Markov decision process with deterministic state transition dynamics, adversarially generated rewards that change arbitrarily from round to round, and a bandit feedback model in which the decision maker only observes the rewards it receives. In this setting, we present a novel and efficient online decision making algorithm named *MarcoPolo*. Under mild assumptions on the structure of the transition dynamics, we prove that MarcoPolo enjoys a regret of $O(T^{3/4}\sqrt{\log T})$ against the best deterministic policy in hindsight. Specifically, our analysis does not rely on the stringent unichain assumption, which dominates much of the previous work on this topic.

1 INTRODUCTION

Sequential decision making problems come in a variety of different flavors. In some cases it is natural to model the environment as a stochastic process, while in other cases we require the more robust assumption that the environment behaves adversarially. In both cases, we can either assume that the environment is stationary or that it evolves over time. For some problems it suffices to assume that the decision-making agent is stateless, while for other problems we may need to assume that the agent has a state that depends on its past actions. We must also specify what level of feedback the agent observes after making each decision. Finally, if we assume that the agent indeed has a state, we must decide whether the state transition dynamics are deterministic or stochastic, and whether they are known or unknown to the agent.

In this paper, we consider sequential decision making in an adversarial and nonstationary environment where the agent has a state. That is, on each round of the sequential decision-making problem, the agent is in one of finitely many states $\mathcal{S}$ and performs an action from a finite action set $\mathcal{A}$. The rewards received by the agent depend both on its action and its state, and the mapping from state-action pairs to rewards is controlled by an adversary and changes arbitrarily over time. We assume that the only feedback that the agent observes is the value of the rewards that it receives, and specifically, that it does not get to see the rewards that it could have received if it had acted differently or if it were in a different state. This limited type of feedback is commonly known as *bandit feedback*. In fact, this feedback model is so natural that a typical researcher in reinforcement learning may not even add the qualifier "bandit" to it. However, we prefer to explicitly mention it to distinguish our setting from early work (Even-Dar et al., 2009) on adversarial MDPs that assumed more extensive feedback.

We also assume that the state transition dynamics are deterministic and fully known to the agent. However, we note that, in the case of deterministic dynamics, it is easy to relax the assumption that the dynamics are known. Following ideas of Ortner (2010), the agent can spend $\text{poly}(|\mathcal{S}|, |\mathcal{A}|)$ time figuring out the dynamics. Therefore, we assume that the agent knows its current state and knows which state it will transition to as a result of each action. This type of state transition model is often called a *deterministic Markov decision process* (DMDP).

Our adversarial reward setting makes sense when the agent faces other agents with conflicting interests. This happens, for instance, when robots interact with other robots, when bidding agents compete against each other in auctions, when trading agents compete to make money, or when the agent is a character in a video game. In these strategic and adversarial settings, the environment is likely to change in unpredictable and arbitrary ways. In all of the aforementioned decision problems, bandit feedback seems to be the most realistic feedback model. Although we

make pessimistic assumptions about the world, we emphasize that our assumptions are strictly more general than the alternative: if the agent is lucky enough to find itself in a stochastic or stationary environment, or if richer feedback is available, our setting still applies.

The harsh assumptions about the rewards and the limited feedback are contrasted by the optimistic assumption that the state transition dynamics form a DMDP. In some cases, deterministic state transitions are a realistic assumption while in other cases this assumption may be overly simplistic. A stochastic system is often well approximated by a deterministic system (Bertsekas, 2005, Chapter 2). For instance, consider the motivating example in Even-Dar et al. (2009), where the goal is to drive along the shortest path from point a to point b. Although driving a car involves stochastic uncertainties, it is usually reasonable to assume that we have deterministic control over which streets our car takes along the path to our destination. Additionally, many of the uncertainties of driving a car can be modeled as part of the adversarial rewards. Another situation that leads to modeling using DMDPs is when the adversary is a finite state machine whose transitions depend on (equivalence classes of) the history of the agent's actions (Maillard and Munos, 2011).

Modeling the state transitions as a DMDP enables us to handle the adversarial bandit-feedback setting in a principled and rigorous fashion. Obtaining a similar result in the general weakly communicating MDP setting (where state transitions are stochastic) remains an elusive open problem. We hope that the techniques developed in this paper will prove useful toward solving that problem.

Another important design choice is how to evaluate the agent's actions. For simplicity, we assume that every action is available at each state, and we note that this assumption can be easily relaxed. Since the rewards are controlled by an adversary, they only become semantically meaningful when we compare the agent's cumulative reward with a suitable benchmark. We let Π be the set of all deterministic policies, where each $\pi \in \Pi$ is a mapping from $\mathcal{S}$ to $\mathcal{A}$, and we choose our benchmark to be the maximal reward obtained by any policy in Π. In other words, we compare the agent's rewards to those of the best deterministic policy in hindsight. Letting $r_1, r_2, \ldots$ denote the adversarial sequence of reward functions (chosen in advance by the adversary before the agent begins its moves), letting $(s_t^\pi, a_t^\pi)_{t=1}^\infty$ denote the trajectory of state-action pairs generated by policy π, and letting $(s_t, a_t)_{t=1}^\infty$ denote the trajectory of state-action pairs generated by the agent, we define the agent's undiscounted *regret* after T rounds as

$$\max_{\pi\in\Pi} \sum_{t=1}^{T} r_t(s_t^\pi, a_t^\pi) - \mathbb{E}\left[\sum_{t=1}^{T} r_t(s_t, a_t)\right] \quad .$$

The agent's rewards appear in expectation because we allow randomized decisions. We say that the agent is learning if its regret can be upper-bounded by a sub-linear function of T, uniformly for all sequences of reward functions.

As mentioned above, we allow the adversary to modify the rewards for any state-action pair without any restrictions. Therefore, the same sequence of state-action pairs may have high rewards at one time and low rewards at another time. It could very well be that a given action sequence (starting from a given state s) may produce different rewards depending on whether we start performing the sequence on an odd or even time step. Therefore, it is insufficient to merely discover high-reward paths through the state space; we must also consider timing issues.

The main contribution of this paper is *MarcoPolo*, an efficient algorithm that enjoys a regret bound of $O(T^{3/4}\sqrt{\log T})$ against the best deterministic policy in hindsight (Theorem 2). The name MarcoPolo is an approximate acronym for **M**DP with **A**dversarial **R**ewards, weakly **CO**mmunicating structure, **P**artial feedback, by reduction to **O**nline **L**inear **O**ptimization. Anecdotally, Marco Polo was a famous Venetian merchant traveler who explored the world in search of high rewards, just as our algorithm explores the DMDP state space in search of rewards. Moreover, Marco Polo is the name of a well-known children's game that is played in a swimming pool, where one swimmer has to capture the other swimmers using only bandit feedback.

The basic idea behind MarcoPolo is quite simple. We observe that a deterministic policy repeats the same cycle of actions again and again. We then realize that if an oracle were to tell us the cycle length of the best deterministic policy and some additional information about its trajectory, then we could do as well as the best action cycle via a reduction to the *bandit linear optimization* (BLO) setting. Such an oracle is unavailable to us, but we can do almost as well as the oracle using a *multi-armed bandit* (MAB) algorithm. The MAB algorithm explores different cycle lengths and trajectories by invoking the BLO algorithm multiple times. To the best of our knowledge, this two-tier hierarchy involving a MAB algorithm that invokes a BLO algorithm is a novel conceptual contribution to the design space of regret minimization algorithms.

1.1 RELATED WORK

Our setting has much in common with a line of research initiated by Even-Dar et al. (2009) and further developed by Yu et al. (2009) and Neu et al. (2010), which deals with MDPs with adversarial rewards. Even-Dar et al. (2009) assumes a full-information feedback model while Yu et al. (2009) and Neu et al. (2010) only require bandit feedback. While these papers assume that the state transition dynamics are stochastic, they also assume that these dynamics have a *unichain* structure. This means that *every* policy must lead to a Markov chain with a single recurrent class (plus some possibly empty set of transient states). This is a very restrictive assumption, as we discuss in Sec. 2.1. In our setting, we assume that the state transition dynamics are deterministic and *weakly communicating*, i.e. it is possible to move from any state to any other state under *some* policy. Although deterministic transition dynamics are less general than stochastic ones, our analysis requires a much weaker assumption on the connectivity between states.

The work in Yu and Mannor (2009) also deals with MDPs with adversarial rewards, bandit feedback, and stochastic state transition dynamics. Moreover, it allows the stochastic transition dynamics to change over time in an adversarial way. The analysis in Yu and Mannor (2009) relies on the strong assumption that the MDP has a finite mixing time. Again, our analysis has the advantage that it relies on a weaker assumption on state connectivity.

Ties between the reinforcement learning literature and the area of individual sequence prediction and compression in information theory are further developed in Farias et al. (2010).

There is a well developed regret minimization and PAC-MDP literature that deals with a similar problem, except that the rewards are stochastic and not adversarial. We cannot do justice to that extensive literature but merely refer the reader to the survey by Szepesvari (2010) and the references therein. Our work bears some similarities to the work of Ortner (2010), which considers sequential decision making in DMDPs with bandit feedback. However, the rewards in that paper are stochastic and not adversarial, hence, the motivation and techniques used are rather different.

The importance of developing machine learning algorithms that work in reactive environments (where agent's actions impact the future evolution of rewards) has caught the attention of many researchers. de Farias and Megiddo (2006) propose algorithms that combine expert advice in a reactive environment. Ryabko and Hutter (2008) consider the problem of learnability in a very general reactive setting and show that environments that allow a rapid recovery from mistakes are asymptotically learnable. Arora et al. (2012) introduce the notion of policy regret as a more meaningful alternative to regret for measuring an online algorithm's performance against adaptive adversaries. Arora et al. (2012) also presents a general reduction that converts any bandit algorithm with a sublinear regret into an algorithm with a sublinear policy regret.

2 PRELIMINARIES

Let $\mathcal{S}$ be a set of n states and let $\mathcal{A}$ be a finite set of actions. An *adversarial-reward deterministic Markov decision process* (DMDP) is a tuple $(\mathcal{S}, \mathcal{A}, f, r_{1:T})$ where $\mathcal{S}$ is a finite set of *states*, $\mathcal{A}$ is a finite set of *actions*, $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a deterministic *state transition function*, which maps a state-action pair to the next state, and $r_{1:T}$ is a sequence $(r_t)_{t=1}^{T}$ of *reward functions* $r_t : \mathcal{S} \times \mathcal{A} \to [0, 1]$.

We slightly overload our notation and allow the second argument of the state transition function to be a sequence of actions, rather than a single action. Namely, $f\big(s, (a_1, \ldots, a_k)\big)$ is the state we reach if we start from s and perform the action sequence $a_1, \ldots, a_k$. More formally, the extended definition of f is given by the recursion

$$f\big(s, (a_1, \ldots, a_k)\big) \;=\; f\Big(f\big(s, (a_1, \ldots, a_{k-1})\big),\; a_k\Big) \;\; .$$

We also find it useful to denote the set of state-action pairs that lead to a given state s by $\mathrm{I}(s)$. That is,

$$\mathrm{I}(s) = \{(s', a') \;:\; f(s', a') = s\} \;\; .$$

A deterministic *policy* is a mapping $\pi : \mathcal{S} \to \mathcal{A}$. Together with an initial state s_1, each policy defines a sequence of actions and states. Since $\mathcal{S}$ is finite, a state will eventually repeat, and from then onwards the policy will keep repeating the same cycle of actions. We categorize a policy π by the length of this cycle and we let Π_k denote the set of deterministic policies that induce cycles of length k. We prove a regret bound that compares the algorithm's cumulative reward with the reward of the best deterministic policy in $\Pi_{\leq L} = \bigcup_{k \leq L} \Pi_k$, where L is a user-defined constant. Since the cycle length is bounded by n, setting $L = n$ gives a regret bound that compares the algorithm to all deterministic policies. The benefit of proving a regret bound for $L < n$ is to demonstrate how the bound degrades gracefully with the complexity of the competitor class.

Our analysis deals with weakly communicating DMDPs (see Section 2.1 for definitions of weakly communicating MDP and the associated closed set of

states) and requires the following additional assumption.

Assumption 1. *For any pair of states $s \neq s'$ in the closed set, there exists an action sequence $a = (a_1, \ldots, a_d)$ (of length exactly d) such that $f(s, a) = s'$.*

Note that the assumption only talks about *existence* of an action sequence. We note that finding the action sequence can be done efficiently, e.g. using dynamic programming. This assumption is a very mild one and it may well be possible to dispense with it entirely. A simple sufficient condition that implies this assumption is the existence of at least one self-loop in the DMDP. Another sufficient condition is that the transition graph is symmetric (i.e. if $f(s, a) = s'$ then there exists an opposite action $a' \in \mathcal{A}$ such that $f(s', a') = s$) and there exists an odd length cycle in the graph.

2.1 THE UNICHAIN CONDITION IS TOO STRINGENT

The difference between a DMDP and a general MDP is that, in the latter, the transition dynamics are stochastic, i.e. f maps an (s, a) pair to a *distribution* over states. Two well known subclasses of MDPs are *recurrent* (or ergodic) and *communicating* MDPs. In a recurrent MDP, *every* policy π induces a Markov chain with a single recurrent class. The more general notion of a communicating MDP only requires that it be possible to move (with non-zero probability) from any state to any other state using *some* policy. Note the difference in quantifiers — "every" versus "some" — in the definitions of recurrent and communicating MDPs. Thus, the recurrent condition is much more stringent than the communicating condition.

Both notions can be generalized a little bit to allow for some transient states. Thus, we arrive at the classes of *unichain* and *weakly communicating* MDPs. In unichain MDPs, *every* policy induces a Markov chain with a single recurrent class plus a possibly empty set of transient states. In weakly communicating MDPs, there is a *closed* set of states such that each state is accessible, under *some* policy, from any other state in the closed set (with non-zero probability under some policy) plus a possibly empty set of states that are transient under every policy. In the same way as recurrent is a much stronger condition compared to communicating, the unichain condition is much stronger compared to weakly communicating.

We have the (strict) inclusions:

$$\text{recurrent} \subset \text{unichain} \subset \text{weakly communicating} ,$$
$$\text{recurrent} \subset \text{communicating} \subset \text{weakly communicating} .$$

The graph associated with a DMDP is a directed graph with $\mathcal{S}$ as the set of vertices and the set of edges given by

$$\{(s, s') \,:\, \exists a \in \mathcal{A} \text{ s.t. } f(s, a) = s'\} .$$

For DMDPs, the unichain condition can be equivalently stated in terms of its graph. A DMDP is unichain iff *any* two cycles in its graph have a common vertex (Feinberg and Yang, 2008). A DMDP is weakly communicating if, after removing a possibly empty set of states that are transient under all policies, we get a graph that is strongly connected. Thus, any learning algorithm will, after at most n steps, end up in the strongly connected component. Therefore, in our regret analysis, we will assume that the DMDP is communicating.

The "no vertex-disjoint cycles" characterization of the unichain condition for DMDPs shows again why the unichain condition is so stringent. Two self-loops on two distinct states will violate it. It is also not preserved under very simple operations of building larger MDPs from smaller ones. For instance, suppose we start from two unichain DMDPs on disjoint state spaces S_1, S_2 and join them by introducing two actions a_1, a_2 such that $f(s_1, a_1) = s_2$ and $f(s_2, a_2) = s_1$ for some $s_1 \in S_1, s_2 \in S_2$, then the resulting DMDP will not, in general, be unichain (but will be weakly communicating).

3 SOLUTION FOR FIXED LENGTH AND STATE

As a warm-up, we consider a toy version of our problem, where the competitor class is restricted to a smaller subset of $\Pi_{\leq L}$. Assume that we are told to focus on a specific cycle length k and a specific initial state $\bar{s}$. Let $\Pi_{k,\bar{s}}$ denote the subset of policies in Π_k that start in state $\bar{s}$ at time 1 and return to $\bar{s}$ at time $k + 1$. Assume that we are only required to compete with policies in $\Pi_{k,\bar{s}}$.

Next, we characterize the action sequences induced by policies in $\Pi_{k,\bar{s}}$. Define the set

$$\mathcal{C}_{k,\bar{s}} \;=\; \{c \in \mathcal{A}^k \;:\; f(\bar{s}, c) = \bar{s}\} \;\; .$$

$\mathcal{C}_{k,\bar{s}}$ is the set of k-length action sequences such that starting from state $\bar{s}$ and performing an action sequence in $\mathcal{C}_{k,\bar{s}}$ brings us back to state $\bar{s}$. We say that $\mathcal{C}_{k,\bar{s}}$ is the set of *length k action cycles* with respect to a starting state $\bar{s}$. Note that this set contains all of the simple cycles of length k, but may also contain cycles that pass through $\bar{s}$. In this section, we present a randomized online algorithm that competes with any action sequence that consists of recurrences of some cycle in $\mathcal{C}_{k,\bar{s}}$. Note that this set of competitors may

also contain action sequences that are not induced by any policy in $\Pi_{k,\bar{s}}$, for example, due to the fact that $\mathcal{C}_{k,\bar{s}}$ is not restricted to simple cycles.

First, we present a general outline of our algorithm. There is an initial "lock-in" part in the algorithm to take care of the fact that the algorithm will eventually be called as a subroutine when the global time clock reads t_1 units and the agent is in some arbitrary state $s_1 \in \mathcal{S}$. Thus, there is a "phase lock-in" problem: the competitors in $\mathcal{C}_{k,\bar{s}}$ start from $\bar{s}$ at global time 1 and return to it at times $k+1, 2k+1, \ldots$. The agent needs to move from s_1 and get "locked into" one of the cycles so that it is not out of phase. Here is where Assumption 1 comes in handy. The algorithm picks an arbitrary cycle $(c_1, \ldots, c_k) \in \mathcal{C}_{k,\bar{s}}$. At time $t_1 + d$, this competitor will be at state $c_{k'}$ where $k' = (t_1 + d - 1) \mod k + 1$. The algorithm performs a sequence of d actions that bring it to $c_{k'}$ at time $t_1 + d$. Then we waste $k'' = (k - k' + 1) \mod k$ more time steps to come back to $\bar{s}$ and the agent is now "in phase" with $\mathcal{C}_{k,\bar{s}}$.

The algorithm then splits the remaining rounds into epochs of length k, where epoch j starts on round $d + k'' + (j-1)k + 1$ and ends on round $d + k'' + jk$. At the beginning of epoch j, the algorithm defines a distribution θ_j over the set $\mathcal{C}_{k,s}$ and samples an action sequence from this distribution. The algorithm then spends the next k rounds executing the actions in the sequence that it has chosen. The algorithm sticks to the chosen sequence for the duration of the epoch, disregarding the rewards it observes along the way. Only at the end of the epoch does the algorithm look back on its rewards and defines a new distribution θ_{j+1} for the next epoch. In fact, our algorithm will only use the sum of the k rewards, rather than the k individual reward values.

It remains to specify how the algorithm sets the distribution θ_j. It does so via a reduction of our problem to a *bandit linear optimization* (BLO) problem[1]. We briefly describe the BLO problem setting and the interested reader is referred to (Flaxman et al., 2005; Abernethy et al., 2008) for details. A BLO problem is a repeated game between a randomized algorithm and an adversary, which takes place over a predefined polyhedral set $\mathcal{X} \subset \mathbb{R}^n$. Before the game begins, the adversary secretly picks a sequence of vectors $\rho_1, \rho_2, \ldots$, each in $\mathbb{R}^n$. On round i of the game, the algorithm begins by choosing a distribution over $\mathcal{X}$ and sampling a point x_i from that distribution. Next, the adversary

[1]A similar approach is used in (Flaxman et al., 2005; Abernethy et al., 2008) to solve the *bandit shortest path* problem. For more information on the bandit shortest path problem, see (McMahan and Blum, 2004; Awerbuch and Kleinberg, 2004; György et al., 2007)

Algorithm 1: Fixed length k and state $\bar{s}$

input: initial state s_1, global time t_1, length k, state $\bar{s}$, iterations T
initialize: `BLO` with $n = nk|\mathcal{A}|$ and $\mathcal{X}$ as defined in Eqs. (2 – 5)
Pick an arbitrary cycle $(c_1 = \bar{s}, \ldots, c_k) \in \mathcal{C}_{k,\bar{s}}$
Set $k' \leftarrow (t_1 + d - 1) \mod k + 1$
$(a_1, \ldots, a_d) \leftarrow$ get action sequence from s_1 to $c_{k'}$
for $t = 1, \ldots, d$
  perform action a_t: $s_{t+1} \leftarrow f(s_t, a_t)$
  observe (bandit) reward $r_t(s_t, a_t)$
  accumulate $R \leftarrow R + r_t(s_t, a_t)$
Set $k'' \leftarrow (k - k' + 1) \mod k$
Spend k'' time steps visiting the states $c_{k'+1}, \ldots, c_k$
Accumulate the k'' rewards in R
for $j = 1, 2, \ldots$
  $x_j \leftarrow$ get vector from `BLO`
  $\theta_j \leftarrow$ `decompose`(x_j)
  sample action cycle $(c_1 = \bar{s}, \ldots, c_k) \sim \theta_j$
  for $i = 1, \ldots, k$
    $t \leftarrow d + k'' + (j-1)k + i$
    if $t > T$ **then**
      return (R, s_t)
    perform action $a_t = c_i$: $s_{t+1} \leftarrow f(s_t, c_i)$
    observe (bandit) reward $r_t(s_t, c_i)$
    accumulate $R_j \leftarrow R_j + r_t(s_t, c_i)$
  provide R_j as feedback to `BLO`
  $R \leftarrow R + R_j$

computes $\rho_i \cdot x_i$ and reveals the result to the algorithm, while still hiding the vector ρ_i. The value $\rho_i \cdot x_i$ is the algorithm's reward on round i, and cumulative reward over a sequence of m rounds equals $\sum_{i=1}^{m} \rho_i \cdot x_i$. The goal is to accumulate a large reward over time. In this setting, the BLO algorithm presented in Abernethy et al. (2008) guarantees that the algorithm's expected regret after m iterations is bounded as

$$\max_{x \in \mathcal{X}} \sum_{i=1}^{m} \rho_i \cdot x - \mathbb{E}\left[\sum_{i=1}^{m} \rho_i \cdot x_i\right] \leq 4Un^{3/2}\sqrt{m \log(m)} \quad , \tag{1}$$

where U is an upper bound on $\rho_i \cdot x_i$ for all i.

A useful extension to the BLO setting is the double randomization technique, which involves adding an additional layer of randomization to the algorithm's strategy. The algorithm chooses the point x_i normally, but then replaces x_i with an *unbiased estimator* of x_i. In other words, the algorithm can choose any distribution θ_i over $\mathcal{X}$ such that $\mathbb{E}_{X \sim \theta_i}[X] = x_i$. The algorithm then samples a point X_i according to θ_i and plays X_i instead of x_i. Due to linearity, the expected reward $\mathbb{E}[\rho_i \cdot X_i | x_i]$ still equals $\rho_i \cdot x_i$, and the regret bound stated above remains valid. We use this exten-

sion in our reduction below. For more details, please see Abernethy et al. (2008, Section 7).

We are now ready to define the reduction from our problem to the BLO problem. The BLO problem takes place in the space $\mathbb{R}^n$, where $n = n|\mathcal{A}|k$. In this space, each coordinate corresponds to a triplet (s, a, i), where s is a state in $\mathcal{S}$, a is an action in $\mathcal{A}$, and i is a phase index in $[k]$. For any vector $v \in \mathbb{R}^n$, we use the notation $v_{(s,a,i)}$ to refer to the element of v that corresponds to the triplet (s, a, i).

We begin by embedding the sequence of reward functions that occur during epoch j in $\mathbb{R}^n$. The relevant reward functions are $r_{d+k''+jk-k+i}, \ldots, r_{d+k''+jk}$. We represent their values by a vector $\rho_j \in \mathbb{R}^n$ defined as

$$\forall\, s \in \mathcal{S}, a \in \mathcal{A}, i \in [k] \quad \rho_{j,(s,a,i)} = r_{d+k''+jk-k+i}(s, a) \ .$$

Next, we represent each cycle $c \in \mathcal{C}_{s,k}$ by a binary vector $x(c) \in \mathbb{R}^n$ defined as

$$x(c)_{s,a,i} = \begin{cases} 1 & \text{if } a = c_i \text{ and } s = f\big(\bar{s}, (a_1, \ldots, a_{i-1})\big) \\ 0 & \text{otherwise} \end{cases} ,$$

for all $s \in \mathcal{S}, a \in \mathcal{A}$, and $i \in [k]$. Note that $x(c)$ has exactly k non-zero entries that indicate the state-action pairs encountered along the cycle c. Moreover, note that the cumulative reward collected while performing the action sequence c on epoch j simply equals the dot product $\rho_j \cdot x(c)$.

Let $x\big(\mathcal{C}_{k,\bar{s}}\big) = \{x(c)\}_{c \in \mathcal{C}_{k,\bar{s}}}$ denote the set of vectors induced by all of the action cycles in $\mathcal{C}_{k,\bar{s}}$. Selecting a sequence of cycles $c_1, c_2, \ldots$ from the set $\mathcal{C}_{k,\bar{s}}$ so as to maximize the reward defined by the adversarial sequence $r_1, r_2, \ldots$ is equivalent to the problem of selecting a sequence of points $x_1, x_2, \ldots$ from the set $x\big(\mathcal{C}_{k,\bar{s}}\big) \subset \mathbb{R}^n$ so as to maximize the linear rewards defined by the adversarial sequence $\rho_1, \rho_2, \ldots$. However, the latter is not quite a BLO problem yet, since the set of actions $x\big(\mathcal{C}_{k,\bar{s}}\big)$ is not a polyhedral set.

To resolve this problem, we define the set of actions for the BLO algorithm to be $\mathcal{X} = \text{conv}\left(x\big(\mathcal{C}_{k,\bar{s}}\big)\right)$, the convex hull of the set $x\big(\mathcal{C}_{k,\bar{s}}\big)$. Since $x\big(\mathcal{C}_{k,\bar{s}}\big)$ is a finite set, its convex hull is polyhedral. Now, we can use a BLO algorithm to generate a sequence of points in $\mathcal{X}$. The set $\mathcal{X}$ can be defined more concisely using network flow constraints (Bazaraa et al., 2010). In our case, $\mathcal{X}$ can be written as

$$x \geq 0 \ , \quad \textstyle\sum_{a \in \mathcal{A}} x_{(\bar{s},a,1)} = 1 \ , \tag{2}$$

$$\forall s \notin \mathcal{S} \setminus \{\bar{s}\},\ a \in \mathcal{A}, \quad x_{(s,a,1)} = 0 \ , \tag{3}$$

$$\forall (s', a') \notin \mathrm{I}(\bar{s}) \quad x_{(s',a',k)} = 0 \ , \tag{4}$$

$$\forall s \in \mathcal{S},\ 2 \leq i \leq k, \quad \textstyle\sum_{(s',a') \in \mathrm{I}(s)} x_{(s',a',i-1)} - \sum_{a \in \mathcal{A}} x_{(s,a,i)} = 0 \ . \tag{5}$$

Equation (2) ensures convexity: all paths have non-negative weight and the total weight of all paths equals one. The remaining constraints ensure that every vector in the set is indeed a combination of valid cycles that start and end at $\bar{s}$. Specifically, Eqs. (3) and (4) require all paths to begin and end at $\bar{s}$, whereas (5) is the classic flow conservation constraint, which requires that the total weight of paths that enter each state be equal to the total weight that exits that state, at each phase i. This concise representation enables us to run the BLO algorithm efficiently.

However, now the BLO algorithm picks points in $\mathcal{X}$ that do not directly correspond to individual cycles in $\mathcal{C}_{k,\bar{s}}$. Instead, these points are convex combinations of points that correspond to individual cycles. This problem is easily resolved using the double randomization technique described above. Namely, we can find an unbiased estimator of the point chosen by the BLO algorithm that always chooses a vector in the discrete set $x\big(\mathcal{C}_{k,\bar{s}}\big)$. We apply the classic theorem of Carathéodory (1911), which states that any point in the polyhedral set $\mathcal{X} \subset \mathbb{R}^n$ can be represented as a convex combination of at most $n + 1$ extreme points (vertices) of that set. The extreme points of $\mathcal{X}$ are points in the set $x\big(\mathcal{C}_{k,\bar{s}}\big)$ and correspond to individual cycles. Moreover, the decomposition of a point in $\mathcal{X}$ as a convex combination of $n+1$ points in $x\big(\mathcal{C}_{k,\bar{s}}\big)$ can be efficiently computed using a greedy augmenting-path-style algorithm (Bazaraa et al., 2010, Thm. 2.1). In summary, we use the BLO algorithm to choose a point in $\mathcal{X}$, we represent this point as a convex combination of points in $x\big(\mathcal{C}_{k,\bar{s}}\big)$, we interpret this convex combination as a distribution over the respective action cycles in $\mathcal{C}_{k,\bar{s}}$, and we sample a concrete cycle from this distribution.

The pseudo-code of our algorithm is given in Algorithm 1. As mentioned above, the pseudo-code references three external procedures: the procedure that finds a path of length d from s_1 to $\bar{s}$, the BLO algorithm, and the decomposition procedure that represents any point in $\mathcal{X}$ as a distribution over $n+1$ points in $x\big(\mathcal{C}_{k,\bar{s}}\big)$.

Next, we state a regret bound for our algorithm.

Theorem 1. *Let $r_1, r_2, \ldots$ be an arbitrary sequence of reward functions, perhaps adversarially generated. Assume that we run Algorithm 1 for T rounds with this sequence, with parameters $k \leq L$ and $\bar{s} \in \mathcal{S}$, and let $(s_t, a_t)_{t=1}^T$ be the resulting sequence of state-action pairs (see Alg. 1). On the other hand, let $\pi^\star$ be any deterministic policy in $\Pi_{k,\bar{s}}$ and let $(s_t^\star, a_t^\star)_{t=1}^T$ be the sequence of state-action pairs defined by this policy.*

Then the regret is upper bounded by

$$\sum_{t=1}^{T} r_t(s_t^\star, a_t^\star) - \mathbb{E}\left[\sum_{t=1}^{T} r_t(s_t, a_t)\right] \leq 4L^2(n|\mathcal{A}|)^{3/2}\sqrt{T\log(T)} \;+\; (2L+d) \quad .$$

Proof. Since the rewards on each round are bounded in $[0,1]$, the regret on the first d rounds is bounded by d. Then $k'' = (k - k' + 1) \mod k < k$ more rounds are wasted before coming back to $\bar{s}$. The regret on the remaining $T - d - k''$ rounds is bounded by applying the bound for BLO in Eq. (1). Specifically, in our case, we use Hölder's inequality to bound

$$\rho_i \cdot x_i \;\leq\; \|\rho_i\|_\infty \|x_i\|_1 \;\leq\; k \quad ,$$

and therefore we can set U in Eq. (1) to k. The dimensionality is $n|\mathcal{A}|k$. Finally, the number of BLO iterations equals the number of epochs of our algorithm, which is $m = \lfloor (T - d - k'')/k \rfloor \leq T/k$. Plugging these values into Eq. (1) gives the bound

$$4k(nk|\mathcal{A}|)^{3/2}\sqrt{T/k\log(T/k)} \quad ,$$

which is upper bounded by

$$4L^2(n|\mathcal{A}|)^{3/2}\sqrt{T\log(T)} \quad .$$

Finally, if $T - d - k''$ does not divide by k, we have a suffix of at most k actions that incur an additional regret of at most L. □

4 SOLUTION TO THE GENERAL PROBLEM

The result obtained in the previous section can be interpreted as follows. Let $\pi^\star$ be the optimal fixed policy in $\Pi_{\leq L}$, let $k^\star$ be the length of its cycle. Assume, for simplicity, that $\pi^\star$ enters its cycle straightaway and let $s^\star$ be its state at time $t = 1$. If an oracle were to reveal $k^\star$ and $s^\star$ to us, Algorithm 1 would solve the problem with a regret of $O(L^2(n|\mathcal{A}|)^{3/2}\sqrt{T\log(T)})$. Although we do not have access to such an oracle, we can exploit the fact that there are only Ln possible values of (k, s) that we need to explore.

In this section, we present the MarcoPolo algorithm, which solves the general problem defined in Sec. 1. It does so by splitting the online rounds into episodes of length τ (τ is specified later on) and running an instance of Alg. 1 on each episode. At the beginning of episode j, the algorithm chooses values (k_j, s_j) and runs Alg. 1 with these parameters for τ rounds. Note that Alg. 1 starts from scratch on each episode, and does not retain information from previous episodes. Also note that Alg. 1 starts from the state that we happen to be in due to the previous episodes. However, the initial "lock-in" part in Alg. 1 makes sure we get locked in phase with policies in Π_{k_j,s_j}.

Algorithm 2: MarcoPolo

input: episode length τ
initialize: $\hat{s}_1 = s_1, t = 1$
for $j = 1, 2, \ldots$
  $(k_j, s_j) \leftarrow$ get length/state from `MAB`
  $(R_j, \hat{s}_{j+1}) \leftarrow$ Alg. 1 with input $(\hat{s}_j, t, k_j, s_j, \tau)$
  provide R_j as feedback to `MAB`
  $R \leftarrow R + R_j$
  $t \leftarrow t + \tau$

It remains to specify how our algorithm chooses (k_j, s_j) on each episode. Again, we resort to a reduction, this time to the *multi-armed bandit* problem (MAB). The MAB setting is very similar to the BLO setting described above, except that the algorithm's action set is the set of standard unit vectors $\mathcal{X} = \{e_1, \ldots, e_n\}$, rather than a polyhedral set. Namely, the adversary chooses a sequence of reward vectors $\rho_1, \rho_2, \ldots$ before the game begins. On iteration i, the algorithm chooses a standard unit vector x_i, which has a single non-zero coordinate, and its reward $\rho_i \cdot x_i$ is simply the value of ρ_i at that coordinate. The algorithm never sees the full reward vector ρ_j.

The EXP3 algorithm (Auer et al., 2002) solves the MAB problem, and comes with the following regret bound: For any number of iterations m, the expected regret after m iterations is bounded by

$$\max_{x\in\mathcal{X}} \sum_{i=1}^{m} \rho_i \cdot x - \mathbb{E}\left[\sum_{i=1}^{m} \rho_i \cdot x_i\right] \leq U\sqrt{7mn\log(n)} \quad , \quad (6)$$

where U is an upper bound on $\rho_i \cdot x_i$ for all i.

To simplify the presentation and analysis of the reduction to the MAB problem, we slightly modify the algorithm described above. Instead of choosing a single setting (k_j, s_j) on episode j and running Alg. 1 only on that setting, imagine that we run nL copies of Alg. 1 in parallel, one for each choice of (s, k), and then disregard all but the one that corresponds to the chosen setting (k_j, s_j). Clearly, this modified algorithm is equivalent to our original algorithm, albeit its inferior computational complexity. The advantage of thinking of our algorithm in this way is that it makes the reduction to the MAB setting very straightforward.

The MAB problem takes place in the space $\mathbb{R}^n$, with $n = nL$. In this space, each coordinate corresponds to a pair (k, s), where k is a cycle length between 1 and L, and s is a state in $\mathcal{S}$. For any vector $v \in \mathbb{R}^n$, we use the notation $v_{(k,s)}$ to refer to the element of v that corresponds to the pair (k, s). Let $\rho_{j,(k,s)}$ be the

total reward collected by the copy of Alg. 1 that runs with parameters (k, s) on episode j. Note that $\rho_{j,(k,s)}$ takes a random value, since Alg. 1 is a randomized algorithm, but this does not invalidate the reduction. The MAB algorithm chooses a pair (k_j, s_j) on each episode, and this is the pair we use in our algorithm.

Next, we prove a regret bound for our algorithm.

Theorem 2. *Let $r_1, r_2, \ldots$ be an arbitrary sequence of reward functions, perhaps adversarially generated. Assume that we run* MarcoPolo *with this sequence for T rounds and with an episode length of $\tau = \sqrt{T}$, and let $(s_t, a_t)_{t=1}^T$ be the resulting sequence of state-action pairs. On the other hand, let $\pi^\star$ be any deterministic policy in $\Pi_{\leq L}$ and let $(s_t^\star, a_t^\star)_{t=1}^T$ be the sequence of state-action pairs defined by $\pi^\star$. Then the regret is upper bounded by*

$$\sum_{t=1}^T r_t(s_t^\star, a_t^\star) - \mathbb{E}\left[\sum_{t=1}^T r_t(s_t, a_t)\right] \leq CT^{3/4}\sqrt{\log(T)} + o(T^{3/4}) \quad .$$

where $C = 4L^2(n|\mathcal{A}|)^{3/2} + \sqrt{7nL\log(nL)}$.

Proof. Let $\bar{T}$ be the last round on the last full episode performed by the algorithm. The regret suffered on rounds $(\bar{T}+1), \ldots, T$ is trivially bounded by $T - \bar{T} \leq \tau$, and we focus on bounding the regret on rounds $1, \ldots, \bar{T}$.

Recall that we assumed, for the purpose of our analysis, that we run nL parallel instances of Alg. 1 on each episode. Also recall that Alg. 1 starts from scratch when a new episode begins. Let $k^\star$ be the length of the action cycle induced by the policy $\pi^\star$ and let $\tilde{s}^\star$ be the first state that gets repeated under $\pi^\star$ after an initial non-repeating sequence of states of length $t^\star$. Let $s^\star$ be the state that is $t^\star$ time units behind $\tilde{s}^\star$ on the cycle induced by $\pi^\star$. The rewards accumulated by $\pi^\star$ and those by the cycle corresponding to the pair $(k^\star, s^\star)$ differ by at most $t^\star \leq n$. Thus, we compute regret relative to $(k^\star, s^\star)$.

Let $(s'_t, a'_t)_{t=1}^{\bar{T}}$ be the sequence of state-action pairs generated by the copy of Alg. 1 that runs with parameters $(k^\star, s^\star)$. On epoch j we apply Thm. 1 and get

$$\mathbb{E}\left[\sum_{t=j\tau-\tau+1}^{j\tau} r_t(s_t^\star, a_t^\star) - r_t(s'_t, a'_t)\right] \leq 4L^2(n|\mathcal{A}|)^{3/2}\sqrt{\tau\log(\tau)} + (2L+d) \quad .$$

We sum both sides above for $j = 1, \ldots, \bar{T}/\tau$ and get

$$\mathbb{E}\left[\sum_{t=1}^{\bar{T}} r_t(s_t^\star, a_t^\star) - r_t(s'_t, a'_t)\right] \leq 4L^2(n|\mathcal{A}|)^{3/2}T^{3/4}\sqrt{\log(T)/2} + o(T^{3/4}) \quad . \quad (7)$$

Next, we use the regret bound for the EXP3 algorithm, given in Eq. (6), to upper-bound

$$\mathbb{E}\left[\sum_{t=1}^{\bar{T}} r_t(s'_t, a'_t) - r_t(s_t, a_t)\right] = \mathbb{E}\left[\sum_{j=1}^{\bar{T}/\tau} \rho_{j,(k^\star,s^\star)} - \rho_{j,(k_j,s_j)}\right] \quad .$$

We face a minor technical difficulty due to the fact that the EXP3 bound holds for deterministic rewards, whereas the rewards $\rho_1, \rho_2, \ldots$ in our problem are random. However, it is rather straightforward to see that if a bound holds for any individual sequence of reward functions, then it also holds for the expected reward sequence (the skeptical reader is referred to Appendix A in the supplementary material for a rigorous proof).

Since $\rho_{j,(k,s)}$ represents the cumulative reward from τ rounds, and the reward on each round is bounded in $[0, 1]$, then $\|\rho_j\|_\infty \leq \tau \leq \sqrt{T}$. Therefore, we can set $U = \sqrt{T}$ in Eq. (6). Additionally, we have that the number of arms is nL. Finally, the number of MAB iterations is $m \leq T/\tau = \sqrt{T}$. Plugging these values into the bound, we get that

$$\mathbb{E}\left[\sum_{t=1}^{\bar{T}} r_t(s'_t, a'_t) - r_t(s_t, a_t)\right] \leq T^{3/4}\sqrt{7nL\log(nL)} \quad .$$

Summing the above inequality with Eq. (7) proves the theorem. □

5 DISCUSSION

We focused on the sequential decision making setting where the environment changes with time adversarially, state transitions are deterministic, and the algorithm receives bandit feedback. In this setting, we presented the MarcoPolo algorithm, with an undiscounted regret bound of $O(T^{3/4}\sqrt{\log(T)})$ against the best deterministic policy in hindsight. In contrast to previous work, we did not rely on the stringent unichain assumption.

Many interesting open questions remain unanswered: Can we derive a similar result with stochastic transition dynamics; with adversarially changing transition dynamics; when the state is only partially observable? Even in our specific setting, can we prove a lower bound of $O(T^{3/4})$ on regret, or alternatively, can we obtain a better upper bound with a different algorithm?

We also have more general questions regarding the true power of the adversarial DMDP model compared to the classic (fully stochastic) MDP model. The adversarial DMDP model makes a more general assumption on the rewards and a less general assumption on the state transition dynamics. Are the two things somehow interchangeable? In other words, can we identify situations where we can use the power and flexibility of the adversarial reward assumption to capture uncertainty in the state transition dynamics?

These questions are left for future research.

Acknowledgements

A major portion of this work was done when RA and AT were visiting OD at MSR Redmond.

References

J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Proceedings of the 21st Annual Conference on Learning Theory*, pages 263–274, 2008.

R. Arora, O. Dekel, and A. Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

B. Awerbuch and R. D. Kleinberg. Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing*, pages 45–53, 2004.

M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows, Third Edition*. John Wiley and Sons, 2010.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Third edition, 2005.

C. Carathéodory. Über den variabilitätsbereich der fourierschen konstanten von positiven harmonischen funktionen. *Rendiconti del Circolo Matematico di Palermo*, 32:193–217, 1911.

D. P. de Farias and N. Megiddo. Combining expert advice in reactive environments. *Journal of the ACM*, 53(5):762–799, 2006.

E. Even-Dar, S. M. Kakade, and Y. Mansour. Online Markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

V. F. Farias, C. C. Moallemi, B. Van Roy, and T. Weissman. Universal reinforcement learning. *IEEE Transactions on Information Theory*, 56(5): 2441–2454, 2010.

E. A. Feinberg and F. Yang. On polynomial cases of the unichain classification problem for Markov decision processes. *Operations Research Letters*, 36(5): 527–530, 2008.

A. D. Flaxman, A. Tauman Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the 16th annual ACM-SIAM symposium on discrete algorithms*, pages 385–394, 2005.

A. György, T. Linder, G. Lugosi, and G. Ottucsák. The on-line shortest path problem under partial monitoring. *Journal of Machine Learning Research*, 8:2369–2403, 2007.

O. Maillard and R. Munos. Adaptive bandits: Towards the best history-dependent strategy. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR W&CP*, pages 570–578, 2011.

H. B. McMahan and A. Blum. Online geometric optimization in the bandit setting against an adaptive adversary. In *Proceedings of the 17th Annual Conference on Learning Theory*, pages 109–123, 2004.

G. Neu, A. György, C. Szepesvári, and A. Antos. Online Markov decision processes under bandit feedback. In *Advances in Neural Information Processing Systems 23*, pages 1804–1812. MIT Press, 2010.

R. Ortner. Online regret bounds for Markov decision processes with deterministic transitions. *Theoretical Computer Science*, 411(29-30):2684–2695, 2010.

D. Ryabko and M. Hutter. On the possibility of learning in reactive environments with arbitrary dependence. *Theoretical Computer Science*, 405(3):274–284, 2008.

C. Szepesvari. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1), 2010.

J. Y. Yu and S. Mannor. Arbitrarily modulated markov decision processes. In *Proceedings of the IEEE Conference on Decision and Control*, 2009.

J. Y. Yu, S. Mannor, and N. Shimkin. Markov decision processes with arbitrary reward processes. *Mathematics of Operations Research*, 34(3):737–757, 2009.

Video In Sentences Out

Andrei Barbu,[1,*] Alexander Bridge,[1] Zachary Burchill,[1] Dan Coroian,[1] Sven Dickinson,[2] Sanja Fidler,[2] Aaron Michaux,[1] Sam Mussman,[1] Siddharth Narayanaswamy,[1] Dhaval Salvi,[3] Lara Schmidt,[1] Jiangnan Shangguan,[1] Jeffrey Mark Siskind,[1] Jarrell Waggoner,[3] Song Wang,[3] Jinlian Wei,[1] Yifan Yin,[1] and Zhiqi Zhang[3]

[1]School of Electrical & Computer Engineering, Purdue University, West Lafayette, IN, USA
[2]Department of Computer Science, University of Toronto, Toronto, ON, Canada
[3]Department of Computer Science & Engineering, University of South Carolina, Columbia, SC, USA

Abstract

We present a system that produces sentential descriptions of video: who did what to whom, and where and how they did it. Action class is rendered as a verb, participant objects as noun phrases, properties of those objects as adjectival modifiers in those noun phrases, spatial relations between those participants as prepositional phrases, and characteristics of the event as prepositional-phrase adjuncts and adverbial modifiers. Extracting the information needed to render these linguistic entities requires an approach to event recognition that recovers object tracks, the track-to-role assignments, and changing body posture.

1 INTRODUCTION

We present a system that produces sentential descriptions of short video clips. These sentences describe *who* did *what* to *whom*, and *where* and *how* they did it. This system not only describes the observed action as a verb, it also describes the participant objects as noun phrases, properties of those objects as adjectival modifiers in those noun phrases, the spatial relations between those participants as prepositional phrases, and characteristics of the event as prepositional-phrase adjuncts and adverbial modifiers. It incorporates a vocabulary of 118 words: 1 coordination, 48 verbs, 24 nouns, 20 adjectives, 8 prepositions, 4 lexical prepositional phrases, 4 determiners, 3 particles, 3 pronouns, 2 adverbs, and 1 auxiliary, as illustrated in Table 1.

*Corresponding author. Email: andrei@0xab.com.

Additional images and videos as well as all code and datasets are available at http://engineering.purdue.edu/~qobi/uai2012.

coordination:	*and*
verbs:	*approached*, *arrived*, *attached*, *bounced*, *buried*, *carried*, *caught*, *chased*, *closed*, *collided*, *digging*, *dropped*, *entered*, *exchanged*, *exited*, *fell*, *fled*, *flew*, *followed*, *gave*, *got*, *had*, *handed*, *hauled*, *held*, *hit*, *jumped*, *kicked*, *left*, *lifted*, *moved*, *opened*, *passed*, *picked*, *pushed*, *put*, *raised*, *ran*, *received*, *replaced*, *snatched*, *stopped*, *threw*, *took*, *touched*, *turned*, *walked*, *went*
nouns:	*bag*, *ball*, *bench*, *bicycle*, *box*, *cage*, *car*, *cart*, *chair*, *dog*, *door*, *ladder*, *left*, *mailbox*, *microwave*, *motorcycle*, *object*, *person*, *right*, *skateboard*, *SUV*, *table*, *tripod*, *truck*
adjectives:	*big*, *black*, *blue*, *cardboard*, *crouched*, *green*, *narrow*, *other*, *pink*, *prone*, *red*, *short*, *small*, *tall*, *teal*, *toy*, *upright*, *white*, *wide*, *yellow*
prepositions:	*above*, *because*, *below*, *from*, *of*, *over*, *to*, *with*
lexical PPs:	*downward*, *leftward*, *rightward*, *upward*
determiners:	*an*, *some*, *that*, *the*
particles:	*away*, *down*, *up*
pronouns:	*itself*, *something*, *themselves*
adverbs:	*quickly*, *slowly*
auxiliary:	*was*

Table 1: The vocabulary used to generate sentential descriptions of video.

Production of sentential descriptions requires recognizing the primary action being performed, because such actions are rendered as verbs and verbs serve as the central scaffolding for sentences. However, event recognition alone is insufficient to generate the remaining sentential components. One must recognize object classes in order to render nouns. But even object recognition alone is insufficient to generate meaningful sentences. One must determine the *roles* that such objects play in the event. The *agent*, i.e. the doer of the action, is typically rendered as the sentential subject while the *patient*, i.e. the affected object, is typically rendered as the direct object. Detected objects that do not play a role in the observed event, no matter how prominent, should not be incorporated into the description. This means that one cannot use common approaches to event recognition, such as spatiotemporal bags of words [Laptev et al., 2007, Niebles et al., 2008, Scovanner et al., 2007], spatiotemporal volumes [Blank et al., 2005, Laptev et al., 2008, Ro-

driguez et al., 2008], and tracked feature points [Liu et al., 2009, Schuldt et al., 2004, Wang and Mori, 2009] that do not determine the class of participant objects and the roles that they play. Even combining such approaches with an object detector would likely detect objects that don't participate in the event and wouldn't be able to determine the roles that any detected objects play.

Producing elaborate sentential descriptions requires more than just event recognition and object detection. Generating a noun phrase with an embedded prepositional phrase, such as *the person to the left of the bicycle*, requires determining spatial relations between detected objects, as well as knowing which of the two detected objects plays a role in the overall event and which serves just to aid generation of a referring expression to help identify the event participant. Generating a noun phrase with adjectival modifiers, such as *the red ball*, not only requires determining the properties, such as color, shape, and size, of the observed objects, but also requires determining whether such descriptions are necessary to help disambiguate the referent of a noun phrase. It would be awkward to generate a noun phrase such as *the big tall wide red toy cardboard trash can* when *the trash can* would suffice. Moreover, one must track the participants to determine the speed and direction of their motion to generate adverbs such as *slowly* and prepositional phrases such as *leftward*. Further, one must track the identity of multiple instances of the same object class to appropriately generate the distinction between *Some person hit some other person* and *The person hit themselves*.

A common assumption in Linguistics [Jackendoff, 1983, Pinker, 1989] is that verbs typically characterize the interaction between event participants in terms of the gross changing motion of these participants. Object class and image characteristics of the participants are believed to be largely irrelevant to determining the appropriate verb label for an action class. Participants simply fill roles in the spatiotemporal structure of the action class described by a verb. For example, an event where one participant (the agent) *picks up* another participant (the patient) consists of a sequence of two sub-events, where during the first sub-event the agent moves towards the patient while the patient is at rest and during the second sub-event the agent moves together with the patient away from the original location of the patient. While determining whether the agent is a *person* or a *cat*, and whether the patient is a *ball* or a *cup*, is necessary to generate the noun phrases incorporated into the sentential description, such information is largely irrelevant to determining the verb describing the action. Similarly, while determining the shapes, sizes, colors, textures, etc. of the participants is necessary to generate adjectival modifiers, such information is also largely irrelevant to determining the verb. Common approaches to event recognition, such as spatiotemporal bags of words, spatiotemporal volumes, and tracked feature points, often achieve high accuracy because of correlation with image or video properties exhibited by a particular corpus. These are often artefactual, not defining properties of the verb meaning (e.g. recognizing *diving* by correlation with *blue* since it 'happens in a pool' [Liu et al., 2009, p. 2002] or confusing *basketball* and *volleyball* 'because most of the time the [...] sports use very similar courts' [Ikizler-Cinibis and Sclaroff, 2010, p. 506]).

2 THE MIND'S EYE CORPUS

Many existing video corpora used to evaluate event recognition are ill-suited for evaluating sentential descriptions. For example, the WEIZMANN dataset [Blank et al., 2005] and the KTH dataset [Schuldt et al., 2004] depict events with a single human participant, not ones where people interact with other people or objects. For these datasets, the sentential descriptions would contain no information other than the verb, e.g. *The person jumped*. Moreover, such datasets, as well as the SPORTS ACTIONS dataset [Rodriguez et al., 2008] and the YOUTUBE dataset [Liu et al., 2009], often make action-class distinctions that are irrelevant to the choice of verb, e.g. `wave1` vs. `wave2`, `jump` vs. `pjump`, `Golf-Swing-Back` vs. `Golf-Swing-Front` vs. `Golf-Swing-Side`, `Kicking-Front` vs. `Kicking-Side`, `Swing-Bench` vs. `Swing-SideAngle`, and `golf_swing` vs. `tennis_swing` vs. `swing`. Other datasets, such as the BALLET dataset [Wang and Mori, 2009] and the UCF50 dataset [Liu et al., 2009], depict larger-scale activities that bear activity-class names that are not well suited to sentential description, e.g. `Basketball`, `Billiards`, `BreastStroke`, `CleanAndJerk`, `HorseRace`, `HulaHoop`, `MilitaryParade`, `TaiChi`, and `YoYo`.

The year-one (Y1) corpus produced by DARPA for the Mind's Eye program, however, was specifically designed to evaluate sentential description. This corpus contains two parts: the development corpus, C-D1, which we use solely for training, and the evaluation corpus, C-E1, which we use solely for testing. Each of the above is further divided into four sections to support the four task goals of the Mind's Eye program, namely recognition, description, gap filling, and anomaly detection. In this paper, we use only the recognition and description portions and apply our entire sentential-description pipeline to the combination of these portions. While portions of C-E1 overlap with C-D1, ***in this paper we train our methods solely on C-D1 and test our methods solely on the***

portion of C-E1 that does not overlap with C-D1.

Moreover, a portion of the corpus was synthetically generated by a variety of means: computer graphics driven by motion capture, pasting foregrounds extracted from green screening onto different backgrounds, and intensity variation introduced by post-processing. ***In this paper, we exclude all such synthetic video from our test corpus.*** Our training set contains 3480 videos and our test set 749 videos. These videos are provided at 720p@30fps and range from 42 to 1727 frames in length, with an average of 435 frames.

The videos nominally depict 48 distinct verbs as listed in Table 1. However, the mapping from videos to verbs is not one-to-one. Due to polysemy, a verb may describe more than one action class, e.g. *leaving an object on the table* vs. *leaving the scene*. Due to synonymy, an action class may be described by more than one verb, e.g. *lift* vs. *raise*. An event described by one verb may contain a component action described by a different verb, e.g. *picking up an object* vs. *touching an object*. Many of the events are described by the combination of a verb with other constituents, e.g. *have a conversation* vs. *have a heart attack*. And many of the videos depict metaphoric extensions of verbs, e.g. *take a puff on a cigarette*. Because the mapping from videos to verbs is subjective, the corpus comes labeled with DARPA-collected human judgments in the form of a single present/absent label associated with each video paired with each of the 48 verbs, gathered using Amazon Mechanical Turk. We use these labels for both training and testing as described later.

3 OVERALL SYSTEM ARCHITECTURE

The overall architecture of our system is depicted in Fig. 1. We first apply detectors [Felzenszwalb et al., 2010a,b] for each object class on each frame of each video. These detectors are biased to yield many false positives but few false negatives. The Kanade-Lucas-Tomasi (KLT) [Shi and Tomasi, 1994, Tomasi and Kanade, 1991] feature tracker is then used to project each detection five frames forward to augment the set of detections and further compensate for false negatives in the raw detector output. A dynamic-programming algorithm [Viterbi, 1971] is then used to select an optimal set of detections that is temporally coherent with optical flow, yielding a set of object tracks for each video. These tracks are then smoothed and used to compute a time-series of feature vectors for each video to describe the relative and absolute motion of event participants. The person detections

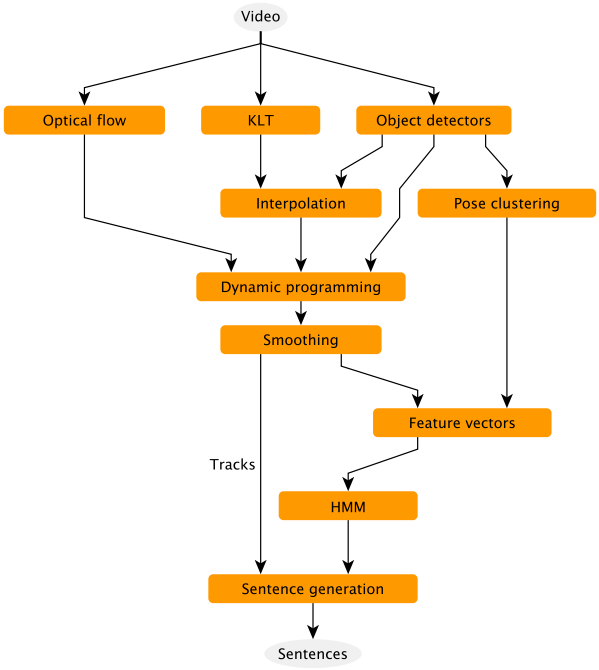


Figure 1: The overall architecture of our system for producing sentential descriptions of video.

are then clustered based on part displacements to derive a coarse measure of human body posture in the form of a body-posture codebook. The codebook indices of person detections are then added to the feature vector. Hidden Markov Models (HMMs) are then employed as time-series classifiers to yield verb labels for each video [Siskind and Morris, 1996, Starner et al., 1998, Wang and Mori, 2009, Xu et al., 2002, 2005], together with the object tracks of the participants in the action described by that verb along with the roles they play. These tracks are then processed to produce nouns from object classes, adjectives from object properties, prepositional phrases from spatial relations, and adverbs and prepositional-phrase adjuncts from track properties. Together with the verbs, these are then woven into grammatical sentences. We describe each of the components of this system in detail below: the object detector and tracker in Section 3.1, the body-posture clustering and codebook in Section 3.2, the event classifier in Section 3.3, and the sentential-description component in Section 3.4.

3.1 OBJECT DETECTION AND TRACKING

In detection-based tracking an object detector is applied to each frame of a video to yield a set of candidate detections which are composed into tracks by selecting a single candidate detection from each frame that maximizes temporal coherency of the track. Felzenszwalb

et al. detectors are used for this purpose. Detection-based tracking requires biasing the detector to have high recall at the expense of low precision to allow the tracker to select boxes to yield a temporally coherent track. This is done by depressing the acceptance thresholds. To prevent massive over-generation of false positives, which would severely impact run time, we limit the number of detections produced per-frame to 12.

Two practical issues arise when depressing acceptance thresholds. First, it is necessary to reduce the degree of non-maximal suppression incorporated in the Felzenszwalb et al. detectors. Second, with the star detector [Felzenszwalb et al., 2010b], one can simply decrease the single trained acceptance threshold to yield more detections with no increase in computational complexity. However, we prefer to use the star cascade detector [Felzenszwalb et al., 2010a] as it is far faster. With the star cascade detector, though, one must also decrease the trained root- and part-filter thresholds to get more detections. Doing so, however, defeats the computational advantage of the cascade and significantly increases detection time. We thus train a model for the star detector using the standard procedure on human-annotated training data, sample the top detections produced by this model with a decreased acceptance threshold, and train a model for the star cascade detector on these samples. This yields a model that is almost as fast as one trained by the star cascade detector on the original training samples but with the desired bias in acceptance threshold.

The Y1 corpus contains approximately 70 different object classes that play a role in the depicted events. Many of these, however, cannot be reliably detected with the Felzenszwalb et al. detectors that we use. We trained models for 25 object classes that can be reliably detected, as listed in Table 2. These object classes account for over 90% of the event participants. Person models were trained with approximately 2000 human-annotated positive samples from C-D1 while nonperson models were trained with approximately 1000 such samples. For each positive training sample, two negative training samples were randomly generated from the same frame constrained to not overlap substantially with the positive samples. We trained three distinct person models to account for body-posture variation and pool these when constructing person tracks. The detection scores were normalized for such pooled detections by a per-model offset computed as follows: A (50 bin) histogram was computed of the scores of the top detection in each frame of a video. The offset is then taken to be the minimum of the value that maximizes the between-class variance [Otsu, 1979] when bipartitioning this histogram and the trained acceptance threshold offset by a fixed, but small, amount (0.4).

We employed detection-based tracking for all 25 object models on all 749 videos in our test set. To prune the large number of tracks thus produced, we discard all tracks corresponding to certain object models on a per-video basis: those that exhibit high detection-score variance over the frames in that video as well as those whose detection-score distributions are neither unimodal nor bimodal. The parameters governing such pruning were determined solely on the training set. The tracks that remain after this pruning still account for over 90% of the event participants.

3.2 BODY-POSTURE CODEBOOK

We recognize events using a combination of the motion of the event participants and the changing body posture of the human participants. Body-posture information is derived using the part structure produced as a by-product of the Felzenszwalb et al. detectors. While such information is far noisier and less accurate than fitting precise articulated models [Andriluka et al., 2008, Bregler, 1997, Gavrila and Davis, 1995, Sigal et al., 2010, Yang and Ramanan, 2011] and appears unintelligible to the human eye, as shown in Section 3.3, it suffices to improve event-recognition accuracy. Such information can be extracted from a large unannotated corpus far more robustly than possible with precise articulated models.

Body-posture information is derived from part structure in two ways. First, we compute a vector of part displacements, each displacement as a vector from the detection center to the part center, normalizing these vectors to unit detection-box area. The time-series of feature vectors is augmented to include these part displacements and a finite-difference approximation of their temporal derivatives as continuous features for person detections. Second, we vector-quantize the part-displacement vector and include the codebook index as a discrete feature for person detections. Such pose features are included in the time-series on a per-frame basis. The codebook is trained by running each pose-specific person detector on the positive human-annotated samples used to train that detector and extract the resulting part-displacement vectors. We then pool the part-displacement vectors from the three pose-specific person models and employ hierarchical k-means clustering using Euclidean distance to derive a codebook of 49 clusters. Fig. 2 shows sample clusters from our codebook. Codebook indices are derived using Euclidean distance from the means of these clusters.

(a)	bag↦*bag*	car↦*car*	door↦*door*	person↦*person*	suv↦*SUV*
	bench↦*bench*	cardboard-box↦*box*	ladder↦*ladder*	person-crouch↦*person*	table↦*table*
	bicycle↦*bicycle*	cart↦*cart*	mailbox↦*mailbox*	person-down↦*person*	toy-truck↦*truck*
	big-ball↦*ball*	chair↦*chair*	microwave↦*microwave*	skateboard↦*skateboard*	tripod↦*tripod*
	cage↦*cage*	dog↦*dog*	motorcycle↦*motorcycle*	small-ball↦*ball*	truck↦*truck*
(b)	cardboard-box↦*cardboard*	person↦*upright*	person-crouch↦*crouched*	person-down↦*prone*	toy-truck↦*toy*
(c)	big-ball↦*big*	small-ball↦*small*			

Table 2: Trained models for object classes and their mappings to (a) nouns, (b) restrictive adjectives, and (c) size adjectives.

Figure 2: Sample clusters from our body-posture codebook.

3.3 EVENT CLASSIFICATION

Our tracker produces one or more tracks per object class for each video. We convert such tracks into a time-series of feature vectors. For each video, one track is taken to designate the agent and another track (if present) is taken to designate the patient. During training, we manually specify the track-to-role mapping. During testing, we automatically determine the track-to-role mapping by examining all possible such mappings and selecting the one with the highest likelihood [Siskind and Morris, 1996].

The feature vector encodes both the motion of the event participants and the changing body posture of the human participants. For each event participant in isolation we incorporate the following single-track features:

1. x and y coordinates of the detection-box center
2. detection-box aspect ratio and its temporal derivative
3. magnitude and direction of the velocity of the detection-box center
4. magnitude and direction of the acceleration of the detection-box center
5. normalized part displacements and their temporal derivatives
6. object class (the object detector yielding the detection)
7. root-filter index
8. body-posture codebook index

The last three features are discrete; the remainder are continuous. For each pair of event participants we incorporate the following track-pair features:

1. distance between the agent and patient detection-box centers and its temporal derivative
2. orientation of the vector from agent detection-box center to patient detection-box center

Our HMMs assume independent output distributions for each feature. Discrete features are modeled with discrete output distributions. Continuous features denoting linear quantities are modeled with univariate Gaussian output distributions, while those denoting angular quantities are modeled with von Mises output distributions.

For each of the 48 action classes, we train two HMMs on two different sets of time-series of feature vectors, one containing only single-track features for a single participant and the other containing single-track features for two participants along with the track-pair features. A training set of between 16 and 200 videos was selected manually from C-D1 for each of these 96 HMMs as positive examples depicting each of the 48 action classes. A given video could potentially be included in the training sets for both the one-track and two-track HMMs for the same action class and even for HMMs for different action classes, if the video was deemed to depict both action classes.

During testing, we generate present/absent judgments for each video in the test set paired with each of the 48 action classes. We do this by thresholding the likelihoods produced by the HMMs. By varying these thresholds, we can produce an ROC curve for each action class, comparing the resulting machine-generated present/absent judgments with the Amazon Mechanical Turk judgments. When doing so, we test videos for which our tracker produces two or more tracks against only the two-track HMMs while we test ones for which our tracker produces a single track against only the one-track HMMs.

We performed three experiments, training 96 different 200-state HMMs for each. Experiment I omitted all discrete features and all body-posture related features. Experiment II omitted only the discrete features. Experiment III omitted only the continuous body-posture related features. ROC curves for each experiment are shown in Fig. 3. Note that the incorporation of body-posture information, either in the form of continuous normalized part displacements or discrete codebook indices, improves event-recognition accuracy, despite the fact that the part displacements produced by the Felzenszwalb et al. detectors are noisy and appear unintelligible to the human eye.

3.4 GENERATING SENTENCES

We produce a sentence from a detected action class together with the associated tracks using the templates from Table 3. In these templates, words in *italics* denote fixed strings, words in **bold** indicate the action class, X and Y denote subject and object noun phrases, and the categories Adv, PP_{endo}, and PP_{exo} denote adverbs and prepositional-phrase adjuncts to describe the subject motion. The processes for generating these noun phrases, adverbs, and prepositional-phrase adjuncts are described below. One-track HMMs take that track to be the agent and thus the subject. For two-track HMMs we choose the mapping from tracks to roles that yields the higher likelihood and take the agent track to be the subject and the patient track to be the object except when the action class is either **approached** or **fled**, the agent is (mostly) stationary, and the patient moves more than the agent.

Brackets in the templates denote optional entities. Optional entities containing Y are generated only for two-track HMMs. The criteria for generating optional adverbs and prepositional phrases are described below. The optional entity for **received** is generated when there is a patient track whose category is `mailbox`, `person`, `person-crouch`, or `person-down`.

We use adverbs to describe the velocity of the subject. For some verbs, a velocity adverb would be awkward:

∗X ***slowly*** *had* Y	∗X *had* ***slowly*** Y

Furthermore, stylistic considerations dictate the syntactic position of an optional adverb:

X jumped ***slowly*** over Y	X ***slowly*** jumped over Y
X ***slowly*** *approached* Y	∗X *approached* ***slowly*** Y
?X ***slowly*** *fell*	X *fell* ***slowly***

The verb-phrase templates thus indicate whether an adverb is allowed, and if so whether it occurs, preferentially, preverbally or postverbally. Adverbs are chosen subject to three thresholds $v_1^{\text{action class}}$, $v_2^{\text{action class}}$, and $v_3^{\text{action class}}$ determined empirically on a per-action-class basis: We select those frames from the subject track where the magnitude of the velocity of the box-detection center is above $v_1^{\text{action class}}$. An optional adverb is generated by comparing the magnitude of the average velocity v of the subject track box-detection centers in these frames to the per-action-class thresholds:

quickly	$v > v_2^{\text{action class}}$
slowly	$v_1^{\text{action class}} \leq v \leq v_3^{\text{action class}}$

We use prepositional-phrase adjuncts to describe the motion direction of the subject. Again, for some verbs, such adjuncts would be awkward:

∗X *had* Y ***leftward***	∗X *had* Y ***from the left***

Moreover, for some verbs it is natural to describe the motion direction endogenously, from the perspective of the subject, while for others it is more natural to describe the motion direction exogenously, from the perspective of the viewer:

X *fell* ***leftward***	X *fell* ***from the left***
X *chased* Y ***leftward***	∗X *chased* Y ***from the left***
∗X *arrived* ***leftward***	X *arrived* ***from the left***

The verb-phrase templates thus indicate whether an adjunct is allowed, and if so whether it is preferentially endogenous or exogenous. The choice of adjunct is determined from the orientation of v, as computed above and depicted in Fig. 4(a,b). We omit the adjunct when $v < v_1^{\text{action class}}$.

We generate noun phrases X and Y to refer to event participants according to the following grammar:

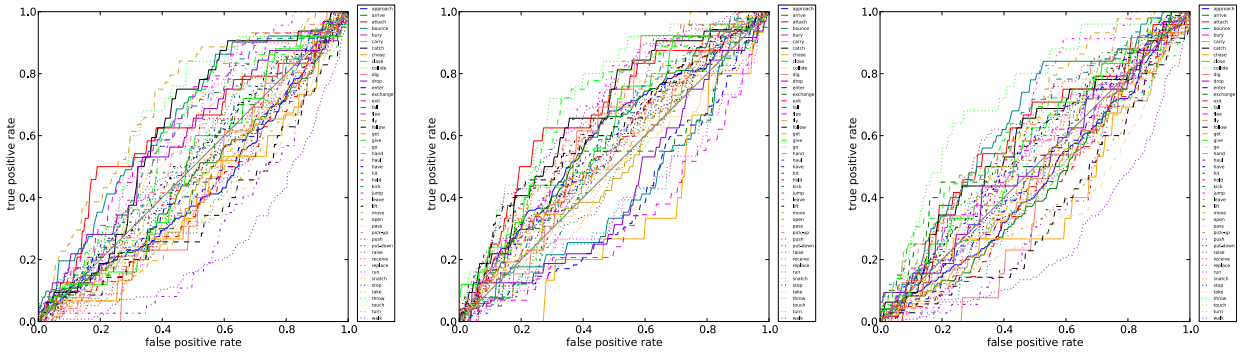
NP → *themselves* | *itself* | *something* | D A^* N [PP]
D → *the* | *that* | *some*

When instantiating a sentential template that has a required object noun-phrase Y for a one-track HMM, we generate a pronoun. A pronoun is also generated when the action class is **entered** or **exited** and the patient class is not `car`, `door`, `suv`, or `truck`. The anaphor *themselves* is generated if the action class is **attached** or **raised**, the anaphor *itself* if the action class is **moved**, and *something* otherwise.

As described below, we generate an optional prepositional phrase for the subject noun phrase to describe the spatial relation between the subject and the object. We choose the determiner to handle coreference, generating *the* when a noun phrase unambiguously refers to the agent or the patient due to the combination of head noun and any adjectives,

The *person jumped over* ***the*** *ball.*
The *red ball collided with* ***the*** *blue ball.*

that for an object noun phrase that corefers to a track referred to in a prepositional phrase for the subject,



(a) (b) (c)

Figure 3: ROC curves for each of the 48 action classes for (a) Experiment I, omitting all discrete and body-posture-related features, (b) Experiment II, omitting only the discrete features, and (c) Experiment III, omitting only the continuous body-posture-related features.

X [Adv] **approached** Y [PP_{exo}]	X [Adv] **entered** Y [PP_{endo}]	X **had** Y	X **put** Y **down**
X **arrived** [Adv] [PP_{exo}]	X [Adv] **exchanged** *an object with* Y	X **hit** [*something with*] Y	X **raised** Y
X [Adv] **attached** *an object to* Y	X [Adv] **exited** Y [PP_{endo}]	X **held** Y	X **received** [*an object from*]Y
X **bounced** [Adv] [PP_{endo}]	X **fell** [Adv] [*because of* Y] [PP_{endo}]	X **jumped** [Adv] [*over* Y] [PP_{endo}]	X [Adv] **replaced** Y
X **buried** Y	X **fled** [Adv] [*from* Y] [PP_{endo}]	X [Adv] **kicked** Y [PP_{endo}]	X **ran** [Adv] [*to* Y] [PP_{endo}]
X [Adv] **carried** Y [PP_{endo}]	X **flew** [Adv] [PP_{endo}]	X **left** [Adv] [PP_{endo}]	X [Adv] **snatched** *an object from* Y
X **caught** Y [PP_{exo}]	X [Adv] **followed** Y [PP_{endo}]	X [Adv] **lifted** Y	X [Adv] **stopped** [Y]
X [Adv] **chased** Y [PP_{endo}]	X **got** *an object from* Y	X [Adv] **moved** Y [PP_{endo}]	X [Adv] **took** *an object from* Y
X **closed** Y	X **gave** *an object to* Y	X **opened** Y	X [Adv] **threw** Y [PP_{endo}]
X [Adv] **collided** *with* Y [PP_{exo}]	X **went** [Adv] *away* [PP_{endo}]	X [Adv] **passed** Y [PP_{exo}]	X **touched** Y
X *was* **digging** [*with* Y]	X **handed** Y *an object*	X **picked** Y **up**	X **turned** [PP_{endo}]
X **dropped** Y	X [Adv] **hauled** Y [PP_{endo}]	X [Adv] **pushed** Y [PP_{endo}]	X **walked** [Adv] [*to* Y] [PP_{endo}]

Table 3: Sentential templates for the action classes indicated in bold.

> ***The*** *person to the right of* ***the*** *car approached* ***that*** *car.*
> ***Some*** *person to the right of* ***some*** *other person approached* ***that*** *other person.*

and *some* otherwise:

> ***Some*** *car approached* ***some*** *other car.*

We generate the head noun of a noun phrase from the object class using the mapping in Table 2(a). Four different kinds of adjectives are generated: color, shape, size, and restrictive modifiers. An optional color adjective is generated based on the average HSV values in the eroded detection boxes for a track: *black* when $V \leq 0.2$, *white* when $V \geq 0.8$, one of *red*, *blue*, *green*, *yellow*, *teal*, or *pink* based on H, when $S \geq 0.7$. An optional size adjective is generated in two ways, one from the object class using the mapping in Table 2(c), the other based on per-object-class image statistics. For each object class, a mean object size $\bar{a}_{\text{object class}}$ is determined by averaging the detected-box areas over all tracks for that object class in the training set used to train HMMs. An optional size adjective for a track is generated by comparing the average detected-box area a for that track to $\bar{a}_{\text{object class}}$:

big	$a \geq \beta_{\text{object class}} \bar{a}_{\text{object class}}$
small	$a \leq \alpha_{\text{object class}} \bar{a}_{\text{object class}}$

The per-object-class cutoff ratios $\alpha_{\text{object class}}$ and $\beta_{\text{object class}}$ are computed to equally tripartition the distribution of per-object-class mean object sizes on the training set. Optional shape adjectives are generated in a similar fashion. Per-object-class mean aspect ratios $\bar{r}_{\text{object class}}$ are determined in addition to the per-object-class mean object sizes $\bar{a}_{\text{object class}}$. Optional shape adjectives for a track are generated by comparing the average detected-box aspect ratio r and area a for that track to these means:

tall	$r \leq 0.7\bar{r}_{\text{object class}} \wedge a \geq \beta_{\text{object class}} \bar{a}_{\text{object class}}$
short	$r \geq 1.3\bar{r}_{\text{object class}} \wedge a \leq \alpha_{\text{object class}} \bar{a}_{\text{object class}}$
narrow	$r \leq 0.7\bar{r}_{\text{object class}} \wedge a \leq \alpha_{\text{object class}} \bar{a}_{\text{object class}}$
wide	$r \geq 1.3\bar{r}_{\text{object class}} \wedge a \geq \beta_{\text{object class}} \bar{a}_{\text{object class}}$

To avoid generating shape and size adjectives for unstable tracks, they are only generated when the detection-score variance and the detected aspect-ratio variance for the track are below specified thresholds. Optional restrictive modifiers are generated from the object class using the mapping in Table 2(b). Person-pose adjectives are generated from aggregate body-posture information for the track: object class, normalized part displacements, and body-posture codebook indices. We generate all applicable adjectives except for color and person pose. Following the Gricean Maxim of Quantity [Grice, 1975], we only generate

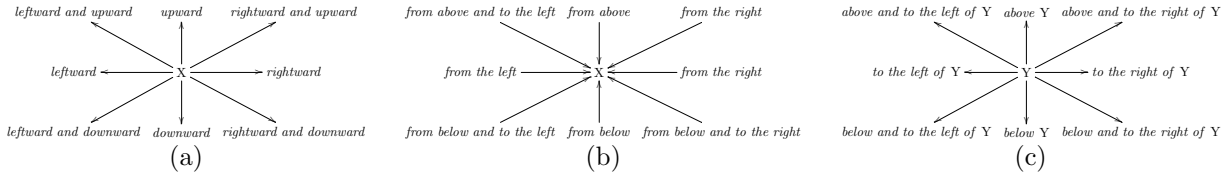


Figure 4: (a) Endogenous and (b) exogenous prepositional-phrase adjuncts to describe subject motion direction. (c) Prepositional phrases incorporated into subject noun phrases describing viewer-relative 2D spatial relations between the subject X and the reference object Y.

color and person-pose adjectives if needed to prevent coreference of nonhuman event participants. Finally, we generate an initial adjective *other*, as needed to prevent coreference. Generating *other* does not allow generation of the determiner *the* in place of *that* or *some*. We order any adjectives generated so that *other* comes first, followed by size, shape, color, and restrictive modifiers, in that order.

For two-track HMMs where neither participant moves, a prepositional phrase is generated for subject noun phrases to describe the static 2D spatial relation between the subject X and the reference object Y from the perspective of the viewer, as shown in Fig. 4(c).

4 EXPERIMENTAL RESULTS

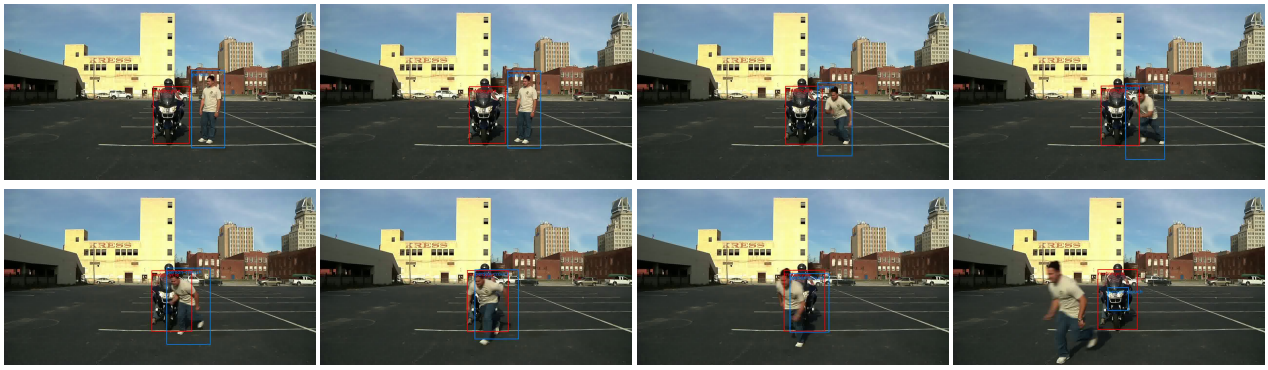
We used the HMMs generated for Experiment III to compute likelihoods for each video in our test set paired with each of the 48 action classes. For each video, we generated sentences corresponding to the three most-likely action classes. Fig. 5 shows key frames from four videos in our test set along with the sentence generated for the most-likely action class. Human judges rated each video-sentence pair to assess whether the sentence was true of the video and whether it described a salient event depicted in that video. 26.7% (601/2247) of the video-sentence pairs were deemed to be true and 7.9% (178/2247) of the video-sentence pairs were deemed to be salient. When restricting consideration to only the sentence corresponding to the single most-likely action class for each video, 25.5% (191/749) of the video-sentence pairs were deemed to be true and 8.4% (63/749) of the video-sentence pairs were deemed to be salient. Finally, for 49.4% (370/749) of the videos at least one of the three generated sentences was deemed true and for 18.4% (138/749) of the videos at least one of the three generated sentences was deemed salient.

5 CONCLUSION

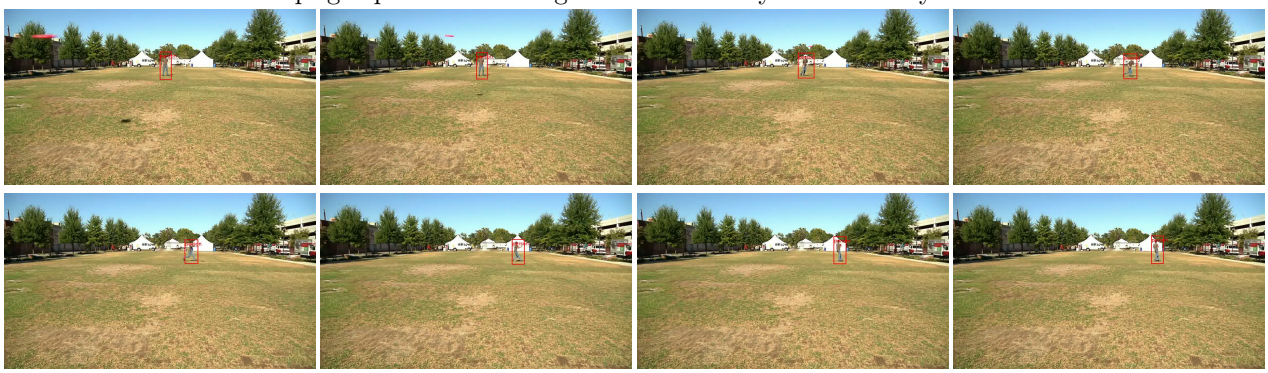
Integration of Language and Vision [Aloimonos et al., 2011, Barzialy et al., 2003, Darrell et al., 2011, McKevitt, 1994, 1995–1996] and recognition of action in video [Blank et al., 2005, Laptev et al., 2008, Liu et al., 2009, Rodriguez et al., 2008, Schuldt et al., 2004, Siskind and Morris, 1996, Starner et al., 1998, Wang and Mori, 2009, Xu et al., 2002, 2005] have been of considerable interest for a long time. There has also been work on generating sentential descriptions of static images [Farhadi et al., 2009, Kulkarni et al., 2011, Yao et al., 2010]. Yet we are unaware of any prior work that generates as rich sentential video descriptions as we describe here. Producing such rich descriptions requires determining event participants, the mapping of such participants to roles in the event, and their motion and properties. This is incompatible with common approaches to event recognition, such as spatiotemporal bags of words, spatiotemporal volumes, and tracked feature points that cannot determine such information. The approach presented here recovers the information needed to generate rich sentential descriptions by using detection-based tracking and a body-posture codebook. We demonstrated the efficacy of this approach on a corpus of 749 videos.

Acknowledgments

This work was supported, in part, by NSF grant CCF-0438806, by the Naval Research Laboratory under Contract Number N00173-10-1-G023, by the Army Research Laboratory accomplished under Cooperative Agreement Number W911NF-10-2-0060, and by computational resources provided by Information Technology at Purdue through its Rosen Center for Advanced Computing. Any views, opinions, findings, conclusions, or recommendations contained or expressed in this document or material are those of the author(s) and do not necessarily reflect or represent the views or official policies, either expressed or implied, of NSF, the Naval Research Laboratory, the Office of Naval Research, the Army Research Laboratory, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

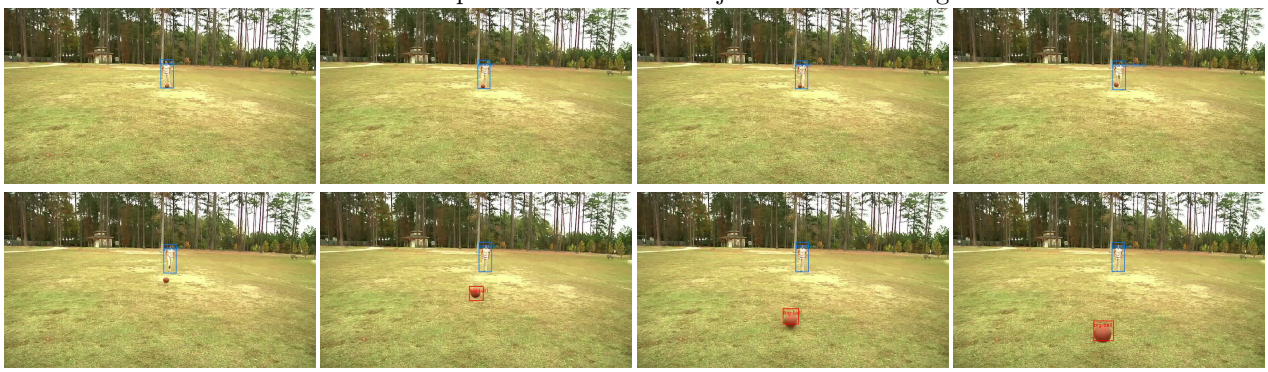


The upright person to the right of the motorcycle went away leftward.



The person walked slowly to something rightward.

The narrow person snatched an object from something.



The upright person hit the big ball.

Figure 5: Key frames from four videos in our test set along with the sentence generated for the most-likely action class.

References

Y. Aloimonos, L. Fadiga, G. Metta, and K. Pastra, editors. *AAAI Workshop on Language-Action Tools for Cognitive Artificial Agents: Integrating Vision, Action and Language*, 2011.

M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, pages 1–8, 2008.

R. Barzialy, E. Reiter, and J.M. Siskind, editors. *HLT-NAACL Workshop on Learning Word Meaning from Non-Linguistic Data*, 2003.

M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *ICCV*, pages 1395–402, 2005.

Christoph Bregler. Learning and recognizing human dynamics in video sequences. In *CVPR*, 1997.

T. Darrell, R. Mooney, and K. Saenko, editors. *NIPS Workshop on Integrating Language and Vision*, 2011.

Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR*, 2009.

P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010a.

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 32(9), 2010b.

D. M. Gavrila and L. S. Davis. Towards 3-d model-based tracking and recognition of human movement. In *International Workshop on Face and Gesture Recognition*, 1995.

H. P. Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics 3: Speech Acts*, pages 41–58. Academic Press, 1975.

Nazli Ikizler-Cinibis and Stan Sclaroff. Object, scene and actions: Combining multiple features for human action recognition. In *ECCV*, pages 494–507, 2010.

Ray Jackendoff. *Semantics and Cognition*. MIT Press, Cambridge, MA, 1983.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. Baby talk: Understanding and generating simple image descriptions. In *CVPR*, pages 1601–8, 2011.

I. Laptev, B. Caputo, C. Schuldt, and T. Lindeberg. Local velocity-adapted motion events for spatio-temporal recognition. *CVIU*, 108(3):207–29, 2007.

I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *CVPR*, pages 1996–2003, 2009.

P. McKevitt, editor. *AAAI Workshop on Integration of Natural Language and Vision Processing*, 1994.

P. McKevitt, editor. *Integration of Natural Language and Vision Processing*, volume I–IV. Kluwer, Dordrecht, 1995–1996.

J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *IJCV*, 79(3):299–318, 2008.

N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. on Systems, Man and Cybernetics*, 9(1):62–6, 1979. ISSN 0018-9472.

Steven Pinker. *Learnability and Cognition*. MIT Press, Cambridge, MA, 1989.

M. D. Rodriguez, J. Ahmed, and M. Shah. Action MACH: A spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008.

C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *ICPR*, pages 32–6, 2004. ISBN 0-7695-2128-2.

P. Scovanner, S. Ali, and M. Shah. A 3-dimensional SIFT descriptor and its application to action recognition. In *International Conference on Multimedia*, pages 357–60, 2007.

J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.

L. Sigal, A. Balan, and M. J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *IJCV*, 87(1-2):4–27, 2010.

J. M. Siskind and Q. Morris. A maximum-likelihood approach to visual event classification. In *ECCV*, pages 347–60, 1996.

Thad Starner, Joshua Weaver, and Alex Pentland. Real-time American sign language recognition using desk and wearable computer based video. *PAMI*, 20 (12):1371–5, 1998.

C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991.

A. J. Viterbi. Convolutional codes and their performance in communication systems. *IEEE Trans. on Communication*, 19:751–72, 1971.

Y. Wang and G. Mori. Human action recognition by semilatent topic models. *PAMI*, 31(10):1762–74, 2009. ISSN 0162-8828.

Gu Xu, Yu-Fei Ma, HongJiang Zhang, and Shiqiang Yang. Motion based event recognition using HMM. In *ICPR*, volume 2, 2002.

Gu Xu, Yu-Fei Ma, HongJiang Zhang, and Shi-Qiang Yang. An HMM-based framework for video semantic analysis. *IEEE Trans. Circuits Syst. Video Techn.*, 15(11):1422–33, 2005.

Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. In *CVPR*, 2011.

B. Z. Yao, Xiong Yang, Liang Lin, Mun Wai Lee, and Song-Chun Zhu. I2t: Image parsing to text description. *Proceedings of the IEEE*, 98(8):1485–1508, 2010.

Causal Inference by Surrogate Experiments: z-Identifiability

Elias Bareinboim and Judea Pearl
Cognitive Systems Laboratory
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA. 90095
{eb,judea} at cs.ucla.edu

Abstract

We address the problem of estimating the effect of intervening on a set of variables X from experiments on a different set, Z, that is more accessible to manipulation. This problem, which we call z-identifiability, reduces to ordinary identifiability when $Z = \emptyset$ and, like the latter, can be given syntactic characterization using the *do-calculus* [Pearl, 1995; 2000]. We provide a graphical necessary and sufficient condition for z-identifiability for arbitrary sets X, Z, and Y (the outcomes). We further develop a complete algorithm for computing the causal effect of X on Y using information provided by experiments on Z. Finally, we use our results to prove completeness of *do*-calculus relative to z-identifiability, a result that does not follow from completeness relative to ordinary identifiability.

1 Introduction

The relation between passive and experimental observations, and how they can aid the estimation of causal effects, is of central interest in the empirical sciences.

In this line of research, the *identification* problem ($I\mathcal{D}$, for short) asks whether causal effects can be computed from the joint distribution P over the observed variables, and theoretical knowledge encoded in the form of a causal diagram G.

This problem has been extensively studied in the literature, and [Pearl, 1995; 2000] gave it rigorous mathematical treatment based on the structural semantics, and introduced several graphical conditions such as the "back-door" and "front-door" criteria, which was later generalized by his *do-calculus*. In the last decades, a number of conditions had emerged for non-parametric identifiability such as the ones given by [Spirtes, Glymour, and Scheines, 1993; Galles and Pearl, 1995; Pearl and Robins, 1995; Halpern, 1998; Kuroki and Miyakawa, 1999]. In a series of breakthrough results starting with the development of the concept of C-component [Tian and Pearl, 2002], the *do-calculus* was finally shown to be complete [Huang and Valtorta, 2006; Shpitser and Pearl, 2006]. This result implies that there exists a finite sequence of applications of the rules of do-calculus that derives the target causal effect Q in terms of the observational distribution P if (and only if) Q is identifiable. The same work also provided algorithms that return a mapping from P to Q whenever Q is identifiable.

In real world applications, it is not uncommon that the quantity Q is unidentifiable, i.e., the distribution P together with the graph G are not able to unambiguously determine Q. A natural question arises whether the investigator could perform some auxiliary experiments (not necessary spelled out in Q), which would enable him/her to estimate the desired causal effects.

For instance, consider the causal diagram G in Fig. 1(a). Suppose one is interested in assessing the effect Q of cholesterol levels (X) on heart disease (Y), and data about subjects' diet (Z) is also collected. It is clear that Q is unidentifiable from the assumptions embodied in G, but it is infeasible in reality to control subjects' cholesterol level by intervention. Assume that an experiment can be conducted in which the subjects' diet (Z) is randomized; a natural question emerges whether Q is computable given this additional piece of experimental information?

Surprisingly, this ubiquitous problem has not received a thorough formal treatment. We introduce a variation of the $I\mathcal{D}$ problem to fill in this gap. Consider a setting in which, in addition to the information available in an ordinary $I\mathcal{D}$ instance (distribution P and graph G), further experiments can be performed over a set of variables Z; decide whether the target causal effects can be computed from the available information at hand. This extension generalizes the $I\mathcal{D}$ problem (when $Z = \emptyset$ the two problems coincide) and is called here the z-identification problem ($zI\mathcal{D}$, for short). The Z is called surrogate experiments, for obvious reasons.

Syntactically, the *zID* problem amounts to transforming $P(y|\hat{x})$[1] into an equivalent expressions in *do*-calculus such that only members of Z may contain the hat symbol. Applying this rationale for the example given above (Fig. 1(a)) entails the following reduction in the *do*-calculus. First apply Rule 3 to add $\hat{z}$,

$$P(y|\hat{x}) = P(y|\hat{x},\hat{z}) \quad \text{since } (Y \perp\!\!\!\perp Z|X)_{G_{\overline{X}\overline{Z}}}$$

Then apply Rule 2 to exchange $\hat{x}$ with x:

$$P(y|\hat{x},\hat{z}) = P(y|x,\hat{z}) \quad \text{since } (Y \perp\!\!\!\perp X|Z)_{G_{\underline{X}\overline{Z}}}$$

This last expression can be rewritten as,

$$P(y|x,\hat{z}) = \frac{P(y,x|\hat{z})}{P(x|\hat{z})} \tag{1}$$

This expression shows that performing an experiment on Z suffices to yield "identifiability" of the causal effect of X on Y without experimenting over X. [2]

The subtlety of this problem can be illustrated by noting that in the graph in Fig. 1(a) the effect is z-identifiable from $P(V)$ and $P(X,Y|\hat{Z})$ in G, whereas in the graph in Fig. 1(b) it is not (to be shown later). The only difference between these two graphs is the bidirected edge between the pairs (X,Z) and (X,Y).

One might surmise that *zID* can be represented by a mutilated graph in which the edges incoming to Z are cut, and the problem would then be solved as ordinary identifiability. Unfortunately, this is not the case as shown in the graph in Fig. 1(c) where $Q = P(y|\hat{x})$. The option of manipulating Z does not enable us to compute the Z-specific causal effect of X on Y, $P(y|\hat{x},z)$ which , if available, would allow us to compute the overall causal effect by averaging over Z. Although $Q' = P(y|\hat{x},\hat{z})$ can be established from the mutilated graph, it does not help in establishing the Z-specific causal effect, or Q.

The first formal treatment of this problem [Pearl, 1995] led to the following sufficient condition for admitting a surrogate variable Z for the causal effect $P(y|\hat{x})$:

(i) X intercepts all directed paths from Z to Y, and

(ii) $P(y|\hat{x})$ is identifiable in $G_{\overline{Z}}$.

These conditions are satisfied indeed in the model of Fig. 1(a) but not in 1(b) or 1(c). Pearl's criterion is sufficient but was not shown to be necessary. Additionally, it was not extended to the case where Z and X are sets of variables. At the same time, the syntactic condition above, which requires the existence of a do-calculus transformation expression containing only do(z) terms is declarative, but is not computationally effective, since it does not specify the sequence of rules leading to the needed transformation, nor does it tell us if such a sequence exists. Even though do-calculus is complete for identifying causal effects, it is not immediately clear whether it is complete for *zID*.

[1] We will use $P(y|\hat{x})$ interchangeably with $P_x(y)$ or $P(y|do(x))$. We also will call the interventional operator *do()* as the "hat" operator.

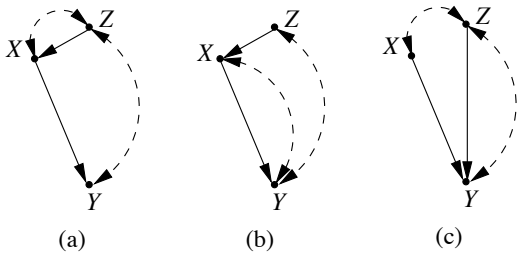


Figure 1: Causal diagrams illustrating z-identifiability of the causal effect $Q = P(y|\hat{x})$. Q can be identified by experiments on Z in model (a), but not in (b) and (c).

This paper provides a systematic study of z-identifiability building on Pearl's condition and the previous results from the identifiability literature; our contributions are as follows:

- We provide a necessary and sufficient graphical condition for the problem of z-identification when $\mathbf{Z}$ is a set of variables.
- We then construct a complete algorithm for deciding z-identification of joint causal effects and returning the correct formula whenever those effects are z-identifiable.
- We further show that *do*-calculus is complete for the task of z-identification.

2 Notation and Definitions

The basic semantical framework in our analysis rests on *probabilistic causal models* as defined in [Pearl, 2000, pp. 205], which are also called structural causal models or data-generating models. In the structural causal framework [Pearl, 2000, Ch. 7], actions are modifications of functional relationships, and each action $do(\mathbf{X} = \mathbf{x})$ on a causal model M produces a new model

[2] The expression also shows that only one level of Z suffices for the identification of $P(y|\hat{x})$ for any value of y and x. In other words, Z need not be varied at all; it can simply be held constant by external means and, if the assumptions embodied in G are valid, the r.h.s. of eq. (1) should attain the same value regardless of the (constant) level at which Z is being held constant. In practice, however, several levels of Z will be needed to ensure that enough samples are obtained for each desired value of X.

$M_{\mathbf{x}} = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}_{\mathbf{x}}, P(\mathbf{U}) \rangle$, where $F_{\mathbf{x}}$ is obtained after replacing $f_X \in \mathbf{F}$ for every $X \in \mathbf{X}$ with a new function that outputs a constant value x given by $do(\mathbf{X} = \mathbf{x})$.

We follow the conventions given in [Pearl, 2000]. We will denote variables by capital letters and their values by small letters. Similarly, sets of variables will be denoted by bold capital letters, sets of values by bold letters. We will use the typical graph-theoretic terminology with the corresponding abbreviations $Pa(\mathbf{Y})_G$, $An(\mathbf{Y})_G$, and $De(\mathbf{Y})_G$, which will denote respectively the set of observable parents, ancestors, and descendants of the node set $\mathbf{Y}$ in G. By convention, these sets will include the arguments as well, for instance, the ancestral set $An(\mathbf{Y})_G$ will include $\mathbf{Y}$. We will usually omit the graph subscript whenever the graph in question is assumed or obvious. A graph $G_{\mathbf{Y}}$ will denote the induced subgraph G containing nodes in $\mathbf{Y}$ and all arrows between such nodes. Finally, $G_{\overline{\mathbf{X}}\underline{\mathbf{Z}}}$ stands for the edge subgraph of G where all incoming arrows into $\mathbf{X}$ and all outgoing arrows from $\mathbf{Z}$ are removed.

We build on the problem of identifiability, defined below, which expresses the requirement that causal effects must be computable from a combination of passive data P and the assumptions embodied in a causal graph G (*without* assuming any availability of additional experimental information).

Definition 1 (Causal Effects Identifiability (Pearl))**.** *Let $\mathbf{X}, \mathbf{Y}$ be two sets of disjoint variables, and let G be the causal diagram. The causal effect of an action $do(\mathbf{X} = \mathbf{x})$ on a set of variables $\mathbf{Y}$ is said to be identifiable from P in G if $P_{\mathbf{x}}(\mathbf{y})$ is (uniquely) computable from $P(V)$ in any model that induces G.*

The following Lemma is the operational way to prove that a causal quantity is not identifiable given the assumptions embedded in G.

Lemma 1. *Let $\mathbf{X}, \mathbf{Y}$ be two sets of disjoint variables, and let G be the causal diagram. $P_{\mathbf{x}}(\mathbf{y})$ is not identifiable in G if there exist two causal models M^1 and M^2 compatible with G such that $P_1(\mathbf{V}) = P_2(\mathbf{V})$, and $P_1(\mathbf{y}|do(\mathbf{x})) \neq P_2(\mathbf{y}|do(\mathbf{x}))$.*

Proof. The latter inequality rules out the existence of a function from P to $P_{\mathbf{x}}(\mathbf{y})$. □

Next, we formally introduce the problem of z-identifiability that generalizes the problem of identifiability whereas it is no longer assumed that experimental information is not available at all, but there exists a set of variable $\mathbf{Z}$ in which experiments were performed and now is available for use. In other words, the explicit acknowledgement of the existence of the set $\mathbf{Z}$ adds a degree of freedom for the researcher, making the analysis more flexible and perhaps realistic.

Definition 2 (Causal Effects z-Identifiability)**.** *Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be disjoint sets of variables, and let G be the causal diagram. The causal effect of an action $do(\mathbf{X} = \mathbf{x})$ on a set of variables $\mathbf{Y}$ is said to be z-identifiable from P in G, if $P_{\mathbf{x}}(\mathbf{y})$ is (uniquely) computable from $P(\mathbf{V})$ together with the interventional distributions $P(\mathbf{V} \setminus \mathbf{Z}'|do(\mathbf{Z}'))$, for all $\mathbf{Z}' \subseteq \mathbf{Z}$, in any model that induces G.*

Armed with this new definition, we state next the sufficiency of the *do*-calculus for $zI\mathcal{D}$ that is analogous to [Pearl, 2000, Corol. 3.4.2] in respect to identification.

Theorem 1. *Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be disjoint sets of variables, let G be the causal diagram, and $Q = P(\mathbf{y}|do(\mathbf{x}))$. Q is $zI\mathcal{D}$ from P in G if the expression $P(\mathbf{y}|do(\mathbf{x}))$ is reducible, using the rules of do-calculus, to an expression in which only elements of $\mathbf{Z}$ may appear as interventional variables.*

Proof. The result follows from soundness of do-calculus and the definition of z-identifiability. □

It is clear that if we have an efficient procedure to establish $zI\mathcal{D}$, we can immediately decide $I\mathcal{D}$ by setting $\mathbf{Z} = \emptyset$. On the other hand, to be able to establish the converse of Theorem 1, we need to understand the conditions for non-$zI\mathcal{D}$, and so, we state next the analogous of Lemma 1 in this context.

Lemma 2. *Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be disjoint sets of variables, and let G be the causal diagram. $P_{\mathbf{x}}(\mathbf{y})$ is not z-identifiable in G if there exist two causal models M^1 and M^2 compatible with G such that $P^1(\mathbf{V}) = P^2(\mathbf{V})$, $P^1(\mathbf{V} \setminus \mathbf{Z}'|do(\mathbf{Z}')) = P^2(\mathbf{V} \setminus \mathbf{Z}'|do(\mathbf{Z}'))$, for all $\mathbf{Z}' \subseteq \mathbf{Z}$, and $P^1_{\mathbf{x}}(\mathbf{y}) \neq P^2_x(\mathbf{y})$.*

Proof. Let I be the set of interventional distributions $P(\mathbf{V} \setminus \mathbf{Z}'|do(\mathbf{Z}'))$, for any $\mathbf{Z}' \subseteq \mathbf{Z}$. The latter inequality rules out the existence of a function from P, I to $P_{\mathbf{x}}(\mathbf{y})$. □

While Lemma 2 might appear convoluted, it is nothing more than a formalization of the statement "Q cannot be computed from information set S alone." Naturally, when S has two components, $\langle P, I \rangle$, the Lemma becomes lengthy. Even though the problems of $I\mathcal{D}$ and $zI\mathcal{D}$ are related, Lemma 2 indicates that proofs of non-$zI\mathcal{D}$ are at least as hard as the ones for non-$I\mathcal{D}$, given that to prove the former requires the construction of two models to agree on $\langle P, I \rangle$, while to prove the latter it is only required for the two models to agree on the distribution P.

3 Characterizing $zI\mathcal{D}$ Relations

The concept of confounded component (or *C-component*) was introduced in [Tian and Pearl, 2002]

to represent clusters of variables connected through bidirected edges, and was instrumental in establishing a number of conditions for ordinary identification (Def. 1). If G is not a C-component itself, it can be uniquely partitioned into a set $\mathcal{C}(G)$ of C-components. We state below this definition that will also play a key role in the problem of $z\mathcal{ID}$. [3]

Definition 3 (C-component)**.** *Let G be a causal diagram such that a subset of its bidirected arcs forms a spanning tree over all vertices in G. Then G is a C-component (confounded component).*

A special subset of C-components that embraces the ancestral set of Y was noted by [Shpitser and Pearl, 2006] to play an important role in deciding identifiability – this observation can also be applied to z-identifiability, as formulated next.

Definition 4 (C-forest)**.** *Let G be a causal diagram, where $\mathbf{Y}$ is the maximal root set. Then G is a $\mathbf{Y}$-rooted C-forest if G is a C-component and all observable nodes have at most one child.*

We next introduce a structure based on C-forests that witnesses unidentifiability characterized by a pair of C-forests. $\mathcal{ID}$ was shown by [Shpitser and Pearl, 2006] infeasible if and only if such structure exists as an edge subgraph of the given causal diagram.

Definition 5 (hedge)**.** *Let $\mathbf{X}, \mathbf{Y}$ be set of variables in G. Let F, F' be $\mathbf{R}$-rooted C-forests such that $F \cap X \neq 0$, $F' \cap X = 0$, $F' \subseteq F$, $\mathbf{R} \subset An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$. Then F and F' form a hedge for $P_{\mathbf{x}}(\mathbf{Y})$ in G.*

The presence of this structure will prove to be an obstacle to z-identifiability of causal effects in various scenarios. For instance, the p-graph in Fig. 1(b) is a Y-rooted C-forest in which $P_x(y)$ will show not to be z-identifiable. However, different than in the $\mathcal{ID}$ case, there is no sharp boundary here, since Fig. 1(a) also contains a Y-rooted C-forest but $P_x(y)$ was already shown to be $z\mathcal{ID}$.

We formally show next that there is a variation of this structure that is able to capture non-$z\mathcal{ID}$ for a broad set of cases.

Theorem 2. *Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ be disjoint sets of variables and let G be the causal diagram. Then, the causal effects $Q = P_{\mathbf{x}}(\mathbf{y})$ is not $z\mathcal{ID}$ if there exists a hedge $\mathcal{F} = \langle F, F' \rangle$ for Q in $G_{\overline{Z}}$.*

Proof. The result is immediate. The existence of the hedge $\mathcal{F}$ for Q in $G_{\overline{\mathbf{Z}}}$ implies that $\mathbf{Z}$ cannot help in the (ordinary) identification of Q. Let us assume that Q is $z\mathcal{ID}$. Note that $\mathbf{Z}$ does not participate in the hedge $\mathcal{F}$ since there is no bidirected edge going towards any of its elements in $G_{\overline{\mathbf{Z}}}$, which is required by the definition of C-forest. Further, consider a parametrization such that all elements of $\mathbf{Z}$ are simply fair coins and disconnected from $\mathbf{V} \setminus \mathbf{Z}$ in G.

We can now use the same proof of non-$\mathcal{ID}$ based on $\mathcal{F}$ to prove non-$z\mathcal{ID}$ in G. The inequality of Q between the two models is obvious, and the agreement of the interventional distributions $do(\mathbf{Z})$ follows since $\mathbf{Z}$ is disconnected from $\mathbf{V} \setminus \mathbf{Z}$ by the chosen parametrization. This is a contradiction since $z\mathcal{ID}$ has to be valid for any parametrization compatible with G, which suffices to prove the result. □

[3] The advent of C-components complements the notion of *inducing path*, which was introduced earlier in [Verma and Pearl, 1990].

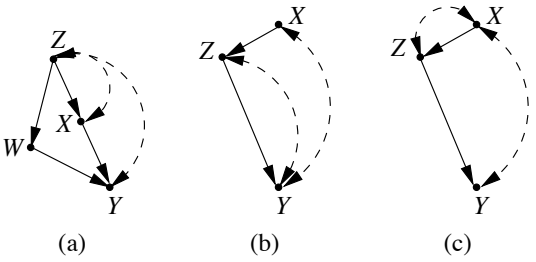


Figure 2: Graphs in which $P(y|\hat{x})$ is non-$z\mathcal{ID}$ from $do(Z)$ and there is no hedge in $G_{\overline{Z}}$.

Consider the next Corollary in regard to the p-graph, which is the smallest example in which Z could aid in the z-identification of Q but Q is still not z-identifiable from $do(Z)$. This and similar structures that prevent $z\mathcal{ID}$ will be one of the base cases for our proof of completeness, which requires a demonstration that whenever the algorithm fails to z-identify a causal relation, the relation is indeed non-$z\mathcal{ID}$.

Corollary 1. *$P_x(y)$ is not $z\mathcal{ID}$ in the p-graph.*

Proof. This follows directly from Theorem 2 since there exists a hedge in $G_{\overline{Z}}$. □

The result of Theorem 2 still does not characterize the $z\mathcal{ID}$ class, which suggests that the machinery used to prove completeness in the $\mathcal{ID}$ class is not immediately applicable to the $z\mathcal{ID}$ class.

For instance, consider the graph in Fig. 2(a) (called here bv-graph), which does not have a hedge for Q in $G_{\overline{Z}}$ but is still non-$z\mathcal{ID}$. The bv-graph coincides as an edge subgraph with Fig. 1(a) (note C-component induced over $\{X, Y, Z\}$), which turns out to be $z\mathcal{ID}$.

This is an interesting case, since up to this point, in ordinary identification, it was enough to locate a hedge for Q as an edge subgraph of the inputted diagram, and all graphs sharing this substructure were equally

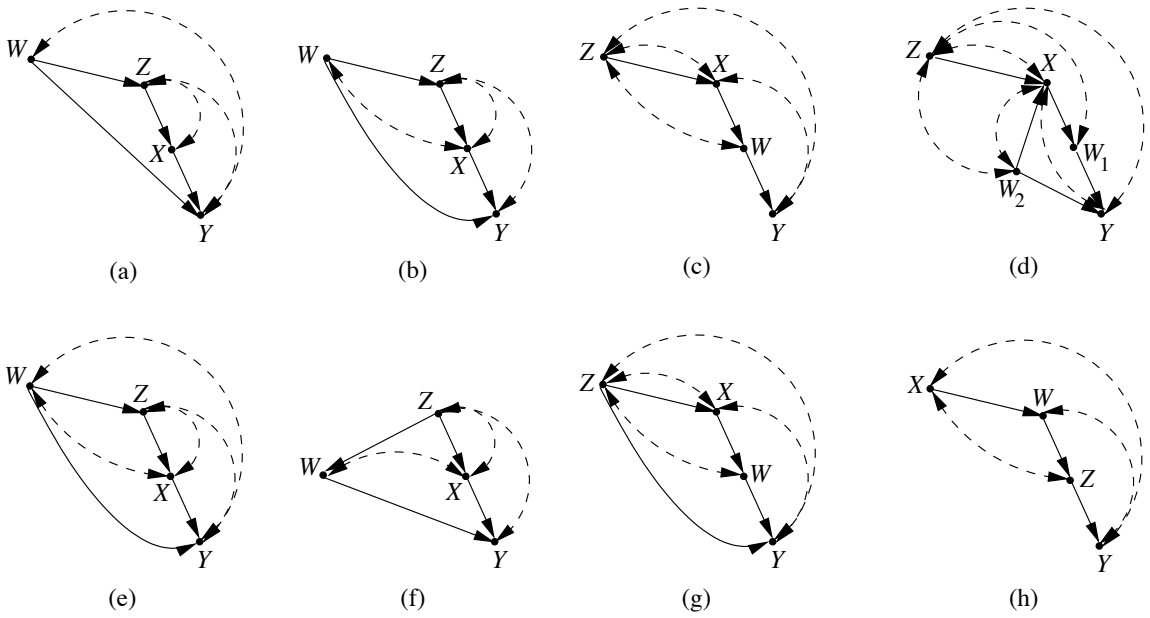


Figure 3: $P(y|\hat{x})$ is $zI\mathcal{D}$ from $\langle P, do(Z)\rangle$ in the graphs in the first row (a–d), but not in the the second row (e–h).

unidentifiable (see Thm. 4 in [Shpitser and Pearl, 2006]) – this is no longer true here since $\mathbf{Z}$ needs to be taken into account. Mainly, note that the directed edges outside a C-component play a very critical role for the $zI\mathcal{D}$ problem as the *bv*-graph demonstrates.

Finally, we expand Pearl's condition [Pearl, 2000, pp. 87] in the following directions. We extend, in the intuitive way, his condition to consider when $\mathbf{Z}$ is a set of variables and, in turn, we supplement the sufficient part with its necessary counterpart. We finally have a complete characterization for the $zI\mathcal{D}$ class as shown below.

Theorem 3. *Let* $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ *be disjoint sets of variables and let* G *be the causal diagram. The causal effect* $Q = P(\mathbf{y}|do(\mathbf{x}))$ *is* $zI\mathcal{D}$ *in* G *if and only if one of the following conditions hold:*

a. Q *is identifiable in* G*; or,*

b. There exists $\mathbf{Z}' \subseteq \mathbf{Z}$ *such that the following conditions hold,*

(i) $\mathbf{X}$ *intercepts all directed paths from* $\mathbf{Z}'$ *to* $\mathbf{Y}$*, and*

(ii) Q *is identifiable in* $G_{\overline{\mathbf{Z}'}}$.[4]

Proof. See Appendix. □

Let $Q = P(y|\hat{x})$ be the effect of interest and assume that experiments were performed over $\{Z\}$. Q is $zI\mathcal{D}$ from P and $do(Z)$ in the graphs in Fig. 3(a-d), while they are non-$zI\mathcal{D}$ in the graphs in Fig. 3(e-h). Except for the trivial case, Theorem 3 is existentially quantified and it is not immediately obvious how to efficiently select the covariates simultaneously satisfying both conditions of the Theorem. Clearly, a naive approach could lead to an exponential number of tests.

For example, consider the graph in Fig. 3(a) that is a variation of the *bv*-graph. In this graph, Q is $zI\mathcal{D}$ using experiments from $\{Z\}$. In turn, consider the graph in Fig. 3(e), which is the same as 3(a) but with the bidirected edge $W \leftarrow\rightarrow X$ added. Now, Q is no longer $zI\mathcal{D}$ for $\{Z\}$ nor $\{Z, W\}$. If we further consider the graph in Fig. 3(b) with the bidirected edge $W \leftarrow\rightarrow X$ removed from 3(e), not only Q becomes $zI\mathcal{D}$ for $\{Z\}$ but also for $\{Z, W\}$. This is a border case, note that if we input $\{Z, W\}$ as the surrogate variables for Pearl's criterion, it will not be able to recognize Q as $zI\mathcal{D}$ given the existence of the directed path $W \rightarrow Y$. Finally, if we consider the graph in Fig. 3(f) in which the directed edge $W \rightarrow Z$ is flipped from 3(b), Q is no longer $zI\mathcal{D}$ for neither $\{Z, W\}$ nor $\{Z\}$.

This example can be extended indefinitely but it is clear that finding a set that satisfies both conditions of the Theorem, in structures more intricate than the given 4-node example, does not follow immediately. The subject of the next section is about finding an efficient (and complete) algorithm to solve this problem.

But for now, consider the following Lemma that confirms our intuition that surrogate experiments should not disturb the causal paths (non-descendents) of the variables that are being analyzed.

[4]This condition can be rephrased graphically as "There exists no hedge for Q as an edge subgraph in $G_{\overline{\mathbf{Z}'}}$."

Corollary 2. *Let G be the causal diagram, $\mathbf{X}$, $\mathbf{Y} \subset \mathbf{V}$ be disjoint sets of variables, and $\mathbf{Z} \subseteq De(\mathbf{X})_{G_{An(\mathbf{Y})}}$. The causal effect $Q = P(\mathbf{y}|do(\mathbf{x}))$ is not zID from P and $do(\mathbf{Z})$ in G, if Q is not ID from P in G.*

Proof. The result follows directly from Theorem 3. □

4 A Complete Algorithm for *zID*

In this section, we propose a simple extension of the ordinary identification algorithms to solve the problem of z-identifiability, which we call $\mathbf{ID^z}$ (see Fig. 4).

We build on previous analysis of identifiability given in [Pearl, 1995; Kuroki and Miyakawa, 1999; Tian and Pearl, 2002; Shpitser and Pearl, 2006; Huang and Valtorta, 2006], and we choose to start with the version provided by Shpitser (called **ID**) since the hedge structure is explicitly employed, which will show to be instrumental to prove completeness.

Before considering the technical results, we explain our strategy and how our version of the algorithm relates to the existent ones for ordinary identifiability.

(i) z-identifiability (sufficiency): Causal relations can be solved in our context through ordinary identifiability or identifiability relying on the experiments performed over $\mathbf{Z}$. The current algorithms already operate on the first part, and they proceed exploring a sequence of equalities in do-calculus based on the C-component decomposition. (The idea is to apply a divide-and-conquer strategy breaking the problem into smaller, more manageable pieces, and then to assemble them back when it is possible.) It turns out that the equalities used by the algorithm are all in the interventional space (between interventional distributions except for the base cases), which is attractive for the *zID* problem since certain interventional distributions $\mathbf{Z}$ are already available to use.

For instance, when steps 3 or 4 succeed in their tests and, at the same time, have non-empty intersection with $\mathbf{Z}$, we exploit the common variables, updating the graph and respective data structures accordingly. We then continue solving an ordinary *ID* instance but no longer have to identify these variables and they possibly can help in the identifiability of others.

(ii) Non-z-identifiability (necessity): The algorithm proceeds until it is not able to resolve a certain subproblem, which implies the existence of a certain hedge. Note that the given hedge can be different than the one used for *ID* in the same graph since the experiments over $\mathbf{Z}$ possibly destroyed the original ones. Further, note that to use the given hedge to prove non-*zID* is not immediate since, in the light of Lemma 2, more constraints need to be satisfied in order to support such claim. Still, it is clear that if $\mathbf{Z}$ is not involved in the hedge, it can be shown that the two problems coincide. The other cases in which $\mathbf{Z}$ has non-empty intersection with the hedge have to be handled more carefully.

function $\mathbf{ID^z}(\mathbf{y}, \mathbf{x}, \mathbf{Z}, \mathcal{I}, \mathcal{J}, P, G)$
INPUT: $\mathbf{x}, \mathbf{y}$: value assignments; $\mathbf{Z}$: variables with interventions available; $\mathcal{I}, \mathcal{J}$: see caption; P: current probability distribution $do(\mathcal{I}, \mathcal{J}, x)$ (observational when $\mathcal{I} = \mathcal{J} = \emptyset$); G: causal graph.
OUTPUT: Expression for $P_{\mathbf{x}}(\mathbf{y})$ in terms of $P, P_{\mathbf{z}}$ or $\mathbf{FAIL}(F, F')$.

1 **if** $\mathbf{x} = \emptyset$, **return** $\sum_{\mathbf{v}\setminus\mathbf{y}} P(\mathbf{v})$.
2 **if** $\mathbf{V} \setminus An(\mathbf{Y})_G \neq \emptyset$,
return $\mathbf{ID^z}(\mathbf{y}, \mathbf{x} \cap An(\mathbf{Y})_G, \mathbf{Z}, \mathcal{I}, \mathcal{J}, \sum_{\mathbf{v}\setminus An(\mathbf{Y})_G} P, An(\mathbf{Y})_G)$.
3 Set $\mathbf{Z_w} = ((\mathbf{V} \setminus (\mathbf{X} \cup \mathcal{I} \cup \mathcal{J})) \setminus An(\mathbf{Y})_{G_{\overline{\mathbf{X}\cup\mathcal{I}\cup\mathcal{J}}}}) \cap \mathbf{Z}$.
Set $\mathbf{W} = ((\mathbf{V} \setminus (\mathbf{X} \cup \mathcal{I} \cup \mathcal{J})) \setminus An(\mathbf{Y})_{G_{\overline{\mathbf{X}\cup\mathcal{I}\cup\mathcal{J}}}}) \setminus \mathbf{Z}$.
if $(\mathbf{Z_w} \cup \mathbf{W}) \neq \emptyset$,
return $\mathbf{ID^z}(\mathbf{y}, \mathbf{x} \cup \mathbf{w}, \mathbf{Z} \setminus \mathbf{Z_w}, \mathcal{I} \cup \mathbf{z_w}, \mathcal{J}, P, G)$.
4 **if** $\mathcal{C}(G \setminus (\mathbf{X} \cup \mathcal{I} \cup \mathcal{J})) = \{S_0, S_1, ..., S_k\}$,
return $\sum_{\mathbf{v}\setminus\{\mathbf{y},\mathbf{x},\mathcal{I}\}} \prod_i \mathbf{ID^z}(s_i, (\mathbf{v} \setminus s_i) \setminus \mathbf{Z}, \mathbf{Z} \setminus (\mathbf{V} \setminus S_i), \mathcal{I}, \mathcal{J} \cup (\mathbf{Z} \cap (\mathbf{v} \setminus \mathbf{s_i})), P, G)$.
if $\mathcal{C}(G \setminus (\mathbf{X} \cup \mathcal{I} \cup \mathcal{J})) = \{S\}$,
5 **if** $\mathcal{C}(G) = \{G\}$, $\mathbf{FAIL}(G, S)$.
6 **if** $S \in \mathcal{C}(G)$,
return $\sum_{s\setminus\mathbf{y}} \prod_{i|V_i \in S} P(v_i|v_G^{(i-1)} \setminus (\mathcal{I} \cup \mathcal{J}))$.
7 **if** $(\exists S')S \subset S' \in \mathcal{C}(G)$,
return $\mathbf{ID^z}(\mathbf{y}, \mathbf{x} \cap S', \mathbf{Z}, \mathcal{I}, \mathcal{J}, \prod_{i|V_i \in C'} P(V_i|V_G^{(i-1)} \cap S', v_G^{(i-1)} \setminus (S' \cup \mathcal{I} \cup \mathcal{J})), S')$.

Figure 4: $\mathbf{ID^z}$: Modified version of *ID* algorithm capable of recognizing *zID*; The variables $\mathcal{I}, \mathcal{J}$ represent indices for currently active Z-interventions introduced respectively by steps 3 or 4. Note that P is sensitive to current instantiations of $\mathcal{I}, \mathcal{J}$.

Note that the key difference between $\mathbf{ID^z}$ and the original **ID** implementation is in steps 3 and 4 in which possibly some $\mathbf{Z}' \subseteq \mathbf{Z}$ is added as an interventional set, and kept as so until the end of the execution. It is clear that these additions just can represent a benefit in computing the target Q since is always easier to identify a quantity in a subgraph of the original input.

We prove next soundness and completeness of $\mathbf{ID^z}$.

Theorem 4 (soundness)**.** *Whenever $\mathbf{ID^z}$ returns an expression for $P_{\mathbf{x}}(\mathbf{y})$, it is correct.*

Proof. The result is immediate since the soundness of **ID** was already established [Shpitser and Pearl, 2006, Thm. 5], which is inherited by $\mathbf{ID^z}$ by construction. Note that adding $\mathbf{Z}' \subseteq \mathbf{Z}$ as an interventional set and

not trying to "identify" it later does not represent a problem, in the zID sense, since by assumption we can use the interventional distributions $do(\mathbf{Z})$ in the final expression returned by the procedure. □

Theorem 5. *Assume* $\mathbf{ID^z}$ *fails to z-identify* $P_{\mathbf{x}}(\mathbf{y})$ *from* P *and* $do(\mathbf{Z})$ *in* G *(executes line 5). Then there exists* $\mathbf{X}' \subseteq \mathbf{X}$, $\mathbf{Y}' \subseteq \mathbf{Y}$, $\mathbf{Z}', \mathbf{Z}'' \subseteq \mathbf{Z}$ *such that the graph pair* G, S *returned by the fail condition of* $\mathbf{ID^z}$ *contain as edge subgraphs C-forests* F, F' *that form a hedge for* $P_{\mathbf{x}',\mathbf{z}'}(\mathbf{y}', \mathbf{z}'')$.

Proof. This property is just partly inherited from the original **ID** since we can add $\mathbf{Z}' \subseteq \mathbf{Z}$ as interventional nodes along the execution of $\mathbf{ID^z}$; we also keep track of $\mathbf{Z}'' \subseteq \mathbf{Z}$ that are related to $An(\mathbf{Y})$ during the execution of the procedure (to be specified below).

Consider G, $\mathbf{Y_f}$, $\mathcal{I}$ and $\mathcal{J}$ local to the call in which $\mathbf{ID^z}$ exited with failure (line 5). It is true that the set $\mathbf{Y_f}$ is such that $\mathbf{Z}'' = \mathbf{Y_f} \cap \mathbf{Z}$ and $\mathbf{Y}' = \mathbf{Y_f} \cap \mathbf{Y}$. Let $\mathbf{Z}' \subseteq \mathbf{Z}$ be the active part of $\mathbf{Z}$ in the faulty call, which we kept track through $\mathcal{I} \cup \mathcal{J}$. The condition that triggered failure is that the whole graph was a single C-component. Let $\mathbf{R}$ be the root set of G. We can remove a set of directed arrows while keeping the root $\mathbf{R}$ such that the resulting F is an $\mathbf{R}$-rooted C-forest.

Similarly to **ID**, note that since $F' = F \cap S$ is closed under descendent and only single directed arrows were removed from S to obtain F', F' is also a C-forest. Now, $F' \cap (\mathbf{X} \cup \mathbf{Z}') = \emptyset$ and $F' \cap (\mathbf{X} \cup \mathbf{Z}') \neq \emptyset$, by construction. Also, $\mathbf{R} \subseteq An(\mathbf{Y}', \mathbf{Z}'')_{G_{\overline{\mathbf{X},\mathbf{Z}'}}}$ and $\mathbf{Z}'' \subseteq An(\mathbf{Y})_{G_{\overline{\mathbf{X},\mathbf{Z}'}}}$, by line 2 and 3 of the algorithm. □

Theorem 6 (completeness)**.** $\mathbf{ID^z}$ *is complete.*

Proof. By Theorem 5, $\mathbf{ID^z}$ failure implies the existence of $\mathbf{X}' \subseteq \mathbf{X}$, $\mathbf{Y}' \subseteq \mathbf{Y}$, $\mathbf{Z}', \mathbf{Z}'' \subseteq \mathbf{Z}$, and C-forests F, F' that form a hedge for $P_{\mathbf{x}',\mathbf{z}'}(\mathbf{y}', \mathbf{z}'')$. Let us proceed our analysis by cases:

Case $\mathbf{Z}' = \emptyset, \mathbf{Z}'' = \emptyset$. The construction provided by [Shpitser and Pearl, 2006, Corollary 2] can be used here since this case reduces to ordinary identifiability.

Case $\mathbf{Z}' = \emptyset, \mathbf{Z}'' \neq \emptyset$. Even though $\mathbf{Z}''$ is in the root set of the hedge, and not related to the interventional part $(F \setminus F')$ where the asymmetry in the construction usually resides (to generate inequality in Q), the previous construction have to be used with certain caution, as given by case 1 of Thm. 3.

There is an interesting border subcase when $\mathbf{Y}' = \emptyset$. We need to keep track of $\{\mathcal{I}, \mathcal{J}\}$ since if the Z-interventions are added in step 3, we should not be concerned with summing over the assignments of the variables added, but if the Z-interventions are added in step 4, we do have to take care of this case. Note that we would have some hedge in a do-equality in the form $Q = \sum_{\mathbf{z}''} P_{\mathbf{x}'}(\mathbf{z}'') f(\mathbf{x}, \mathbf{y}, ...)$, in which if $f(.)$ is identifiable and uniformly distributed, Q would equate in both models and spoil the counter-example. The problem is not difficult to fix, and we just have to create a map for $f()$ that is non-uniform. (See Thm. 3.)

Case $\mathbf{Z}' \neq \emptyset, \mathbf{Z}'' = \emptyset$. The construction provided in cases 2 and 3 of Thm. 3 were more involved since it was not know a priori which C-factor yielded the "faulty" call. In the $\mathbf{ID^z}$ case, we already located the hedge based on the trace of the algorithm, then we can essentially use the same construction of these cases to provide a counterexample.

Case $\mathbf{Z}' \neq \emptyset, \mathbf{Z}'' \neq \emptyset$. The construction provided in the two previous cases are not incompatible, and they can be combined to provide a counter-example to this scenario.

Moreover, the previous constructions were given over the subgraph H of G, and how to extend the counter-example to G is discussed in Theorem 3. □

Corollary 3. *The rules of do-calculus, together with standard probability manipulations are complete for determining z-identifiability of* $P_{\mathbf{x}}(\mathbf{y})$.

Proof. It was already shown [Shpitser and Pearl, 2006, Thm. 7] that the operations of **ID** correspond to sequences of standard probability manipulations and application of the rules of do-calculus, which is also true by construction for $\mathbf{ID^z}$, and so the result follows. □

Conclusion

This paper was concerned with a variation of the identifiability problem in which experiments can be conducted over a subset of the variables $\mathbf{Z}$ in addition to the assumptions embodied in a causal digram G and the statistical knowledge given as a probability distribution. (If $\mathbf{Z}$ is an empty set, the two problems coincide.)

We provide a graphical necessary and sufficient condition for the cases when the causal effect of an arbitrary set of variables on another arbitrary set can be determined uniquely from the available information. We further provide a complete algorithm for computing the resulting mapping, that is, a formula fusing available observational and experimental data to synthesize an estimate of the desired causal effects. Furthermore, we use our results to prove completeness of *do*-calculus in respect to the z-identifiability class.

Our results were developed in a non-parametric setting in the tradition of the do-calculus. For a future

research direction, it would be interesting to explore how experimental data can aid the identification in the linear case. This is a harder problem, since a complete characterization of ordinary identifiability (i.e., $\mathbf{Z} = \emptyset$) in the linear case is still an open problem.

This paper complements two recent works on generalizability of causal and statistical knowledge. The first, dubbed "transportability" [Pearl and Bareinboim, 2011; Bareinboim and Pearl, 2012b] , deals with transferring causal information from an experimental to an observational environment, potentially different from the first. The second, called "selection bias" [Bareinboim and Pearl, 2012c], deals with extrapolation between an environment in which samples are selected preferentially and one in which no preferential sampling takes place. The extrapolation involved in z-Identification problems takes place between two different regimes; one in which experiments are performed over $\mathbf{Z}$, and one in which future experiments are anticipated over $\mathbf{X}$. Extensions to "meta synthesis" tasks, where information from multiple heterogeneous sources are combined to increase the effective sample size, are considered in [Pearl, 2012b; 2012a].

Acknowledgement

The authors would like to thank the reviewers for their comments that help improve the manuscript. This research was supported in parts by grants from NIH #1R01 LM009961-01, NSF #IIS-0914211 and #IIS-1018922, and ONR #N000-14-09-1-0665 and #N00014-10-1-0933.

References

[Bareinboim and Pearl, 2012a] Bareinboim, E., and Pearl, J. 2012a. Causal inference by surrogate experiments: z-identifiability. Technical Report Technical Report r-397, Cognitive Systems Laboratory, Department of Computer Science, UCLA.

[Bareinboim and Pearl, 2012b] Bareinboim, E., and Pearl, J. 2012b. Transportability of causal effects: Completeness results. In *Proceedings of the Twenty-Sixth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press. (to appear).

[Bareinboim and Pearl, 2012c] Bareinboim, E., and Pearl, J. 2012c. Controlling selection bias in causal inference. In Girolami, M., and Lawrence, N., eds., *Proceedings of The Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2012), JMLR (22)*, 100–108.

[Galles and Pearl, 1995] Galles, D., and Pearl, J. 1995. Testing identifiability of causal effects. In Besnard, P., and Hanks, S., eds., *Uncertainty in Artificial Intelligence 11*. San Francisco: Morgan Kaufmann. 185–195.

[Halpern, 1998] Halpern, J. 1998. Axiomatizing causal reasoning. In Cooper, G., and Moral, S., eds., *Uncertainty in Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann. 202–210. Also, *Journal of Artificial Intelligence Research* 12:3, 17–37, 2000.

[Huang and Valtorta, 2006] Huang, Y., and Valtorta, M. 2006. Identifiability in causal bayesian networks: A sound and complete algorithm. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press. 1149–1156.

[Kuroki and Miyakawa, 1999] Kuroki, M., and Miyakawa, M. 1999. Identifiability criteria for causal effects of joint interventions. *Journal of the Royal Statistical Society* 29:105–117.

[Pearl and Bareinboim, 2011] Pearl, J., and Bareinboim, E. 2011. Transportability of causal and statistical relations: A formal approach. In *Proceedings of the Twenty-Fifth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press. 247–254.

[Pearl and Robins, 1995] Pearl, J., and Robins, J. 1995. Probabilistic evaluation of sequential plans from causal models with hidden variables. In Besnard, P., and Hanks, S., eds., *Uncertainty in Artificial Intelligence 11*. San Francisco: Morgan Kaufmann. 444–453.

[Pearl, 1995] Pearl, J. 1995. Causal diagrams for empirical research. *Biometrika* 82(4):669–710.

[Pearl, 2000] Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. New York: Cambridge University Press. Second ed., 2009.

[Pearl, 2012a] Pearl, J. 2012a. The latent power of do-calculus. In *this proceedings*.

[Pearl, 2012b] Pearl, J. 2012b. Some thoughts concerning transfer learning with applications to meta-analysis and data sharing estimation. Technical Report Technical Report r-387, Cognitive Systems Laboratory, Department of Computer Science, UCLA.

[Shpitser and Pearl, 2006] Shpitser, I., and Pearl, J. 2006. Identification of joint interventional distributions in recursive semi-Markovian causal models. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press. 1219–1226.

[Spirtes, Glymour, and Scheines, 1993] Spirtes, P.; Glymour, C.; and Scheines, R. 1993. *Causation, Prediction, and Search*. New York: Springer-Verlag.

[Tian and Pearl, 2002] Tian, J., and Pearl, J. 2002. A general identification condition for causal effects. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press/The MIT Press. 567–573.

[Verma and Pearl, 1990] Verma, T., and Pearl, J. 1990. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence (UAI-90)*, 220–227. Also in P. Bonissone, M. Henrion, L.N. Kanal and J.F. Lemmer (Eds.), *Uncertainty in Artificial Intelligence 6*, Elsevier Science Publishers, B.V., 255–268, 1991.

An Efficient Message-Passing Algorithm for the M-Best MAP Problem

Dhruv Batra
TTI-Chicago
dbatra@ttic.edu

Abstract

Much effort has been directed at algorithms for obtaining the highest probability configuration in a probabilistic random field model – known as the maximum a posteriori (MAP) inference problem. In many situations, one could benefit from having not just a single solution, but the top M most probable solutions – known as the M-Best MAP problem.

In this paper, we propose an efficient message-passing based algorithm for solving the M-Best MAP problem. Specifically, our algorithm solves the recently proposed Linear Programming (LP) formulation of M-Best MAP [7], while being *orders of magnitude* faster than a generic LP-solver. Our approach relies on studying a particular partial Lagrangian relaxation of the M-Best MAP LP which exposes a natural combinatorial structure of the problem that we exploit.

1 Introduction

A large number of problems in computer vision, natural language processing and computational biology can be formulated as the search for the most probable state under a discrete probabilistic graphical model – known the MAP inference problem.

In a number of such applications, one can benefit from having not just a single best solution, rather a list of M-best hypotheses. For example, sentences are often ambiguous and machine translation systems benefit from working with multiple plausible parses of a sentence. In computational biology, practitioners are often interested in computing the top M most stable configurations of a protein structure. Moreover, computing such a set of M-best hypotheses is useful for assessing the sensitivity of the model w.r.t. variations in the input and/or the parameters of the model.

In the graphical models literature, this problem is known as the M-Best MAP problem [7, 18, 28]. Interestingly (and perhaps understandably), algorithms for the M-Best MAP problem have closely followed the development of the algorithms for solving the MAP problem. Similar to MAP, the first family of algorithms for M-Best MAP [18,20] were junction-tree based exact algorithms, feasible only for low-treewidth graphs. For high-treewidth models, where Belief Propagation (BP) is typically used to perform approximate MAP inference, Yanover and Weiss [28] showed how the pseudo-max-marginals produced by BP may be used to compute approximate M-Best MAPs.

However, with the development of Linear Programming (LP) relaxations for MAP, this concurrence between MAP and M-Best MAP is no longer true. While message-passing algorithms for solving the MAP LP were available as soon as the LPs were studied [8, 13, 19, 24, 26], no such algorithm is known for solving the M-Best MAP LP [7]. This discrepancy is not merely a theoretical concern – large-scale empirical comparisons [27] have found that message-passing algorithms for MAP significantly outperform commercial LP solvers (like CPLEX). More importantly, message-passing algorithms can be applied to large-scale problems where solvers like CPLEX simply would not scale. Thus, if we are to apply M-Best MAP to real instances appearing in computational biology, computer vision and NLP, we must develop scalable distributed message-passing algorithms.

Overview. The principal contribution of this paper is to develop an efficient message-passing algorithm for the M-Best MAP problem in discrete undirected graphical models, specifically Markov Random Fields (MRFs). Our approach studies a particular partial Lagrangian relaxation of the M-Best MAP LP [7] which exposes the natural modular structure in the problem. For graphs with cycles, this Lagrangian relaxation involves an exponentially large set of dual variables, and we use a dynamic subgradient method (DSM) [4] for solving this Lagrangian dual. DSMs are a recently formalized class of methods that interleave a separation

oracle procedure (that selects a subset of active dual variables), with the dual update procedure (that takes a step in the direction of the subgradient).

At a high-level, our algorithm brings MAP and M-Best MAP to an equal footing vis-a-vis a message-passing algorithm for solving the corresponding LP-relaxation. Importantly, our algorithm retains all the guarantees of the LP formulation of Fromer and Globerson [7], while being *orders of magnitude* faster. Similar to the observations in [27] for MAP, we find that our algorithm enables solving M-Best MAP on large instances that were unsolvable with generic LP solvers.

Outline. We begin with a brief history of the M-Best MAP problem in Section 2; present preliminaries and background in Section 3; revisit the M-Best MAP LP formulation of Fromer and Globerson [7] in Section 4; study the Lagrangian dual of this LP and present a message-passing dual ascent algorithm for tree-MRF in Section 5 and for general MRFs in Section 6.

2 Previous Work on M-Best MAP

The problem of finding the top M solutions to a general combinatorial optimization problem (not just inference in MRFs) has typically been studied in the context of k-shortest paths [6] in a search graph. Lawler [16] proposed a general algorithm to compute the top M solutions for a large family of discrete optimization problems, and the ideas used in Lawler's algorithm form the basis of most algorithms for the M-Best MAP problem. If the complexity of finding the best solution is $T(n)$, where n is the number of variables, Lawler's algorithm solves n new problems. The best solution among these n problems is the second best solution to the original problem. Thus at an $O(nM)$ multiplicative overhead, the top M solutions can be iteratively found. Hamacher and Queyranne [10] reduced this overhead to $O(M)$ by assuming access to an algorithm that can compute the first and the second best solutions.

Dechter and colleagues [5, 17] have recently provided dynamic-programming algorithms for M-Best MAP, but these are exponential in treewidth. Yanover and Weiss [28] proposed an algorithm that requires access only to max-marginals. Thus, for certain classes of MRFs that allow efficient exact computation of max-marginals, *e.g.* binary pairwise supermodular MRFs [11], M-Best solutions can be found for arbitrary treewidth graphs. Moreover, *approximate* M-Best solutions may be found by approximating the max-marginal computation, *e.g.* via loopy BP.

More recently, Fromer and Globerson [7] provided a LP view of the M-Best MAP problem. They proposed an algorithm (STRIPES) that repeatedly partitions the space of solutions and solves a 2^{nd}-Best MAP LP within each partition (with a generic LP-solver). We revisit [7] in detail in Section 4, and describe a significantly more efficient message-passing algorithm for solving the LP in the remainder of the paper.

3 Preliminaries: MAP-MRF Inference

Notation. For any positive integer n, let $[n]$ be shorthand for the set $\{1, 2, \ldots, n\}$. We consider a set of discrete random variables $\mathbf{x} = \{x_i \mid i \in [n]\}$, each taking value in a finite label set, $x_i \in X_i$. For a set $A \subseteq [n]$, we use x_A to denote the tuple $\{x_i \mid i \in A\}$, and X_A to be the cartesian product of the individual label spaces $\times_{i \in A} X_i$. For ease of notation, we use x_{ij} as a shorthand for $x_{\{i,j\}}$. For two vector $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$, we use $\boldsymbol{a} \cdot \boldsymbol{b}$ to denote the inner product.

MAP. Let $G = (\mathcal{V}, \mathcal{E})$ be a graph defined over these variables, *i.e.* $\mathcal{V} = [n]$, $\mathcal{E} \subseteq \binom{\mathcal{V}}{2}$, and let $\theta_A : X_A \to \mathbb{R}$, $(\forall A \in \mathcal{V} \cup \mathcal{E})$ be functions defining the energy at each node and edge for the labeling of variables in scope. The goal of MAP inference is to find the labeling $\mathbf{x}$ of the variables that minimizes this real-valued energy function:

$$\min_{\mathbf{x} \in X_{\mathcal{V}}} \sum_{A \in \mathcal{V} \cup \mathcal{E}} \theta_A(x_A) \tag{1a}$$

$$= \min_{\mathbf{x} \in X_{\mathcal{V}}} \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j). \tag{1b}$$

The techniques developed in this paper are naturally applicable to higher-order MRFs as well. However, to simplify the exposition we restrict ourselves to pairwise energy functions.

MAP Integer Program. MAP inference is typically set up as an integer programming problem over boolean variables. For each node and edge $A \in \mathcal{V} \cup \mathcal{E}$, let $\boldsymbol{\mu}_A = \{\mu_A(s) \mid s \in X_A,\ \mu_A(s) \in \{0, 1\}\}$, be a vector of indicator variables encoding all possible configurations of x_A. If $\mu_A(s)$ is set to 1, this implies that x_A takes label s. Moreover, let $\boldsymbol{\theta}_A = \{\theta_A(s) \mid s \in X_A\}$ be a vector holding energies for all possible configurations of x_A, and $\boldsymbol{\mu} = \{\boldsymbol{\mu}_A \mid A \in \mathcal{V} \cup \mathcal{E}\}$ be a vector holding the entire configuration. Using this notation, the MAP inference integer program can be written as:

$$\min_{\boldsymbol{\mu}_i, \boldsymbol{\mu}_{ij}} \quad \sum_{i \in \mathcal{V}} \boldsymbol{\theta}_i \cdot \boldsymbol{\mu}_i + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{\theta}_{ij} \cdot \boldsymbol{\mu}_{ij} \tag{2a}$$

$$s.t. \quad \sum_{s \in X_i} \mu_i(s) = 1 \qquad \forall i \in \mathcal{V} \tag{2b}$$

$$\sum_{s \in X_i} \mu_{ij}(s, t) = \mu_j(t) \qquad \forall \{i, j\} \in \mathcal{E} \tag{2c}$$

$$\sum_{t \in X_j} \mu_{ij}(s, t) = \mu_i(s) \qquad \forall \{i, j\} \in \mathcal{E} \tag{2d}$$

$$\mu_i(s), \mu_{ij}(s, t) \in \{0, 1\} \quad \forall i \in \mathcal{V}, \forall \{i, j\} \in \mathcal{E} \tag{2e}$$

Here (2b) enforces that exactly one label is assigned to a variable, and (2c),(2d) that assignments are consistent across edges. To be concise, we will use $\mathcal{P}(G)$ to denote the set of $\boldsymbol{\mu}$ that satisfy the three constraints (2b),(2c),(2d). Thus,

the above problem (2) can be written concisely as: $\min_{\boldsymbol{\mu}\in\mathcal{P}(G),\ \mu_A(s)\in\{0,1\}} \sum_{A\in\mathcal{V}\cup\mathcal{E}} \boldsymbol{\theta}_A\cdot\boldsymbol{\mu}_A$.

MAP LP. Problem (2) is known to be NP-hard in general [21]. A number of techniques [8, 13, 19, 24, 26] solve a Linear Programming (LP) relaxation of this problem, which is given by relaxing the boolean constraints (2e) to the unit interval, *i.e.* $\mu_i(s), \mu_{ij}(s,t) \geq 0$. Thus, the MAP LP minimizes the energy over the following polytope: $\mathcal{L}(G) = \{\mu_A(\cdot) \geq 0, \mid \boldsymbol{\mu} \in \mathcal{P}(G)\}$, also known as the *local polytope*. The LP relaxation of MAP is known to be *tight* for special cases like tree-graphs and binary submodular energies [25], meaning that the optimal vertex of the local polytope is an integer $\boldsymbol{\mu}_A \in \{0,1\}$, for these special cases.

4 M-Best MAP Linear Program

Let us now revisit the M-Best MAP LP formulation [7]. Let $\boldsymbol{\mu}^m$ denote the m^{th}-best MAP. Thus $\boldsymbol{\mu}^1$ is the MAP, $\boldsymbol{\mu}^2$ is the second-best MAP and so on. The M-Best MAP integer program is given by

$$\boldsymbol{\mu}^M = \operatorname*{argmin}_{\boldsymbol{\mu}\in\mathcal{P}(G),\ \mu_A(s)\in\{0,1\}} \sum_{A\in\mathcal{V}\cup\mathcal{E}} \boldsymbol{\theta}_A\cdot\boldsymbol{\mu}_A \tag{3a}$$

$$s.t. \qquad \boldsymbol{\mu} \neq \boldsymbol{\mu}^m \quad \forall m \in [M-1] \tag{3b}$$

While the MAP integer program suggested a natural LP relaxation, that is not the case with the M-Best MAP integer program due to the exclusion constraints (3b), which are not linear constraints. Fromer and Globerson [7] introduced a concise representation of a polytope called *assignment-excluding local polytope* $\mathcal{L}(G, \{\boldsymbol{\mu}^m\}_1^{M-1})$ that excludes previous solutions $\{\boldsymbol{\mu}^m \mid m \in [M-1]\}$ with the help of additional linear inequalities, called the *spanning-tree inequalities*.

Spanning Tree Inequalities. Let $T \subseteq \mathcal{E}$ be a spanning tree of G, and $\mathcal{T}(G)$ be the set of all such spanning trees in G. Let d_i^T be the degree of node i in T. Let us define $I^T(\boldsymbol{\mu}, \boldsymbol{\mu}^m) \triangleq \sum_{i\in\mathcal{V}} (1-d_i^T)\,\boldsymbol{\mu}_i\cdot\boldsymbol{\mu}_i^m + \sum_{(i,j)\in T} \boldsymbol{\mu}_{ij}\cdot\boldsymbol{\mu}_{ij}^m$.

Now, a spanning tree inequality is defined as: $I^T(\boldsymbol{\mu}, \boldsymbol{\mu}^m) \leq 0$.

Notice that $I^T(\boldsymbol{\mu}^m, \boldsymbol{\mu}^m) = 1$. Moreover, it can be shown [7] that $I^T(\boldsymbol{\mu}, \boldsymbol{\mu}^m) \leq 0,\ \forall \boldsymbol{\mu} \neq \boldsymbol{\mu}^m$. Thus, the spanning tree inequality *separates* the vertex $\boldsymbol{\mu}^m$ from other vertices in the polytope.

Assignment-Excluding Local Polytope (AELP) is defined as:

$$\mathcal{L}(G, \{\boldsymbol{\mu}^m\}_1^{M-1}) = \Big\{ \boldsymbol{\mu} \;\Big|\; \boldsymbol{\mu} \in \mathcal{L}(G),\ I^T(\boldsymbol{\mu}, \boldsymbol{\mu}^m) \leq 0, \quad \forall T \in \mathcal{T}(G), \forall m \in [M-1] \Big\}. \tag{4}$$

Thus, we can see that the AELP excludes each of the previous solutions $\{\boldsymbol{\mu}^m \mid m \in [M-1]\}$ with the help of spanning tree inequalities.

Recall that the LP relaxation over the local polytope for a tree-structured MRFs is tight. It can also be shown [7] that the LP relaxation over AELP for tree MRFs is tight for $m = 2$. However, for $m \geq 3$ in tree MRFs and any for any $m \geq 1$ in loopy MRFs, the AELP is not guaranteed to be a tight relaxation.

Efficient Separation Oracle. Note that for tree MRFs, there is a single spanning tree inequality for each previous solution since $|\mathcal{T}(G)| = 1$, while for general graphs there may be an exponentially large collection of spanning trees, *e.g.* $|\mathcal{T}(G)| = n^{n-2}$ for complete graphs. However, not all such inequalities need to be explicitly included. We can use a cutting-plane algorithm that maintains a working set of spanning trees $\mathcal{T}'$ and incrementally adds the most violated inequality: $\mathcal{T}' \longleftarrow \mathcal{T}' \cup \operatorname{argmax}_{T\in\mathcal{T}(G)} I^T(\boldsymbol{\mu}, \boldsymbol{\mu}^m)$. For a given $\boldsymbol{\mu}$, Fromer and Globerson [7] showed that this separation oracle can be efficiently implemented with a maximum-weight spanning tree algorithm with the edge weights given by $w_{ij} = \boldsymbol{\mu}_{ij}\cdot\boldsymbol{\mu}_{ij}^m - \boldsymbol{\mu}_i\cdot\boldsymbol{\mu}_i^m - \boldsymbol{\mu}_j\cdot\boldsymbol{\mu}_j^m$.

Notice that this algorithm requires solving a linear program over the AELP in each iteration. For large problems arising in computer vision and computational biology, solving this LP with a standard LP-solver *even once* may be infeasible. In the next section, we present our proposed message-passing algorithm for solving the M-Best MAP LP.

5 M-Best MAP Lagrangian Relaxation: Tree-MRF

Let us first restrict our attention to tree-structured MRFs. This is simple enough a scenario to describe the main elements of our approach; we then discuss the general case in Section 6. For tree MRFs, there is a single spanning tree inequality (for each m), and to simplify notation we will refer to $I^T(\boldsymbol{\mu}, \boldsymbol{\mu}^m)$ simply as $I(\boldsymbol{\mu}, \boldsymbol{\mu}^m)$. Then M-Best MAP LP can be written as:

$$\min_{\boldsymbol{\mu}\in\mathcal{L}(G)} \quad \sum_{A\in\mathcal{V}\cup\mathcal{E}} \boldsymbol{\theta}_A\cdot\boldsymbol{\mu}_A \tag{5a}$$

$$s.t. \qquad I(\boldsymbol{\mu}, \boldsymbol{\mu}^m) \leq 0 \quad \forall m \in [M-1] \tag{5b}$$

Now instead of solving the above problem in the primal with an LP-solver as Fromer and Globerson [7] did, we will study the Lagrangian relaxation of this LP, formed by dualizing the spanning tree constraints:

$$f(\boldsymbol{\lambda}) = \min_{\boldsymbol{\mu}\in\mathcal{L}(G)} \sum_{A\in\mathcal{V}\cup\mathcal{E}} \boldsymbol{\theta}_A\cdot\boldsymbol{\mu}_A + \sum_{m=1}^{M-1} \lambda_m I(\boldsymbol{\mu}, \boldsymbol{\mu}^m), \tag{6}$$

where $\boldsymbol{\lambda} = \{\lambda_m \mid m \in [M-1]\}$ is the set of Lagrange multipliers, that determine the weight of the penalty imposed for violating the spanning tree constraints.

Note that this is a partial Lagrangian because we have only dualized the spanning tree constraints (5b), and have not dualized the constraints hidden inside the local polytope $\mathcal{L}(G)$.

Key Idea: Exploiting Structure. The partial Lagrangian immediately exposes a structure in the problem that the primal formulation was obfuscating, namely that the spanning tree inequality distributes according to a tree structure, *i.e.*

$$f(\boldsymbol{\lambda}) = \min_{\boldsymbol{\mu}\in\mathcal{L}(G)} \left\{ \sum_{i\in\mathcal{V}} \left(\boldsymbol{\theta}_i + \sum_{m=1}^{M-1} \lambda_m(1-d_i)\boldsymbol{\mu}_i^m\right)\cdot\boldsymbol{\mu}_i + \sum_{(i,j)\in\mathcal{E}} \left(\boldsymbol{\theta}_{ij} + \sum_{m=1}^{M-1}\lambda_m\boldsymbol{\mu}_{ij}^m\right)\cdot\boldsymbol{\mu}_{ij} \right\}. \quad (7)$$

Also recall that for a tree the local polytope $\mathcal{L}(G)$ has integral vertices. Thus the minimization above can be efficiently performed by running a combinatorial optimization algorithm (the standard two-pass max-product BP) on this perturbed MRF, and does not need to be solved with a generic LP solver. As we will see next, being able to efficiently *evaluate* the Lagrangian is all we need to be able to *optimize* the Lagrangian dual.

5.1 Projected Supergradient Ascent on the Lagrangian Dual

From the theory of Lagrangian duality, we know that for all values of $\boldsymbol{\lambda} \geq 0$, $f(\boldsymbol{\lambda})$ is a lower-bound on the value of the primal problem (5). The tightest lower-bound is obtained by solving the Lagrangian dual problem: $\max_{\boldsymbol{\lambda}\geq 0} f(\boldsymbol{\lambda})$. Since f is a non-smooth concave function, this can be achieved by the *supergradient ascent* algorithm, analogous to the subgradient descent for minimizing non-smooth convex functions [22]. Since $\boldsymbol{\lambda}$ is a constrained variable, we follow the projected supergradient ascent algorithm: iteratively updating the Lagrange multipliers according to the following update rule: $\boldsymbol{\lambda}^{(t+1)} \longleftarrow \left[\boldsymbol{\lambda}^{(t)} + \alpha_t \nabla f(\boldsymbol{\lambda}^{(t)})\right]_+$, where $\nabla f(\boldsymbol{\lambda}^{(t)})$ is the supergradient of f at $\boldsymbol{\lambda}^{(t)}$, α_t is the step-size and $[\cdot]_+$ is the projection operator that projects a vector onto the positive orthant. If the sequence of multipliers $\{\alpha_t\}$ satisfies $\alpha_t \geq 0$, $\lim_{t\to\infty}\alpha_t = 0$, $\sum_{t=0}^{\infty}\alpha_t = \infty$, then projected supergradient ascent converges to the optimum of the Lagrangian dual [22].

To find the supergradient of $f(\boldsymbol{\lambda})$, consider the following lemma (proved in [14]):

Lemma 1 *If $g(\boldsymbol{\lambda})$ is a point-wise minimum of linear functions:* i.e. $g(\boldsymbol{\lambda}) = \min_{\boldsymbol{\mu}} \boldsymbol{a}_{\boldsymbol{\mu}}\cdot\boldsymbol{\lambda} + b_{\boldsymbol{\mu}}$, *then one supergradient of g is given by $\nabla g(\boldsymbol{\lambda}) = \boldsymbol{a}_{\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda})}$, where $\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda}) \in \operatorname{argmin}_{\boldsymbol{\mu}} \boldsymbol{a}_{\boldsymbol{\mu}}\cdot\boldsymbol{\lambda} + b_{\boldsymbol{\mu}}$.*

Notice that f is indeed a point-wise minimum of linear functions. Mapping this lemma to (5), we can see that the supergradient of f for our formulation is given by:

$$\nabla f(\boldsymbol{\lambda}) = \left[I(\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda}), \boldsymbol{\mu}^1),\ \ldots,\ I(\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda}), \boldsymbol{\mu}^{M-1})\right]^T \quad (8)$$

where $\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda})$ is an optimal primal solution of (5) for the current setting of $\boldsymbol{\lambda}$. Thus the computation of the supergradient can be done with the same dynamic programming algorithm as for evaluating the Lagrangian.

This supergradient (and the update procedure) has an intuitive interpretation. Recall that the Lagrangian relaxation minimizes a linear combination of the energy and the value of the spanning tree inequality, with the weighting given by $\boldsymbol{\lambda}$. If $\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda}^{(t)})$ violates one of the spanning tree constraints, *i.e.* is not different from a previous solution $\boldsymbol{\mu}^m$, then the supergradient w.r.t. $\lambda_m^{(t)}$ will be positive and the cost for violating the constraint will increase after the update, thus encouraging the next solution $\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda}^{(t+1)})$ to satisfy the spanning tree constraints. Conversely, if the constraints are (strictly) satisfied, the supergradient is negative indicating that $\lambda_m^{(t)}$ may be over-penalizing for violations and may be reduced to allow lower energy solutions.

Tightness of the Lagrangian Relaxation. The primal problem (5) is an LP, and strong duality holds. Thus, the projected supergradient algorithm described above exactly solves the M-Best MAP LP of Fromer and Globerson [7]. The total complexity of the algorithm is $O(knL^2)$, where k is the number of dual ascent iterations, n is the number of nodes and L is the largest label space, *i.e.* $L = \max_i |X_i|$.

6 M-Best MAP Lagrangian Relaxation: General MRFs

In this section, we build on the basic ideas from the previous section to develop an algorithm for general graphs, which may contain exponentially many spanning trees. Recall that the M-Best MAP LP for general graphs is given by:

$$\min_{\boldsymbol{\mu}\in\mathcal{L}(G)} \sum_{A\in\mathcal{V}\cup\mathcal{E}} \boldsymbol{\theta}_A\cdot\boldsymbol{\mu}_A \quad (9a)$$

$$s.t. \quad I^T(\boldsymbol{\mu}, \boldsymbol{\mu}^m) \leq 0, \quad \forall T\in\mathcal{T}(G), \forall m\in[M-1] \quad (9b)$$

where $\mathcal{T}(G)$ is the set of all spanning trees in G. As before, the Lagrangian formed by dualizing the spanning tree constraints is given by:

$$f(\boldsymbol{\lambda}) = \min_{\boldsymbol{\mu}\in\mathcal{L}(G)} \sum_{A\in\mathcal{V}\cup\mathcal{E}} \boldsymbol{\theta}_A\cdot\boldsymbol{\mu}_A + \sum_{m=1}^{M-1}\sum_{T\in\mathcal{T}(G)} \lambda_{mT} I^T(\boldsymbol{\mu}, \boldsymbol{\mu}^m) \quad (10)$$

There are two main concerns that prevent us from directly solving this Lagrangian relaxation as before. First, the set of Lagrange multipliers $\boldsymbol{\lambda} = \{\lambda_{mT} \mid m \in [M-1], T\in\mathcal{T}(G)\}$ is exponentially large. And second, the graph is no longer a tree, so the supergradient can no longer be computed by max-product BP. We address both these concerns in the next subsections.

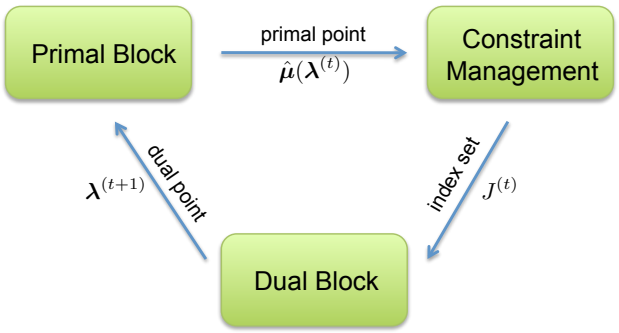


Figure 1: Overview of a Dynamic Supergradient Method. Figure adapted from [4].

6.1 Optimizing over Exponentially Many Dual Variables with a Dynamic Supergradient Method

In order to optimize over the exponentially large set of dual variables, we follow a *dynamic* supergradient method (DSM) (see [4] for an overview). Intuitively, DSMs can be thought of as the dual-procedure to the cutting-plane algorithm.

A DSM maintains an index set of *active* dual variables ($J = \cup_m J^m$, where $J^m \subset \{1, \ldots, |T(G)|\}$), and all inactive dual variables are fixed to zero, *i.e.* $\lambda_{mj} = 0,\ \forall j \notin J^m$. As visualized in Fig. 1, DSMs consist of three kinds of operations:

1. **Primal Block:** Given the current dual variable $\boldsymbol{\lambda}^{(t)}$, the primal block evaluates the Lagrangian (10) to find a new primal point $\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda}^{(t)})$.
2. **Constraint Management Block:** Given the current primal point $\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda}^{(t)})$, the constraint management block augments the index set of active dual variables to get the new index set $J^{(t)}$. Optionally, this block may also choose to drop some dual variables from the index set. This block is described in detail below.
3. **Dual Block:** Given the current index $J^{(t)}$, the dual block constructs the dual update direction (the supergradient) and stepsize to produce a new dual variable $\boldsymbol{\lambda}^{(t+1)}$ with $\lambda_{mj}^{(t+1)} = 0,\ \forall j \notin J^{m(t)}$.

The primal and dual blocks will be discussed in the next subsection, where we describe how the supergradient may be efficiently computed for general graphs.

We now describe the constraint management block in detail. To develop an intuition for this step, recall that the complementary slackness condition [2] tells us that given a pair of optimal primal dual variables ($\boldsymbol{\mu}^*$ and $\boldsymbol{\lambda}^*$): $I^T(\boldsymbol{\mu}^*, \boldsymbol{\mu}^m) < 0 \implies \lambda^*_{mT} = 0$. Thus, intuitively the active set $J^{(t)}$ must focus on the dual variables corresponding to the violated inequalities. We use a maximum violated oracle that adds to the active set $J^{m(t)}$, the index of the dual variable corresponding to the most violated inequality, *i.e.*

$$J^{m(t+1)} \longleftarrow J^{m(t)} \cup \text{index}(\hat{T}) \tag{11}$$

where,

$$\hat{T} = \operatorname*{argmax}_{T \in \mathcal{T}(G)} \sum_{(i,j) \in T} w_{ij}, \qquad \textbf{[Max-Wt Span-Tree]} \tag{12a}$$

$$w_{ij} = \hat{\boldsymbol{\mu}}_{ij}(\boldsymbol{\lambda}^{(t)}) \cdot \boldsymbol{\mu}^m_{ij} - \hat{\boldsymbol{\mu}}_i(\boldsymbol{\lambda}^{(t)}) \cdot \boldsymbol{\mu}^m_i - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\lambda}^{(t)}) \cdot \boldsymbol{\mu}^m_j. \tag{12b}$$

We note that this process is the dualized version of the cutting-plane method of Fromer and Globerson [7], where instead of adding the most violated spanning tree inequality to the LP, we include the index of its dual variable to the working set.

It can be shown that a dynamic supergradient method with such a maximum-violation-oracle constraint management block is guaranteed to converge to the optimum of the Lagrangian relaxation with the same choice of stepsize rules as standard supergradient methods. A rigorous proof can be found here [4]. DSMs actually allow for dual variables to be removed from the active set as well, under certain conditions. However, to keep the exposition simple and to match our implementation, we do not discuss that here, and refer the reader to [4].

The theory allows for several iterations of primal and dual blocks to be performed for each constraint management step. We found this to be crucial in practice.

6.2 Computing the Supergradient for General Graphs

Now let us address the problem of computing the supergradient, and implementing the primal and dual blocks. Given the current index set of active dual variables $J = \cup_m J^m$, the supergradient w.r.t. the active dual variables $\boldsymbol{\lambda}_J$ is given by:

$$\nabla_{\lambda_{mj}} f(\boldsymbol{\lambda}) = I^j(\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda}), \boldsymbol{\mu}^m) \qquad \forall j \in J^m \tag{13}$$

where $\hat{\boldsymbol{\mu}}(\boldsymbol{\lambda})$ is an optimal primal solution of (10) for the current setting of $\boldsymbol{\lambda}$. For the case of tree-MRFs we could compute the optimal solution via dynamic programming. For general MRFs, the supergradient computation involves solving:

$$f(\boldsymbol{\lambda}) = \min_{\boldsymbol{\mu} \in \mathcal{L}(G)} \left\{ \sum_{i \in \mathcal{V}} \left(\boldsymbol{\theta}_i + \sum_{m=1}^{M-1} \sum_{j \in J^m} \lambda_{mj} (1 - d_i^{T_j}) \boldsymbol{\mu}_i^m \right) \cdot \boldsymbol{\mu}_i + \sum_{(i,j) \in \mathcal{E}} \left(\boldsymbol{\theta}_{ij} + \sum_{m=1}^{M-1} \sum_{j \in J^m} \lambda_{mj} \boldsymbol{\mu}_{ij}^m \right) \cdot \boldsymbol{\mu}_{ij} \right\}, \tag{14}$$

where G is now a graph with loops. Evaluating the above Lagrangian involves solving a MAP inference LP (with modified potentials) and thus any message-passing algorithm for MAP (MPLP [8], Dual-Decomposition [13], or Max-Sum Diffusion [26]) may

be used to solve this problem. However, unlike the two-pass max-product BP used for tree-MRFs, these message-passing algorithms typically require many hundreds of iterations to converge. Running these iterations for each step of supergradient ascent can (and in practice does) become prohibitively slow.

The key to speeding up this process is to realize that $f(\boldsymbol{\lambda})$ is a *partial* Lagrangian, and if evaluating it is difficult, then this suggests that some more constraints should be dualized till the partial Lagrangian becomes tractable. This is precisely what we do, using ideas from the dual-decomposition literature [9, 13].

Expanding the Partial Lagrangian. In order to expand the partial Lagrangian, we will first try to identify tractable (tree-structured) subcomponents. The spanning-tree inequalities are already tree structured. G is not, but we can convert it into a collection of tree-structured factors.

Let $TC(G) = \{T_1, \ldots, T_p\}$ be a *spanning-tree cover* of G, *i.e.* a collection of spanning trees such that each edge of G appears in at least one tree in $TC(G)$. Our approach doesn't really need the trees to be spanning, but we describe the following with a spanning-tree cover to keep the notation simple. With a slight abuse of notation, we use $TC(i,j)$ to denote the subset of trees that contain edge $(i,j) \in \mathcal{E}$. Moreover, we decompose the original energy function $\boldsymbol{\theta}$ into a collection of energy functions $\{\boldsymbol{\theta}^T \mid T \in TC(G)\}$, one for each tree in the tree cover, such that:

$$\sum_{T \in TC(G)} \boldsymbol{\theta}_i^T = \boldsymbol{\theta}_i \;\; \forall i \in \mathcal{V} \quad \& \quad \sum_{T \in TC(i,j)} \boldsymbol{\theta}_{ij}^T = \boldsymbol{\theta}_{ij} \;\; \forall (i,j) \in \mathcal{E} \tag{15a}$$

$$\implies \sum_{T \in TC(G)} \boldsymbol{\theta}^T \cdot \boldsymbol{\mu} = \boldsymbol{\theta} \cdot \boldsymbol{\mu} \tag{15b}$$

This can be easily satisfied by distributing the node and edge energies "evenly", *i.e.* $\boldsymbol{\theta}_i^T = \frac{1}{|T(G)|}\boldsymbol{\theta}_i$, $\boldsymbol{\theta}_{i,j}^T = \frac{1}{|T(i,j)|}\boldsymbol{\theta}_{ij}$. Thus, these energies specify a tree decomposition [24] of $\boldsymbol{\theta}$.

Let us now assign to each tree in $TC(G)$, its own copy of the primal variables $\boldsymbol{\mu}^T$, $\forall T \in TC(G)\}$. Also assign to each spanning tree inequality in the active set J^m, its own copy of the primal variables $\boldsymbol{\mu}^{mj}$, $\forall j \in J^m$. Finally, we use $\boldsymbol{\theta}_i^{mj} \triangleq \lambda_{mj}(1-d_i)\boldsymbol{\mu}_i^m$ to denote the node energy of the spanning tree factor $j \in J^m$, and $\boldsymbol{\theta}_i^{mj} = \lambda_{mj}\boldsymbol{\mu}_{ij}^m$ to denote the edge energy.

With these new variables, we can now write the existing partial Lagrangian as:

$$f(\boldsymbol{\lambda}) = \min_{\boldsymbol{\mu} \in \mathcal{L}(G), \tilde{\boldsymbol{\mu}}} \left\{ \sum_{T \in TC(G)} \boldsymbol{\theta}^T \cdot \boldsymbol{\mu}^T + \sum_{m=1}^{M-1} \sum_{j \in J^m} \boldsymbol{\theta}^{mj} \cdot \boldsymbol{\mu}^{mj} \right\} \tag{16a}$$

$$s.t. \quad \boldsymbol{\mu}_i^\tau = \tilde{\boldsymbol{\mu}}_i \qquad \forall \tau \in TC(G) \cup J, \;\; \forall i \in \mathcal{V} \tag{16b}$$

This formulation of the partial Lagrangian uses a global variable $\tilde{\boldsymbol{\mu}}$ to force all tree-structured subproblems to agree on the labellings at the nodes, and thus is equivalent to the earlier formulation (14). However, we can now expand this partial Lagrangian by further dualizing these constraints (16b):

$$f(\boldsymbol{\lambda}, \boldsymbol{\delta}) = \min_{\boldsymbol{\mu}^\tau \in \mathcal{L}(\tau), \tilde{\boldsymbol{\mu}}} \left\{ \sum_{T \in TC(G)} \boldsymbol{\theta}^T \cdot \boldsymbol{\mu}^T + \sum_{m=1}^{M-1} \sum_{j \in J^m} \boldsymbol{\theta}^{mj} \cdot \boldsymbol{\mu}^{mj} + \sum_{\tau \in TC(G) \cup J} \sum_{i \in \mathcal{V}} \boldsymbol{\delta}_{\tau i} \cdot \left(\boldsymbol{\mu}_i^\tau - \tilde{\boldsymbol{\mu}}_i \right) \right\} \tag{17}$$

where $\boldsymbol{\delta} = \{\boldsymbol{\delta}_{\tau i} \mid \forall \tau \in TC(G) \cup J, \;\; \forall i \in \mathcal{V}\}$ is the set of Lagrangian multipliers for the dualized equality constraints, and $\mathcal{L}(\tau)$ is now the local polytope of each of the tree-structured subproblems.

Notice that this expanded partial Lagrangian completely decouples into independent minimizations over tree-structured subproblems:

$$f(\boldsymbol{\lambda}, \boldsymbol{\delta}) = \sum_{\tau \in TC(G) \cup J} \min_{\boldsymbol{\mu}^\tau \in \mathcal{L}(\tau)} \left\{ \sum_{i \in \mathcal{V}} \left(\boldsymbol{\theta}_i^\tau + \boldsymbol{\delta}_{\tau i} \right) \cdot \boldsymbol{\mu}_i^\tau + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{\theta}_{ij}^\tau \cdot \boldsymbol{\mu}_{ij}^\tau \right\} - \min_{\tilde{\boldsymbol{\mu}}} \left(\sum_{\tau \in TC(G) \cup J} \sum_{i \in \mathcal{V}} \boldsymbol{\delta}_{\tau i} \right) \cdot \tilde{\boldsymbol{\mu}}_i \tag{18}$$

We can see that the unconstrained minimization over $\tilde{\boldsymbol{\mu}}$ forces a constraint on the Lagrangian variables, *i.e.* $\sum_{\tau \in TC(G) \cup J} \sum_{i \in \mathcal{V}} \boldsymbol{\delta}_{\tau i} = 0$; otherwise the Lagrangian will not have finite value. We denote by $\Delta \triangleq \{\boldsymbol{\delta} \mid \sum_{\tau \in TC(G) \cup J} \sum_{i \in \mathcal{V}} \boldsymbol{\delta}_{\tau i} = 0\}$, the set of all feasible Lagrangian multipliers $\boldsymbol{\delta}$.

Note that the expanded partial Lagrangian can be efficiently evaluated by running two-pass max-product BP on each of the tree-structured subproblems. The number of such tree-structured subproblems is equal to $|TC(G)|+|J|$, *i.e.* the size of the spanning-tree cover (upper bounded by n, and typically a small constant) and the number of active spanning tree inequalities.

Optimizing the Expanded Partial Lagrangian. The dual problem for the expanded partial Lagrangian is given by $\max_{\boldsymbol{\lambda} \geq 0, \boldsymbol{\delta} \in \Delta} f(\boldsymbol{\lambda}, \boldsymbol{\delta})$. In the previous section, we described how the dual of the partial Lagrangian $f(\boldsymbol{\lambda})$ can be optimized with a dynamic supergradient method. Optimizing the dual of the expanded Lagrangian is very similar to the described procedure, with the minor modification that the dual variable $\boldsymbol{\delta}$ always stays in the active set J.

The supergradient w.r.t. $\boldsymbol{\delta}$ is given by:

$$\nabla_{\boldsymbol{\delta}_{\tau i}} f(\boldsymbol{\lambda}, \boldsymbol{\delta}) = \hat{\boldsymbol{\mu}}_i^\tau(\boldsymbol{\lambda}, \boldsymbol{\delta}) \quad \forall \tau \in TC(G) \cup J \;\; \forall i \in \mathcal{V}, \tag{19}$$

where $\hat{\boldsymbol{\mu}}_i^\tau(\boldsymbol{\lambda}, \boldsymbol{\delta})$ is an optimal primal solution of the tree suproblem τ for the current setting of $\boldsymbol{\lambda}$ and $\boldsymbol{\delta}$.

The projection step onto Δ is fairly simple – it involves satisfying the zero-sum constraint, which can

be enforced by subtracting the mean of the dual variables. Overall, the dual update w.r.t. $\boldsymbol{\delta}$ is given by $\boldsymbol{\delta}^{(t+1)} = \left[\boldsymbol{\delta}^{(t)} + \alpha_t \nabla_{\boldsymbol{\delta}} f(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\delta}^{(t)})\right]_0$, where α_t is the stepsize and $[\cdot]_0$ is the zero-projection operator *i.e.* $[\boldsymbol{\delta}^{(t)}]_0 = \{\boldsymbol{\delta}^{(t)}_{\tau i} - s \mid \forall \tau \in TC(G) \cup J,\ \forall i \in \mathcal{V}\}$, where $s = \frac{1}{|\boldsymbol{\delta}|} \sum_{\tau' \in TC(G) \cup J} \sum_{j \in \mathcal{V}} \boldsymbol{\delta}^{(t)}_{\tau' j}$.

The entire algorithm is summarized in Algorithm 1. We call our algorithm STEELARS for Spanning TrEe inEquality LAgrangian Relaxation Scheme.

Algorithm 1 STEELARS

$(\boldsymbol{\mu}^*, \{\boldsymbol{\lambda}^*, \boldsymbol{\delta}^*\}) = \text{STEELARS}(G, \theta, \{\boldsymbol{\mu}^m\})$
Input: $G = (\mathcal{V}, \mathcal{E})$, graph instance,
$\boldsymbol{\theta} = \{\theta_A \mid A \in \mathcal{V} \cup \mathcal{E}\}$ energy vector
$\{\boldsymbol{\mu}^m \mid m \in [M-1]\}$, $M-1$ previous solutions
Output: $\boldsymbol{\mu}^*_A \in [0,1]^{X_A}$, optimum solution vector to M-Best MAP LP
$\{\boldsymbol{\lambda}^*, \boldsymbol{\delta}^*\}$, optimum dual variables
Algorithm:
Construct a tree-decomposition of G: $TC(G), \{\boldsymbol{\theta}^T \mid T \in TC(G)\}$
$\boldsymbol{\lambda}^{(0)} \longleftarrow \mathbf{0}$; $\boldsymbol{\delta}^{(0)} \longleftarrow \mathbf{0}$; $\hat{\boldsymbol{\mu}} \longleftarrow \mathbf{0}$
$J^{m(0)} \longleftarrow \emptyset, \quad \forall m \in [M-1]$
$t \longleftarrow 0$
while Not Converged **do**
  {Constraint Management Block}
  for $m = 1, \ldots, M-1$ **do**
    $w_{ij} = \hat{\boldsymbol{\mu}}_{ij} \cdot \boldsymbol{\mu}^m_{ij} - \hat{\boldsymbol{\mu}}_i \cdot \boldsymbol{\mu}^m_i - \hat{\boldsymbol{\mu}}_j \cdot \boldsymbol{\mu}^m_j$
    $\hat{T} = \text{argmax}_{T \in \mathcal{T}(G)} \sum_{(i,j) \in T} w_{ij}$ {Max-weight Spanning Tree.}
    $J^{m(t+1)} \longleftarrow J^{m(t)} \cup \text{index}(\hat{T})$
  end for
  $J^{(t)} \longleftarrow \cup_{m \in [M-1]} J^{m(t)}$
  {Multiple Iteration of Primal & Dual Blocks}
  for $t' = 1, \ldots, 20$ **do**
    $\boldsymbol{\theta}^{mj}_i \longleftarrow \lambda^{(t)}_{mj}(1 - d_i)\boldsymbol{\mu}^m_i$
    $\boldsymbol{\theta}^{mj}_{ij} \longleftarrow \lambda^{(t)}_{mj}\boldsymbol{\mu}^m_{ij} \qquad \forall m \in [M-1], \forall j \in J^m$
    {Primal Block}
    for $\tau \in TC(G) \cup J$ **do**
      $\boldsymbol{\theta}^\tau_i \longleftarrow \boldsymbol{\theta}^\tau_i + \boldsymbol{\delta}^{(t)}_{\tau i}$; $\boldsymbol{\theta}^\tau_{ij} \longleftarrow \boldsymbol{\theta}^\tau_{ij}$
      $\hat{\boldsymbol{\mu}}^\tau(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\delta}^{(t)}) = \text{argmin}_{\boldsymbol{\mu}^\tau \in \mathcal{L}(\tau)} \boldsymbol{\theta}^\tau \cdot \boldsymbol{\mu}^\tau$ {Two-pass max-product BP.}
    end for
    {Dual Block}
    $\nabla_{\boldsymbol{\lambda}_{mj}} f(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\delta}^{(t)}) \longleftarrow I^j(\hat{\boldsymbol{\mu}}^{mj}(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\delta}^{(t)}), \boldsymbol{\mu}^m)$
    $\nabla_{\boldsymbol{\delta}_{\tau i}} f(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\delta}^{(t)}) \longleftarrow \hat{\boldsymbol{\mu}}^\tau_i(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\delta}^{(t)})$
    $\boldsymbol{\lambda}^{(t+1)} \longleftarrow \left[\boldsymbol{\lambda}^{(t)} + \alpha_t \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\delta}^{(t)})\right]_+$
    $\boldsymbol{\delta}^{(t+1)} \longleftarrow \left[\boldsymbol{\delta}^{(t)} + \alpha_t \nabla_{\boldsymbol{\delta}} f(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\delta}^{(t)})\right]_0$
    $t \longleftarrow t + 1$
  end for
  $\hat{\boldsymbol{\mu}} \longleftarrow$ Best Feasible Primal So Far.
  $\hat{\boldsymbol{\mu}}^* \longleftarrow$ Running Average of $\hat{\boldsymbol{\mu}}^\tau$
end while

Tightness of STEELARS. At first glance, it may seem like the expanded partial Lagrangian (18) solves a weaker relaxation than our original partial Lagrangian (10). However, similar to our argument in the tree-MRF case, problem (9) is an LP. Thus, strong duality holds and *all* partial Lagrangians achieve the same optimum. This implies that even for non-tree MRFs, STEELARS *exactly* solves the M-Best MAP LP of Fromer and Globerson [7] and does not introduce any new approximation gap to the integer program. Moreover, this guarantee this does not depend on the choice of the spanning-tree cover $TC(G)$; in fact, any tree cover (even non-spanning) may be used.

Finally, note that we described the expanded partial Lagrangian in terms of tree-structured subproblems. However, any efficient subproblem may be used, *e.g.* submodular subproblems solved with graph-cuts [12].

7 Experiments

Setup. We tested our algorithm in three scenarios:

1. Tree MRFs with $M = 2$. This is the simplest case, where the M-Best MAP LP is guaranteed to be tight. Moreover, there is a single spanning tree inequality and the constraint management block plays no role.
2. 2-label Submodular MRFs with $M = 5$. For such problems the MAP LP is guaranteed to be tight, but not the M-Best MAP LP. Moreover, a tree decomposition is not required because the 2-label submodular factor may be efficiently minimized via graph-cuts [12].
3. General 4-label loopy MRFs with $M = 5$. This is the most general case described in Section 6.

Baselines. We compared our algorithm with the STRIPES algorithm of Fromer and Globerson [7], the Lawler-Nilsson algorithm [16,18], and the BMMF algorithm of Yanover and Weiss [28]. Recall that STRIPES does not directly solve the M-Best MAP LP, but rather solves a sequence of 2nd-Best MAP LPs encapsulated in the Lawler-Nilsson [16, 18] partitioning scheme. This is a tradeoff between efficiency and accuracy – it would be more efficient to directly solve the M-Best MAP LP, but the LP is fractional and thus the partitioning scheme performs better. Note that STEELARS could also be encapsulated in the Lawler-Nilsson [16, 18] partitioning scheme in a straightforward manner. However, we wish to study the performance of the Lagrangian relaxation for the M-Best MAP LP directly, and leave this partitioning scheme extension for future work.

Implementation details. The implementations for STRIPES and Lawler-Nilsson were provided by the authors of [7], while BMMF is provided by the authors of [28]. For STRIPES, the LPs in each iteration were solved using the `GNU LPK` library. STEELARS is implemented in MATLAB, but max-product BP is

written in C++ for efficiency. All experiments are performed on a 64-bit 8-Core Intel i7 machine with 12GB RAM and the timing reported is cputime. Following [15], we chose the stepsize at iteration t to be $\alpha_t = \frac{1}{\eta_t+1}$, where η_t is the number of times the objective value $f(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\delta}^{(t)})$ has decreased from one iteration to the next. This rule has the same convergence guarantees as the standard $\alpha_t = \frac{1}{t}$ decaying rule, but empirically performs much better.

Integer Primal Extraction. STEELARS is a dual-ascent algorithm and thus always maintains a feasible dual solution, but not necessarily an (integer) primal feasible solution, which is what we are interested in. However, since the primal block is repeatedly called for computing the supergradient, we simply keep track of the best (integer) primal feasible solution produced so far, and output that at the end of the algorithm.

Evaluation. We compare different algorithms on two metrics – run-time and accuracy of solutions returned. For tree-MRFs with $M = 2$ all methods are guaranteed to return exact solutions, and thus we can simply compare the run-times. For general MRFs, we follow the protocol of [7] and measure relative accuracy of different methods. Specifically, we pool all solutions returned by all methods, note the top M solutions in this pool, and then for each method report the fraction of these solutions that it contributed.

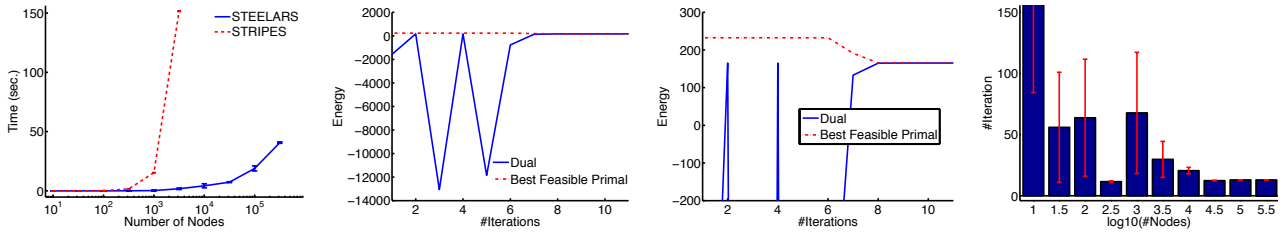
Our results will demonstrate the effectiveness of STEELARS primarily in terms of efficiency. We will show that since STEELARS is a message-passing algorithm it is significantly faster than a generic LP solver (`GLPK`), sometimes by *orders of magnitude* even though it is guaranteed to converge to the same solution. STEELARS naturally scales to large instances that were previously unsolvable using LP solvers.

Tree MRFs with $M = 2$. We generated synthetic problems by sampling random spanning trees on n nodes. Each variable could take 4 labels. Node and edge potentials were sampled from standard Gaussians. For $M = 2$ the M-Best MAP LP is guaranteed to be tight, and thus both STRIPES and STEELARS produces precisely the same answers. Fig. 2a shows the time taken by both algorithms as a function of the size of the tree n, averaged over 5 samplings of the parameters. Note that the x-axis is in log-scale. As n increase, STRIPES very quickly becomes intractable, with `GPLK` ultimately running out of memory. However, STEELARS shows a much better behaviour in runtime. Fig. 2 also shows the value of the dual and best feasible integer primal produced by STEELARS as a function of the number of iterations. Recall that for STEELARS each iteration corresponds to running max-product BP on a single tree. Fig. 2d shows that generally the number of such iterations is pretty low $\sim$ 12. Note that since this is a tree MRF, BMMF would only require 2 call to BP to compute max-marginals under its partitioning scheme (assuming it was carefully implemented to recognize a tree-graph and not run asynchronous BP). Thus STEELARS is not too much slower than the optimal thing to do for a tree. Of course, BMMF would also require multiple iterations of synchronous BP on loopy graphs, which is what we checked next.

2-label Submodular MRFs. For this scenario, we constructed $\sqrt{n} \times \sqrt{n}$ grids. Each variable could take 2 labels. We again sampled node and edge energies from Gaussians, but ensured that edge energies were submodular. This allows for the use of graph-cuts [12] for optimizing the submodular factor. Fig. 3 shows the value of the dual and best feasible integer primal as a function of the number of iterations. The sharp falls in the dual correspond to the constraint management block calling the separation oracle to increment the working set. Each iteration now involves one call to graph-cuts and $|J^{(t)}|$ calls to two-pass max-product BP. Notice that the number of iterations are much larger than for the tree MRF ($\sim$150 as opposed to 12). However, max-flow algorithms in general and the implementation of [12] in particular, are highly efficient. Fig. 3a shows the run-time of all algorithms as a function of n. We can see that STEELARS is by far the fastest, with other algorithms becoming intractable very quickly. Unfortunately, as Fig. 3d shows it is also the least accurate, validating the choice of Fromer and Globerson [7] to not solve the LP directly, and use a partitioning scheme instead. We plan to follow up on this direction.

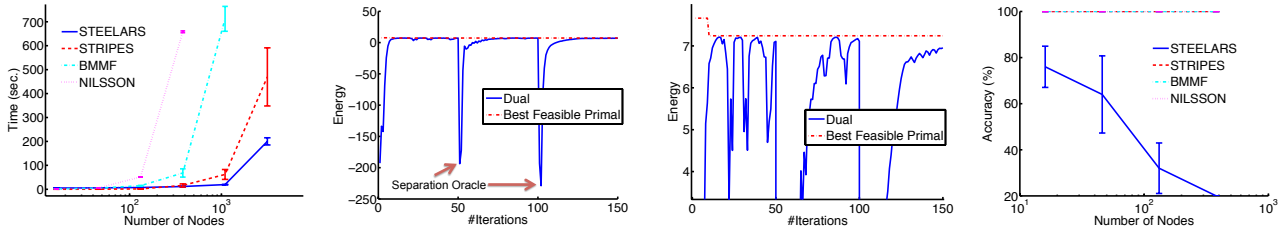
Interestingly, Fig. 3a seems to suggest that BMMF performs worse than STRIPES, even though BMMF simply involves M calls to loopy BP. We believe this an artifact caused by the fact that the BMMF implementation of [28] is written in MATLAB, and not particularly optimized. Moreover, in this specific experiment, it could be made faster by computing max-marginals via the approach of [11] instead of loopy BP. At a high-level, the key difference between BMMF and STEELARS is analogous to the difference between loopy BP and MPLP – both are message-passing algorithms, but only one solves the LP relaxation and provides improving lower-bounds.

General MRFs. For this scenario, we constructed $\sqrt{n} \times \sqrt{n}$ grid graphs as well, but each variable could take 4-labels and edge energies were not restricted to be "attractive". We used a standard two-tree decomposition of the grid graph. Thus, each iteration of supergradient ascent involved $2 + |J^{(t)}|$ calls to max-product BP. We observed trends similar to the submodular MRFs case, both in terms of number of iterations required for STEELARS to converge, and in the relative standing w.r.t. the baselines. This case is



(a) Time vs. n (b) Primal and Dual vs. Iterations. (c) Primal and Dual vs. Iterations (Zoomed). (d) No. of Supergradient Iterations.

Figure 2: Tree MRF with $M = 2$.



(a) Time vs. n (b) Primal and Dual vs. Iterations. (c) Primal and Dual vs. Iterations (Zoomed). (d) Accuracy of Solutions.

Figure 3: 2-label Submodular MRFs with $M = 5$.

specially interesting because for small values of $|J^{(t)}|$, the work done by STEELARS (*i.e.* ∼150 iteration of two-pass BP) for solving M-Best MAP is comparable to running TRW [13,24] for MAP. Moreover, we could run STEELARS on a 300×500 image labelling problem, but similar to the observation of [27], could not even solve the MAP LP with `GLPK`.

8 Conclusions

In conclusion, we presented the *first* message-passing algorithm for solving the LP relaxation of the M-Best MAP problem in discrete undirected graphical models. Our approach used a particular Lagrangian relaxation to construct a partial Lagrangian that allowed the use of combinatorial optimization algorithms. To handle the exponentially large set of constraints, we used a dynamic supergradient scheme that is essentially a dual procedure to the cutting-plane algorithm. Our message-passing algorithm retains all the guarantees of the LP formulation of Fromer and Globerson [7], while being *orders of magnitude* faster.

Extracting Diverse M-Best Solutions. In a number of applications, especially computer vision, the M-Best MAP solutions are essentially minor perturbations of each other. In concurrent work [1], we have also presented a solution to the *Diverse* M-Best MAP problem, where given a measure of 'distance' between two solutions, we block all solutions within some k-distance-ball of the previous solutions.

We hope that both these algorithms will be useful for practitioners where the problem size prohibits the use of generic LP-solvers.

Future Work. There are a number of interesting directions in front of us. In the short-term, we are interested in encapsulating STEELARS inside the Lawler-Nilsson partitioning scheme, similar to STRIPES to further increase the accuracy of the method. Since the M-Best MAP LP is so often fractional, another direction is to *tighten* the LP, *e.g.* using techniques proposed by Sontag *et al.* [23]. It would also be interesting to compare LP-relaxation based methods like STRIPES and STEELARS with heurisitic-search methods [3].

Acknowledgements. We thank Sebastian Nowozin for being the human encyclopedia of optimization and pointing us to the literature of Dynamic Subgradient Methods; Amir Globerson for sharing the STRIPES implementation and useful discussions; Danny Tarlow for discussions that helped formalize the formulation.

References

[1] D. Batra, P. Yadollahpour, A. Guzmn-Rivera, and G. Shakhnarovich. Diverse M-Best Solutions in Markov Random Fields. In *ECCV*, 2012. 9

[2] D. Bertismas and J. Tsitsiklis. *Introduction to linear optimization.* Athena Scientific, 1997. 5

[3] R. Dechter, N. Flerova, and R. Marinescu. Search algorithms for m best solutions for graphical models. In *AAAI*, 2012. 9

[4] G. Emiel and C. Sagastizabal. Dynamic subgradient methods. Optimization Online: http://www.optimization-online.org/DB_HTML/2008/08/2077.html, 2008. 1, 5

[5] N. F. Emma Rollon and R. Dechter. Inference schemes for m best solutions for soft csps. In *Proceedings of Workshop on Preferences and Soft Constraints*, 2011. 2

[6] D. Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2):652–673, 1998. 2

[7] M. Fromer and A. Globerson. An LP view of the m-best MAP problem. In *NIPS*, 2009. 1, 2, 3, 4, 5, 7, 8, 9

[8] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, 2007. 1, 3, 5

[9] M. Guignard. Lagrangean relaxation. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 11(2):151–200, 2003. 6

[10] H. W. Hamacher and M. Queyranne. K best solutions to combinatorial optimization problems. *Annals of Operations Research*, 4:123–143, 1985. 10.1007/BF02022039. 2

[11] P. Kohli and P. H. S. Torr. Measuring uncertainty in graph cut solutions. *CVIU*, 112(1):30–38, 2008. 2, 8

[12] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26(2):147–159, 2004. 7, 8

[13] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007. 1, 3, 5, 6, 9

[14] N. Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *PAMI*, 33(3):531 –552, march 2011. 4

[15] T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *Conference on Empirical Methods in Natural Language Processing*, 2010. 8

[16] E. L. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18:401–405, 1972. 2, 7

[17] E. R. Natalia Flerova and R. Dechter. Bucket and mini-bucket schemes for m best solutions over graphical models. In *IJCAI Workshop on Graph Structures for Knowledge Representation and Reasoning*, 2011. 2

[18] D. Nilsson. An efficient algorithm for finding the m most probable configurations in probabilistic expert systems. *Statistics and Computing*, 8:159–173, 1998. 10.1023/A:1008990218483. 1, 7

[19] M. I. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions (in Russian). *Kibernetika*, 4:113–130, 1976. 1, 3

[20] B. Seroussi and J. Golmard. An algorithm directly finding the k most probable configurations in bayesian networks. *Int. J. of Approx. Reasoning*, 11(3):205 – 233, 1994. 1

[21] S. E. Shimony. Finding MAPs for belief networks is np-hard. *Artificial Intelligence*, 68(2):399–410, August 1994. 3

[22] N. Shor. *Minimization methods for non-differentiable functions*. Springer series in computational mathematics. Springer-Verlag, 1985. 4

[23] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *UAI*, 2008. 9

[24] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *Trans. Inf. Th.*, 51(11):3697–3717, 2005. 1, 3, 6, 9

[25] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008. 3

[26] T. Werner. A linear programming approach to max-sum problem: A review. *PAMI*, 29(7):1165–1179, 2007. 1, 3, 5

[27] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – an empirical study. *J. Mach. Learn. Res.*, 7:1887–1907, 2006. 1, 2, 9

[28] C. Yanover and Y. Weiss. Finding the m most probable configurations using loopy belief propagation. In *NIPS*, 2003. 1, 2, 7, 8

Lifted Relax, Compensate and then Recover: From Approximate to Exact Lifted Probabilistic Inference

Guy Van den Broeck*
Department of Computer Science
KU Leuven
guy.vandenbroeck@cs.kuleuven.be

Arthur Choi and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
{aychoi,darwiche}@cs.ucla.edu

Abstract

We propose an approach to lifted approximate inference for first-order probabilistic models, such as Markov logic networks. It is based on performing exact lifted inference in a simplified first-order model, which is found by relaxing first-order constraints, and then compensating for the relaxation. These simplified models can be incrementally improved by carefully recovering constraints that have been relaxed, also at the first-order level. This leads to a spectrum of approximations, with lifted belief propagation on one end, and exact lifted inference on the other. We discuss how relaxation, compensation, and recovery can be performed, all at the first-order level, and show empirically that our approach substantially improves on the approximations of both propositional solvers and lifted belief propagation.

1 INTRODUCTION

Probabilistic logic models combine aspects of first-order logic and probabilistic graphical models, enabling them to model complex logical and probabilistic interactions between large numbers of objects (Getoor and Taskar, 2007; De Raedt et al., 2008). This level of expressivity comes at the cost of increased complexity of inference, motivating a new line of research in lifted inference algorithms (Poole, 2003). These algorithms exploit logical structure and symmetries in probabilistic logics to perform efficient inference in these models.

For exact inference, lifted variants of variable elimination (Poole, 2003; de Salvo Braz et al., 2005; Milch et al., 2008; Taghipour et al., 2012) and weighted model counting (Gogate and Domingos, 2011; Van den Broeck et al., 2011) have been proposed. With some exceptions, lifted approximate inference has focused on lifting iterative belief propagation (Singla and Domingos, 2008; Kersting et al., 2009).

In this paper, we propose an approach to *approximate* lifted inference that is based on performing *exact* lifted inference in a simplified first-order model. Namely, we simplify the structure of a first-order model until it is amenable to exact lifted inference, by relaxing first-order equivalence constraints in the model. Relaxing equivalence constraints ignores (many) dependencies between random variables, so we compensate for this relaxation by restoring a weaker notion of equivalence, in a lifted way. We then incrementally improve this approximation by recovering first-order equivalence constraints back into the model.

In fact our proposal corresponds to an approach to approximate inference, called Relax, Compensate and then Recover (RCR) for (ground) probabilistic graphical models.[1] For such models, the RCR framework gives rise to a spectrum of approximations, with iterative belief propagation on one end (when we use the coarsest possible model), and exact inference on the other (when we use the original model). In this paper, we show how relaxations, compensations and recovery can all be performed at the first-order level, giving rise to a spectrum of first-order approximations, with lifted first-order belief propagation on one end, and exact lifted inference in the other.

We evaluate our approach on benchmarks from the lifted inference literature. Experiments indicate that recovering a small number of first-order equivalences can improve on the approximations of lifted belief propagation by several orders of magnitude. We show that, compared to Ground RCR, Lifted RCR can re-

*Part of this research was conducted while the author was a visiting student at UCLA.

[1] A solver based on the RCR framework won first place in two categories evaluated at the UAI'10 approximate inference challenge (Elidan and Globerson, 2010).

cover many more equivalences in the same amount of time, leading to better approximations.

2 RCR FOR GROUND MLNs

While our main objective is to present a lifted version of the RCR framework, we start by adapting RCR to ground Markov logic networks (MLNs). This is meant to both motivate the specifics of Lifted RCR and to provide a basis for its semantics (i.e., the correctness of Lifted RCR will be against Ground RCR). RCR can be understood in terms of three steps: Relaxation (R), Compensation (C), and Recovery (R). Next, we introduce MLNs and examine each of these steps.

2.1 MARKOV LOGIC NETWORKS

We first introduce some standard concepts from function-free first-order logic. An *atom* $\mathrm{p}(t_1, \ldots, t_n)$ consists of a predicate p/n of arity n followed by n arguments, which are either (lowercase) *constants* or (uppercase) *logical variables*. A formula combines atoms with logical connectives (e.g., $\wedge$, $\Leftrightarrow$). A formula is *ground* if it does not contain any logical variables. The groundings of a formula are the formulas obtained by substituting all variables for constants.

Many probabilistic logical languages have been proposed in recent years. We will work with one such language: Markov logic networks (MLN) (Richardson and Domingos, 2006). An MLN is a set of tuples (w, f), where w is a real number representing a weight and f is a formula in function-free first-order logic. First-order logic formulas without a weight are called *hard formulas* and correspond to formulas with an infinite weight. We will assume that all logical variables in f are universally quantified.[2] Consider the MLN

$$1.3 \quad \mathrm{smokes}(X) \Rightarrow \mathrm{cancer}(X) \tag{1}$$
$$1.5 \quad \mathrm{smokes}(X) \wedge \mathrm{friends}(X, Y) \Rightarrow \mathrm{smokes}(Y) \tag{2}$$

which states that (1) smokers are more likely to get cancer and (2) smokers are more likely to be friends with other smokers (Singla and Domingos, 2008). This MLN will be a running example throughout this paper.

The *grounding* of an MLN Δ is the MLN which results from replacing each formula in Δ with all its groundings (using the same weight). For the domain $\{a, b\}$, the above first-order MLN represents the following ground MLN:

$$1.3 \quad \mathrm{smokes}(a) \Rightarrow \mathrm{cancer}(a) \tag{3}$$
$$1.3 \quad \mathrm{smokes}(b) \Rightarrow \mathrm{cancer}(b)$$
$$1.5 \quad \mathrm{smokes}(a) \wedge \mathrm{friends}(a, a) \Rightarrow \mathrm{smokes}(a)$$
$$1.5 \quad \mathrm{smokes}(a) \wedge \mathrm{friends}(a, b) \Rightarrow \mathrm{smokes}(b)$$
$$1.5 \quad \mathrm{smokes}(b) \wedge \mathrm{friends}(b, a) \Rightarrow \mathrm{smokes}(a)$$
$$1.5 \quad \mathrm{smokes}(b) \wedge \mathrm{friends}(b, b) \Rightarrow \mathrm{smokes}(b)$$

This ground MLN contains eight different random variables, which correspond to all groundings of atoms $\mathrm{smokes}(X)$, $\mathrm{cancer}(X)$ and $\mathrm{friends}(X, Y)$. This leads to a distribution over 2^8 possible worlds. The weight of each world is simply the product of all weights e^w, where (w, f) is a ground MLN formula and f is satisfied by the world. The weights of worlds that do not satisfy a hard formula are set to zero. The probabilities of worlds are obtained by normalizing their weights. The *ground distribution* of Δ is the distribution induced by the grounding of Δ.

2.2 GROUND RELAXATION

Relaxation is the process of ignoring interactions between the formulas of a ground MLN. An interaction takes place when the same ground atom a_g appears in more than one ground formula in an MLN. We can ignore this interaction via a two step process. First, we rename one occurrence of a_g into, say, a'_g, through a process that we call *cloning*. We then assert an equivalence constraint between the original ground atom a_g and its clone, $a_g \Leftrightarrow a'_g$. At this point, we can ignore the interaction by simply dropping the equivalence constraint, through a process that we call *relaxation*. Bringing back the equivalence is known as *recovery* and will be discussed in more detail later.

The $\mathrm{smokes}(a)$ atom in Formula 3 leads to an interaction between this formula and some of the other five formulas in the ground MLN. To ignore this interaction, we first rename this atom occurrence into $\mathrm{smokes}_1(a)$ leading to the modified formula

$$1.3 \quad \mathrm{smokes}_1(a) \Rightarrow \mathrm{cancer}(a) \tag{4}$$

which replaces Formula 3 in the MLN. The corresponding equivalence constraint is

$$\mathrm{smokes}_1(a) \Leftrightarrow \mathrm{smokes}(a) \tag{5}$$

Dropping this constraint amounts to removing the interaction between Formula 4 and the rest of the MLN.

2.3 GROUND COMPENSATION

When relaxing a constraint $a_g \Leftrightarrow a'_g$, we ignore a connection between the ground atoms a_g and a'_g. We can *compensate* for this loss by adding two weighted atoms

$$w: \quad a_g \qquad \text{and} \qquad w': \quad a'_g$$

If the weights w and w' are chosen carefully, one can reestablish a weaker connection between the ground

[2] We transform existential quantifiers into disjunctions.

atoms. For example, one can choose these weights to ensure that the ground atoms have the same probability, establishing a weaker notion of equivalence.

We will now suggest a specific compensation scheme based on a key result from Choi and Darwiche (2006). Suppose that we relax a single equivalence constraint, $a_g \Leftrightarrow a'_g$, which splits the MLN into two disconnected components, one containing atom a_g and another containing atom a'_g. Suppose further that we choose the compensations w and w' such that

$$\Pr(a_g) = \Pr(a'_g) = \frac{e^{w+w'}}{1+e^{w+w'}}. \quad (6)$$

We now have a number of guarantees. First, the resulting MLN will yield exact results when computing the probability of any ground atom. Second, the compensations w and w' can be identified by finding a fixed point for the following equations:

$$\begin{aligned} w_{i+1} &= \log\left(\Pr_i(a'_g)\right) - \log\left(\Pr_i(\neg a'_g)\right) - w'_i \\ w'_{i+1} &= \log\left(\Pr_i(a_g)\right) - \log\left(\Pr_i(\neg a_g)\right) - w_i. \end{aligned} \quad (7)$$

Following Choi and Darwiche (2006), we will seek compensations using these update equations even when the relaxed equivalence constraint does not disconnect the MLN, and even when relaxing multiple equivalence constraints. In this more general case, a fixed-point to the above equations will still guarantee the weak equivalence given in (6). However, when computing the probabilities of ground atoms, we will only get approximations instead of exact results.[3]

Searching for compensations using Equations 7 will lead to the generation of a sequence of MLNs that differ only on the weights of atoms added during the compensation process. The first MLN in this sequence is obtained by using zero weights for all compensating atoms, leading to an initial ground distribution $\Pr_0$. Each application of Equations 7 will then lead to a new MLN (with new compensations) and, hence, a new ground distribution, $\Pr_{i+1}$. Upon convergence, the resulting MLN and its ground distribution will then be used for answering queries. This is typically done using an exact inference algorithm as one usually relaxes enough equivalence constraints to make the ground MLN amenable to exact inference. Note that applying Equations 7 also requires exact inference, as one must compute the probabilities $\Pr_i(a_g)$ and $\Pr_i(a'_g)$.

2.4 GROUND RECOVERY

Now that we can relax equivalences and compensate for their loss, the remaining question is which equivalences to relax. In general, deciding which equivalences to relax is hard, because it requires inference in the original model, which is intractable. Instead, Choi and Darwiche (2006) take the approach of relaxing every equivalence constraint and then incrementally recovering them as time and exact inference allow.

It follows from their results that when (i) relaxing all equivalence constraints, (ii) using the above compensation scheme and (iii) doing exact inference in the approximate model, the approximate marginals found correspond to the approximations found by *iterative belief propagation* (IBP) (Pearl, 1988). The connection to IBP is even stronger: the compensating weights computed in each iteration of Equations 7 exactly correspond to the messages passed by IBP.

Several heuristics have been proposed to decide which equivalences to recover, by doing inference in the relaxed model. We will work with the *residual recovery* heuristic (Choi and Darwiche, 2011). It is based on the practical observation that when IBP converges easily, the quality of its approximation is high. The heuristic tries to recover those constraints that have the most difficulty converging throughout the iterative process, i.e., those that least satisfy Equation 6. We measure this by keeping track of the three-way symmetric KL divergence between the three terms of Equation 6.

3 LIFTED RCR

We now introduce a lifted version of the relax, compensate and recover framework, which is meant to operate directly on first-order MLNs without necessarily having to ground them. Lifted RCR is based on first-order relaxation, compensation and recovery.

3.1 FIRST-ORDER RELAXATION

We will now discuss a first-order notion of relaxation where the goal is to ignore interactions between ground MLN formulas, yet without necessarily having to fully ground the MLN. This requires a first-order version of atom cloning and first-order equivalences.

Definition 1 (First-Order Cloning)**.** Cloning an atom occurrence in an MLN formula amounts to renaming the atom by concatenating its predicate with (i) an identifier of the formula, (ii) an identifier of the occurrence of the atom within the formula, and (iii) the logical variables appearing in the atom's formula.

For example, the first-order cloning of the atom occurrence $\text{smokes}(Y)$ in Formula 2 gives

$$1.5 \quad \text{smokes}(X) \wedge \text{friends}(X,Y) \Rightarrow \text{smokes}_{2b<X,Y>}(Y) \quad (8)$$

[3] In this more general case, there is no longer a guarantee that Equations 7 will converge to a fixed point. Convergence can be improved by using damping in Equations 7.

Here, 2 is an identifier of the formula, b is an identifier of the atom occurrence in the formula, and $<X, Y>$ are the logical variables appearing in the formula.

As in the ground case, each first-order cloning is associated with a corresponding equivalence between the original atom and its clone, except that the equivalence is first-order in this case. The first-order cloning of atom occurrence $\text{smokes}(Y)$ into $\text{smokes}_{2b<X,Y>}(Y)$ in the example above leads to introducing the following first-order equivalence:

$$\text{smokes}(Y) \Leftrightarrow \text{smokes}_{2b<X,Y>}(Y) \tag{9}$$

Let us now consider the groundings of Formulas 8 and 9, assuming a domain of $\{a, b\}$:

$$\begin{aligned}
1.5 \quad & \text{smokes}(a) \wedge \text{friends}(a,a) \Rightarrow \text{smokes}_{2b<a,a>}(a) \\
1.5 \quad & \text{smokes}(a) \wedge \text{friends}(a,b) \Rightarrow \text{smokes}_{2b<a,b>}(b) \\
1.5 \quad & \text{smokes}(b) \wedge \text{friends}(b,a) \Rightarrow \text{smokes}_{2b<b,a>}(a) \\
1.5 \quad & \text{smokes}(b) \wedge \text{friends}(b,b) \Rightarrow \text{smokes}_{2b<b,b>}(b) \\
& \text{smokes}(a) \Leftrightarrow \text{smokes}_{2b<a,a>}(a) \\
& \text{smokes}(b) \Leftrightarrow \text{smokes}_{2b<a,b>}(b) \\
& \text{smokes}(a) \Leftrightarrow \text{smokes}_{2b<b,a>}(a) \\
& \text{smokes}(b) \Leftrightarrow \text{smokes}_{2b<b,b>}(b)
\end{aligned}$$

We have a few observations on the proposed cloning and relaxation techniques. First, the four groundings of (8) contain distinct groundings of the clone $\text{smokes}_{2b<X,Y>}(Y)$. Second, if we relax the equivalence in (9), the ground formulas of (8) will no longer interact through the clone $\text{smokes}_{2b<X,Y>}(Y)$. Third, if we did not append the logical variables $<X, Y>$ during the cloning process, the previous statement would no longer hold. In particular, without appending logical variables, the four groundings of (8) would have contained only the two distinct clone groundings, $\text{smokes}_{2b}(a)$ and $\text{smokes}_{2b}(b)$. This will lead to continued interactions between the four groundings of (8).[4]

The proposed cloning technique leads to MLNs in which one quantifies over predicate names (as in second-order logic). This can be avoided, but it leads to less transparent semantics. In particular, we can avoid quantifying over predicate names by using ground predicate names with increased arity. For example, $\text{smokes}_{2b<X,Y>}(Y)$ could have been written as $\text{smokes}_{2b}(X, Y)$ where we pushed $<X, Y>$ into the predicate arguments. The disadvantage of this, however, is that the semantics of the individual arguments is lost as the arguments become overloaded.

[4]We need to remove all interactions among groundings of the same formula because otherwise the formula might not even be tractable for exact inference. For example, there is currently no exact lifted inference algorithm that can handle the formula $w: \quad \text{p}_a(X, Y) \wedge \text{p}_b(Y, Z) \Rightarrow \text{p}_c(X, Z)$ without grounding it first (Van den Broeck, 2011).

We now have the following key theorem.

Theorem 1. *Let Δ^r be the MLN resulting from cloning all atom occurrences in MLN Δ and then relaxing all introduced equivalences. Let Δ^g be the grounding of Δ^r. The formulas of Δ^g are then fully disconnected (i.e., they share no atoms).*

With this theorem, the proposed first-order cloning and relaxation technique allows one to fully disconnect the grounding of an MLN by simply relaxing first-order equivalences in the first-order MLN.

3.2 FIRST-ORDER COMPENSATION

In principle, one can just clone atom occurrences, relax some equivalence constraints, and then use the resulting MLN as an approximation of the original MLN. By relaxing enough equivalences, the approximate MLN can be made arbitrarily easy for exact inference. Our goal in this section, however, is to improve the quality of approximations by compensating for the relaxed equivalences, yet without making the relaxed MLN any harder for exact inference. This will be done through a notion of first-order compensation.

3.2.1 Equiprobable Equivalences

The proposed technique is similar to the one for ground MLNs, that is, using *weighted atoms* whose weights will allow for compensation. The key, however, is to use first-order weighted atoms instead of ground ones. For this, we need to define the following notions.

Definition 2 (Equiprobable Set)**.** A set of random variables V is called equiprobable w.r.t. distribution Pr iff for all $v_1, v_2 \in V : \Pr(v_1) = \Pr(v_2)$.

Definition 3 (Equiprobable Equivalence)**.** Let Δ be an MLN from which a first-order equivalence $a \Leftrightarrow a'$ was relaxed. Let $a_1 \Leftrightarrow a'_1, \ldots, a_n \Leftrightarrow a'_n$ be all groundings of $a \Leftrightarrow a'$. The equivalence $a \Leftrightarrow a'$ is equiprobable iff the sets $\{a_1, \ldots, a_n\}$ and $\{a'_1, \ldots, a'_n\}$ are both equiprobable w.r.t the ground distribution of MLN Δ.

The basic idea of first-order compensation is that when relaxing an equiprobable equivalence $a \Leftrightarrow a'$, under certain conditions, one can compensate for its loss using only two weighted first-order atoms of the form:

$$w: \quad a \qquad \text{and} \qquad w': \quad a'$$

This follows because if we were to fully ground the equivalence into $a_1 \Leftrightarrow a'_1, \ldots, a_n \Leftrightarrow a'_n$ and then apply ground compensation, the relevant ground atoms will attain the same weights. That is, by the end of ground compensation, the weighted ground atoms,

$$w_i: \quad a_i \qquad \text{and} \qquad w'_i: \quad a'_i$$

will have $w_i = w_j$ and $w'_i = w'_j$ for all i and j.

3.2.2 Partitioning Equivalences

To realize first-order compensation, one must address two issues. First, a relaxed first-order equivalence may not be equiprobable to start with. Second, even when the equivalence is equiprobable, it may cease to be equiprobable as we adjust the weights during the compensation process. Recall that equiprobability is defined with respect to the ground distribution of an MLN. Yet, this distribution changes during the compensation process, which iteratively changes the weights of compensating atoms and, hence, also iteratively changes the ground distribution.

Consider for example the following relaxed equivalences: $\mathrm{p}(X) \Leftrightarrow \mathrm{q}(X)$ and $\mathrm{q}(X) \Leftrightarrow \mathrm{r}(X)$. Suppose the domain is $\{a, b\}$ and the current ground distribution, $\Pr_i$, is such that $\Pr_i(\mathrm{p}(a)) = \Pr_i(\mathrm{p}(b))$, $\Pr_i(\mathrm{q}(a)) = \Pr_i(\mathrm{q}(b))$, and $\Pr_i(\mathrm{r}(a)) \neq \Pr_i(\mathrm{r}(b))$. In this case, the equivalence $\mathrm{p}(X) \Leftrightarrow \mathrm{q}(X)$ is equiprobable, but $\mathrm{q}(X) \Leftrightarrow \mathrm{r}(X)$ is not equiprobable.

If an equivalence constraint is not equiprobable, one can always partition it into a set of equiprobable equivalences — in the worst case, the partition will include all groundings of the equivalence. In the above example, one can partition the equivalence $\mathrm{q}(X) \Leftrightarrow \mathrm{r}(X)$ into the equivalences $\mathrm{q}(a) \Leftrightarrow \mathrm{r}(a)$ and $\mathrm{q}(b) \Leftrightarrow \mathrm{r}(b)$, which are trivially equiprobable.

Given this partitioning, the compensation algorithm will add distinct weights for the compensating atoms $\mathrm{q}(a)$ and $\mathrm{q}(b)$. Therefore, the set $\{\mathrm{q}(a), \mathrm{q}(b)\}$ will no longer be equiprobable in the next ground distribution, $\Pr_{i+1}$. As a result, the equivalence $\mathrm{p}(X) \Leftrightarrow \mathrm{q}(X)$ will no longer be equiprobable w.r.t. the ground distribution $\Pr_{i+1}$, even though it was equiprobable with respect to the previous ground distribution $\Pr_i$.

3.2.3 Strongly Equiprobable Equivalences

To attain the highest degree of lifting during compensation, one needs to dynamically partition equivalences after each iteration of the compensation algorithm, to ensure equiprobability. We defer the discussion on dynamic partitioning to Appendix A, focusing here on a strong version of equiprobability that allows one to circumvent the need for dynamic partitioning.

The mentioned technique is employed by our current implementation of Lifted RCR, which starts with equivalences that are *strongly equiprobable.* An equivalence is strongly equiprobable if it is equiprobable w.r.t all ground distributions induced by the compensation algorithm (i.e., ground distributions that result from only modifying the weights of compensating atoms).

Consider again Formula 2 where we cloned the atom occurrence $\mathrm{smokes}(Y)$ and relaxed its equivalence, leading to the MLN:

$$1.5\ \ \mathrm{smokes}(X) \wedge \mathrm{friends}(X, Y) \Rightarrow \mathrm{smokes}_{2b<X,Y>}(Y)$$

and relaxed equivalence

$$\mathrm{smokes}(Y) \Leftrightarrow \mathrm{smokes}_{2b<X,Y>}(Y) \tag{10}$$

Suppose we partition this equivalence as follows:[5]

$$X = Y,\ \ \mathrm{smokes}(Y) \Leftrightarrow \mathrm{smokes}_{2b<X,Y>}(Y)$$
$$X \neq Y,\ \ \mathrm{smokes}(Y) \Leftrightarrow \mathrm{smokes}_{2b<X,Y>}(Y)$$

These equivalences are not only equiprobable w.r.t. the relaxed MLN, but also strongly equiprobable. That is, suppose we add to the relaxed model the compensating atoms

$$w_1 :\ \ \mathrm{smokes}(X)$$
$$w_1' :\ X = Y,\ \ \mathrm{smokes}_{2b<X,Y>}(Y)$$
$$w_2 :\ \ \mathrm{smokes}(X)$$
$$w_2' :\ X \neq Y,\ \ \mathrm{smokes}_{2b<X,Y>}(Y)$$

The two equivalences will be equiprobable w.r.t. any ground distribution that results from adjusting the weights of these compensating atoms. We will present an equivalence partitioning algorithm in Section 4 that guarantees strong equiprobability of the partitioned equivalences. This algorithm is employed by our current implementation of Lifted RCR and will be used when reporting experimental results later.

3.3 COUNT-NORMALIZATION

We are one step away from presenting our first-order compensation scheme. What is still missing is a discussion of *count-normalized* equivalences, which are also required by our compensation scheme.

Consider Equivalence 10, which has four groundings

$$\mathrm{smokes}(a) \Leftrightarrow \mathrm{smokes}_{2b<a,a>}(a)$$
$$\mathrm{smokes}(b) \Leftrightarrow \mathrm{smokes}_{2b<a,b>}(b)$$
$$\mathrm{smokes}(a) \Leftrightarrow \mathrm{smokes}_{2b<b,a>}(a)$$
$$\mathrm{smokes}(b) \Leftrightarrow \mathrm{smokes}_{2b<b,b>}(b)$$

for the domain $\{a, b\}$. There are two distinct groundings of the original atom $\mathrm{smokes}(Y)$ in this case and each of them appears in two groundings. When each grounding of the original atom appears in exactly the

[5]We are using an extension of MLNs that allows constraints, such as $X \neq Y$. Our implementation is in terms of parfactor graphs, which are more expressive than standard MLNs and do allow for the representation of such constraints. In extended MLNs, we will write $C,\ f$ to mean that C is a constraint that applies to formula f.

same number of ground equivalences, we say that the first-order equivalence is count-normalized.

Consider now a constrained version of Equivalence 10

$$X \neq b \vee Y \neq b,\ \text{smokes}(Y) \Leftrightarrow \text{smokes}_{2b<X,Y>}(Y)$$

which has the following groundings

$$\text{smokes}(a) \Leftrightarrow \text{smokes}_{2b<a,a>}(a)$$
$$\text{smokes}(b) \Leftrightarrow \text{smokes}_{2b<a,b>}(b)$$
$$\text{smokes}(a) \Leftrightarrow \text{smokes}_{2b<b,a>}(a)$$

This constrained equivalence is not count-normalized since the atom smokes(a) appears in two ground equivalences while the atom smokes(b) appears in only one. More generally, we have the following definition.

Definition 4. Let $C,\ a \Leftrightarrow a'$ be a first-order equivalence. Let α be an instantiation of the variables in original atom a and assume that α satisfies constraint C. The equivalence is *count-normalized* iff $C \wedge \alpha$ has the same number of solutions for each instantiation α. Moreover, the number of groundings for $C,\ a$ is called the *original count* and the number of groundings for $C,\ a'$ is called the *clone count.*

Count-normalization can only be violated by constrained equivalences. Moreover, for a certain class of constraints, count-normalization is always preserved. The algorithm we shall present in Section 4 for partitioning equivalences takes advantage of this observation. In particular, the algorithm generates constrained equivalences whose constraint structure guarantees count-normalization.

3.4 THE COMPENSATION SCHEME

We now have the following theorem.

Theorem 2. *Let Δ_i be an MLN with relaxed equivalences $C,\ a \Leftrightarrow a'$ and, hence, corresponding compensating atoms:*

$$w_i:\quad C,\ a \qquad \text{and} \qquad w_i':\quad C,\ a'$$

Suppose that the equivalences are count-normalized and strongly equiprobable. Let $a_g \Leftrightarrow a_g'$ be one grounding of equivalence $C,\ a \Leftrightarrow a'$, let n be its original count and n' be its clone count. Consider now the MLN Δ_{i+1} obtained using the following updates:

$$w_{i+1} = \frac{n'}{n}\left(\log\left(\Pr_i(a_g')\right) - \log\left(\Pr_i(\neg a_g')\right) - w_i'\right)$$
$$w_{i+1}' = \log\left(\Pr_i(a_g)\right) - \log\left(\Pr_i(\neg a_g)\right) - w_i \qquad (11)$$

The ground distribution of MLN Δ_{i+1} equals the one obtained by applying Ground RCR to MLN Δ_i.

Note that first-order compensation requires exact inference on the MLN Δ_i, which is needed for computing $\Pr_i(a_g)$ and $\Pr_i(a_g')$. Moreover, these computations will need to be repeated until one obtains a fixed point of the update equations given by Theorem 2.

3.5 FIRST-ORDER RECOVERY

Recovering a first-order equivalence $C,\ a \Leftrightarrow a'$ amounts to removing its compensating atoms

$$w_i:\quad C,\ a \qquad \text{and} \qquad w_i':\quad C,\ a'$$

and then adding the equivalence back to the MLN.

Adapting the ground recovery heuristic suggested earlier, one recovers the first-order equivalence that maximizes the symmetric pairwise KL-divergence

$$n' \cdot \text{KLD}\left(\Pr(a_g), \Pr(a_g'), \frac{e^{w_i + w_i'}}{1 + e^{w_i + w_i'}}\right),$$

where n' is the clone count of the equivalence. Note here that n' is also the number of equivalence groundings since, by definition, the clone atom contains all logical variables that appear in the equivalence.

Note that recovering first-order equivalences may violate the equiprobability of equivalences that remain relaxed, which in turn may require re-partitioning.

4 PARTITIONING EQUIVALENCES

We will now discuss a method for partitioning first-order equivalences, which guarantees both strong equiprobability and count-normalization. This method is used by our current implementation of Lifted RCR that we describe in Section 6.

Our method is based on the procedure of *preemptive shattering* given by Poole et al. (2011), which is a conceptually simpler version of the influential *shattering* algorithm proposed by Poole (2003) and de Salvo Braz et al. (2005) in the context of exact lifted inference. We will first describe this shattering procedure, which partitions atoms. We will then use it to partition all atoms in a relaxed MLN. We will finally show how these atom partitions can be used to partition first-order equivalences.

4.1 PREEMPTIVE SHATTERING

Preemptive shattering takes as input an atom $p(X_1, \ldots, X_n)$ and a set of constants $K = \{k_1, \ldots, k_m\}$. It then returns a set of constrained atoms of the form $C,\ p(X_1, \ldots, X_n)$ which represent a partitioning of the input atom. That is, the groundings of constrained atoms are guaranteed to be disjoint and cover all groundings of the input atom.

We start with an intuitive description of preemptive shattering. The set of constants K are the ones explicitly mentioned in the MLN of interest. If an argument X_i of the input atom $p(X_1, \ldots, X_n)$ can take on one of the constants $k_j \in K$, preemptive shattering splits the atom into two constrained atoms: one where X_i is substituted by constant k_i and one where the constant k_j is excluded from the domain of X_i. Moreover, when two arguments X_i and X_j of the input atom can take on the same value, preemptive shattering splits the atom into two constrained atoms: one with the constraint $X_i = X_j$ and another with $X_i \neq X_j$.

We will next describe the shattering procedure and its complexity more formally. We need some definitions first. For each argument X_i of the input atom, let

$$C_i = \{(X_i = k_1), \ldots, (X_i = k_m), \\ (X_i \neq k_1, \ldots, X_i \neq k_m)\}.$$

Preemptive shattering generates all possible combinations of the above constraints:

$$\mathbf{C}_A = \{c_1 \wedge \cdots \wedge c_n \mid (c_1, \ldots, c_n) \in \times_{i=1}^{n} C_i\},$$

where $\times_{i=1}^{n} C_i$ is the cartesian product of $C_1, \ldots, C_n$.

Consider now a subset $\mathcal{X} \subseteq \{X_1, \ldots, X_n\}$ of the input atom arguments and let

$$\mathbf{C}(\mathcal{X}) = \bigwedge_{X_i, X_j \in \mathcal{X}} (X_i = X_j) \wedge \bigwedge_{i=1}^{n} \bigwedge_{X_j \notin \mathcal{X}} (X_i \neq X_j)$$
$$\mathbf{C}_B = \{\mathbf{C}(\mathcal{X}) \mid \mathcal{X} \subseteq \{X_1, \ldots, X_n\}\}$$

Preemptive shattering will then create the following partition of the input atom $p(X_1, \ldots, X_n)$:

$$A = \{c_a \wedge c_b,\ \mathrm{p}(X_1, \ldots, X_n) \mid c_a \in \mathbf{C}_A, c_b \in \mathbf{C}_B\}$$

Many of the generated constraints $c_a \wedge c_b$ will have no solutions and can be dropped. What remains is a set of constrained atoms whose groundings form a partition of the groundings for the input atom. Preemptive shattering can be implemented in time that is exponential in the arity n of input atom and polynomial in the number of input constants m.

For an example, consider the formula $\mathrm{smokes}(X) \Leftrightarrow \mathrm{smokes}_{<X,Y>}(X)$ and assume we have evidence $\mathrm{smokes}(a)$ and therefore $K = \{a\}$. Preemptive shattering of the input atom $\mathrm{smokes}(X)$ returns back

$$X = a,\ \mathrm{smokes}(X)$$
$$X \neq a,\ \mathrm{smokes}(X)$$

Preemptive shattering of the input atom $\mathrm{smokes}_{<X,Y>}(X)$ returns back

$$X = a, Y = a,\ \mathrm{smokes}_{<X,Y>}(X)$$
$$X = a, Y \neq a,\ \mathrm{smokes}_{<X,Y>}(X)$$
$$X \neq a, Y = a,\ \mathrm{smokes}_{<X,Y>}(X)$$
$$X \neq a, Y \neq a, X = Y,\ \mathrm{smokes}_{<X,Y>}(X)$$
$$X \neq a, Y \neq a, X \neq Y,\ \mathrm{smokes}_{<X,Y>}(X)$$

We will next show how this shattering procedure forms the basis of a method for partitioning equivalence constraints, with the aim of ensuring both strong equiprobability and count-normalization.

4.2 PARTITIONING EQUIVALENCES BY PREEMPTIVE SHATTERING

Consider an MLN which results from cloning some atom occurrences and then adding corresponding equivalence constraints. Let K be all the constants appearing explicitly in the MLN.

To partition a first-order equivalence $a \Leftrightarrow a'$, our method will first apply preemptive shattering to the original atom a and clone atom a', yielding a partition for each. Suppose that $C_1,\ a_1, \ldots, C_n,\ a_n$ is the partition returned for original atom a. Suppose further that $C'_1,\ a'_1, \ldots, C'_m,\ a'_m$ is the partition returned for clone atom a'. By definition of cloning, all variables that appear in original atom a must also appear in clone atom a'. This implies the following property. For every (original) constraint C_i, there is a corresponding set of (clone) constraints C'_j that partition C_i. Each pair of constraints C_i and C'_j will then generate a member of the equivalence partition: $C_i \wedge C'_j,\ a_i \Leftrightarrow a'_j$. Note that $C_i \wedge C'_j = C'_j$ since C'_j implies C_i.

Let us consider an example. Suppose we are partitioning the equivalence $\mathrm{smokes}(X) \Leftrightarrow \mathrm{smokes}_{<X,Y>}(X)$. Section 4.1 showed the preemptive shattering of the atoms $\mathrm{smokes}(X)$ and $\mathrm{smokes}_{<X,Y>}(X)$. These give rise to the following equivalence partition:

$$X = a, Y = a,\ \mathrm{smokes}(X) \Leftrightarrow \mathrm{smokes}_{<X,Y>}(X)$$
$$X = a, Y \neq a,\ \mathrm{smokes}(X) \Leftrightarrow \mathrm{smokes}_{<X,Y>}(X)$$
$$X \neq a, Y = a,\ \mathrm{smokes}(X) \Leftrightarrow \mathrm{smokes}_{<X,Y>}(X)$$
$$X \neq a, Y \neq a, X = Y,\ \mathrm{smokes}(X) \Leftrightarrow \mathrm{smokes}_{<X,Y>}(X)$$
$$X \neq a, Y \neq a, X \neq Y,\ \mathrm{smokes}(X) \Leftrightarrow \mathrm{smokes}_{<X,Y>}(X)$$

The following result is proven in Appendix B.

Theorem 3. *Partitioning by preemptive shattering returns count-normalized, strongly equiprobable equivalences.*

When K is small, preemptive shattering will find partitions that are close to minimal. When K is large, however, it will create large partitions, defeating the

purpose of lifted inference. In the next section, we will mention some alternative partitioning algorithms that work on a fully relaxed model. However, preemptive shattering is the only partitioning algorithm to our knowledge that works for any level of relaxation. We believe our work can motivate future work on finding more efficient general partitioning algorithms and even approximate partitioning algorithms.

5 RELATED WORK

The RCR framework has previously been used to characterize iterative belief propagation (IBP) and some of its generalizations. In the case where the simplified model is fully disconnected, the approximate marginals of RCR correspond to the approximate marginals given by IBP (Pearl, 1988; Choi and Darwiche, 2006). The approximation to the partition function further corresponds to the *Bethe free energy* approximation (Yedidia et al., 2003; Choi and Darwiche, 2008). When equivalence constraints have been recovered, RCR corresponds to a class of *generalized belief propagation* (GBP) approximations (Yedidia et al., 2003), and in particular *iterative joingraph propagation* (Aji and McEliece, 2001; Dechter et al., 2002). RCR also inspired a system that was successfully employed in a recent approximate inference competition (Elidan and Globerson, 2010; Choi and Darwiche, 2011). *Mini-buckets* can also be viewed as an instance of RCR where no compensations are used (Kask and Dechter, 2001; Dechter and Rish, 2003; Choi et al., 2007), which leads to upper bounds on the partition function (for example). Any approximate MLN found by Lifted RCR corresponds to one found by Ground RCR on the ground MLN, thus all of the above results carry over to the lifted setting.

The motivation for calling our approach *lifted* is threefold. First, in the compensation phase, we are compensating for many ground equivalences at the same time. Computing compensating weights for all of these requires inferring only a *single* pair of marginal probabilities. Second, computing marginal probabilities is done by an *exact lifted* inference algorithm. Third, we relax and recover first-order equivalence constraints, which correspond to *sets* of ground equivalences.

The work on lifted approximate inference has mainly focused on lifting the IBP algorithm. The correspondence between IBP and RCR carries over to their lifted counterparts: Lifted RCR compensations on a fully relaxed model correspond to *lifted belief propagation* (Singla and Domingos, 2008). Starting from a first-order model, Singla and Domingos (2008) proposed *lifted network construction* (LNC), which partitions atoms into so-called *supernodes*. The ground atoms represented by these supernodes send and receive the same messages when running IBP. This means that they partition the atoms into equiprobable sets and that LNC can be used for equivalence partitioning in Lifted RCR for the fully relaxed model. Kersting et al. (2009) proposed a *color-passing* (CP) algorithm that achieves similar results as LNC, only starting from a ground model, where the first-order structure is not apparent. Two other approximate lifted inference algorithms are *probabilistic theorem proving* (Gogate and Domingos, 2011), which contains a Monte-Carlo method and *bisimulation-based lifted inference* (Sen et al., 2009), which uses a mini-bucket approximation on a model that was compressed by detecting symmetries. Because of the correspondence between Ground RCR and mini-buckets mentioned above, this approach can also be seen as an instance of Lifted RCR with the compensation phase removed.

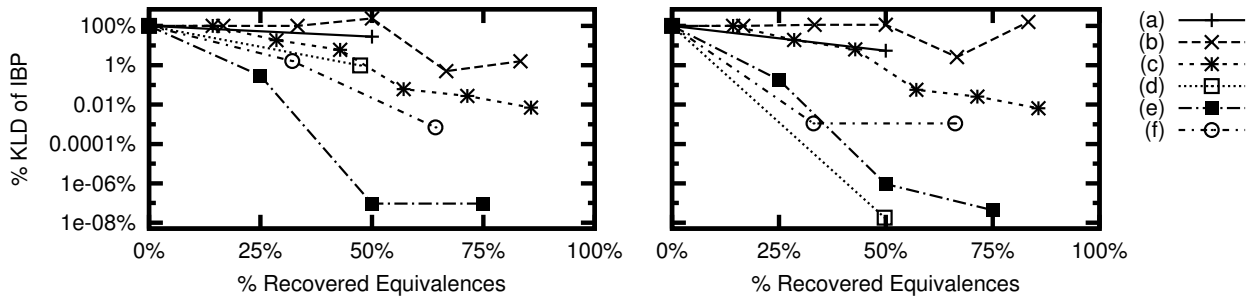
6 EXPERIMENTS

In this section, we evaluate the Lifted RCR algorithm on common benchmarks from the lifted inference literature. The experiments were set up to answer the questions: **(Q1)** To which extent does recovering first-order equivalences improve the approximations found by Lifted RCR? **(Q2)** Can IBP be improved considerably through the recovery of a small number of equivalences? **(Q3)** Is there a significant advantage to using Lifted RCR over Ground RCR?

We implemented Lifted RCR and released it as open source software.[6] To compute exact marginal probabilities in Equations 11, we use first-order knowledge compilation (Van den Broeck et al., 2011). It compiles the MLN into a logical circuit where probabilistic inference is performed by weighted model counting, which exploits context-specific independencies and determinism in the MLN. It is arguably the state of the art in exact lifted inference (Van den Broeck, 2011; Van den Broeck and Davis, 2012). In combination with preemptive shattering, we compile a first-order circuit once and re-evaluate it in each iteration of the compensation algorithm. This is possible because the structure of the compensated MLNs do not change between iterations, only their parameters change. An already compiled first-order circuit can re-evaluated very efficiently. See Appendix C for further details.

To answer **(Q1-2)** we ran Lifted RCR on MLNs from the exact lifted inference literature, where computing exact marginals is tractable. This allows us to evaluate the approximation quality of Lifted RCR for different degrees of relaxation. We used the models *p-r* and *sick-death* (de Salvo Braz et al., 2005), *work-*

[6] `http://dtai.cs.kuleuven.be/ml/systems/wfomc`



(a) Small domains - 100 random variables

(b) Large Domains - 10,000 random variables

Figure 1: Normalized approximation error of Lifted RCR for different levels of approximation on the models (a) *p-r*, (b) *sick-death*, (c) *workshop attributes*, (d) *smokers*, (e) *smokers and drinkers* and (f) *symmetric smokers*.

shop attributes (Milch et al., 2008), *smokers* (Singla and Domingos, 2008), *smokers and drinkers* (Van den Broeck et al., 2011) and *symmetric smokers* (Van den Broeck, 2011). Each of these models represents a new advance in exact lifted inference. They are incrementally more complex and challenging for lifted inference. The results are shown in Figure 1, where we ran Lifted RCR on the above models for two sets of domain sizes: a small and a large set, where the number of random variables is on the order of 100 and 10,000 respectively. We plot the symmetric KL divergence between the exact marginals and the approximations found by Lifted RCR, as a percentage of the KL divergence of the fully relaxed approximation. The horizontal axis shows the level of relaxation in terms of the percentage of recovered ground equivalences. The 0% point corresponds to the approximations found by (lifted) IBP. The 100% point corresponds to exact inference.[7]

We see that each recovered first-order equivalence tends to improve the approximation quality significantly, often by more than one order of magnitude, answering **(Q1)**. In the case of *smokers* with a large domain size, recovering a single equivalence more than the IBP approximation reduced the KL divergence by 10 orders of magnitude, giving a positive answer to **(Q2)**. The *sick-death* model is the only negative case for **(Q2)**, where recovering equivalences does not lead to approximations that are better than IBP.[8]

To answer **(Q3)**, first note that as argued in Section 5, the 0% recovery point of Lifted RCR using LNC or CP to partition equivalences corresponds to lifted IBP. For this case, the work of Singla and Domingos (2008) and Kersting et al. (2009) has extensively shown that Lifted IBP/RCR methods can significantly outperform Ground IBP/RCR. Similarly, computational gains for the 100% recovery point were shown in the exact lifted inference literature. For intermediate levels of relaxation, we ran Ground RCR on the above models with large domain sizes. On these, Ground RCR could never recover more than 5% of the relaxed equivalences before exact inference in the relaxed model becomes intractable. This answers **(Q3)** positively.

For the above experiments, Appendix D further reports on the quality of the approximations and the convergence of the compensation algorithm, both as a function of runtime and the number of iterations.

7 CONCLUSIONS

We presented Lifted RCR, a lifted approximate inference algorithm that performs exact inference in a simplified model. We showed how to obtain a simplified model by relaxing first-order equivalences, compensating for their loss, and recovering them as long as exact inference remains tractable. The algorithm can traverse an entire spectrum of approximations, from lifted iterative belief propagation to exact lifted inference. Inside this spectrum is a family of lifted join-graph propagation (and GBP) approximations. We empirically showed that recovering first-order equivalences in a relaxed model can substantially improve the quality of an approximation. We also remark that Lifted RCR relies on an exact lifted inference engine as a black box, and that any future advances in exact lifted inference have immediate impact in lifted approximate inference.

Acknowledgements

This work has been partially supported by ONR grant #N00014-12-1-0423, NSF grant #IIS-1118122, and NSF grant #IIS-0916161. GVdB is supported by the Research Foundation-Flanders (FWO-Vlaanderen).

[7] All experiments ran up to the 100% point, which is not shown in the plot because it has a KL divergence of 0.

[8] Interestingly, it is also the only example where some compensations failed to converge without using damping.

References

S. M. Aji and R. J. McEliece. The generalized distributive law and free energy minimization. In *Proceedings of the 39th Allerton Conference on Communication, Control and Computing*, pages 672–681, 2001.

A. Choi and A. Darwiche. An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence,*, pages 1107–1114, 2006.

A. Choi and A. Darwiche. Approximating the partition function by deleting and then correcting for model edges. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 79–87, 2008.

A. Choi and A. Darwiche. Relax, compensate and then recover. In T. Onada, D. Bekki, and E. McCready, editors, *New Frontiers in Artificial Intelligence*, volume 6797 of *Lecture Notes in Computer Science*, pages 167–180. Springer Berlin / Heidelberg, 2011.

A. Choi, M. Chavira, and A. Darwiche. Node splitting: A scheme for generating upper bounds in Bayesian networks. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 57–66, 2007.

L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, editors. *Probabilistic inductive logic programming: theory and applications*. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 3-540-78651-1, 978-3-540-78651-1.

R. de Salvo Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI),*, pages 1319–1325, 2005.

R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, 50(2):107–153, 2003.

R. Dechter, K. Kask, and R. Mateescu. Iterative join-graph propagation. In *Proceedings of the 18th conference on uncertainty in artificial intelligence (UAI)*, pages 128–136, 2002.

G. Elidan and A. Globerson. Summary of the 2010 UAI approximate inference challenge. http://www.cs.huji.ac.il/project/UAI10/, 2010.

L. Getoor and B. Taskar, editors. *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

V. Gogate and P. Domingos. Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 256–265, 2011.

K. Kask and R. Dechter. A general scheme for automatic generation of search heuristics from specification dependencies. *Artificial Intelligence*, 129(1-2): 91–131, 2001.

K. Kersting, B. Ahmadi, and S. Natarajan. Counting belief propagation. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 277–284. AUAI Press, 2009.

K. Kersting, Y. El Massaoudi, B. Ahmadi, and F. Hadiji. Informed lifting for message-passing. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence,*. AAAI Press, 2010.

B. Milch, L. Zettlemoyer, K. Kersting, M. Haimes, and L. Kaelbling. Lifted probabilistic inference with counting formulas. *Proceedings of the 23rd AAAI Conference on Artificial Intelligence,*, pages 1062–1068, 2008.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

D. Poole. First-order probabilistic inference. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI),*, pages 985–991, 2003.

D. Poole, F. Bacchus, and J. Kisynski. Towards completely lifted search-based probabilistic inference. *CoRR*, abs/1107.4035, 2011.

M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1):107–136, 2006.

P. Sen, A. Deshpande, and L. Getoor. Bisimulation-based approximate lifted inference. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 496–505. AUAI Press, 2009.

P. Singla and P. Domingos. Lifted first-order belief propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence,*, pages 1094–1099, 2008.

N. Taghipour, D. Fierens, J. Davis, and H. Blockeel. Lifted variable elimination with arbitrary constraints. In *Proceedings of the fifteenth international conference on artificial intelligence and statistics*, 2012.

G. Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Proceedings of the 25th Conference on Neural Information Processing Systems (NIPS)*, Dec. 2011.

G. Van den Broeck and J. Davis. Conditioning in first-order knowledge compilation and lifted probabilistic inference. In J. Hoffmann and B. Selman, editors,

Proceedings of the 26th AAAI Conference on Artificial Intelligence,. AAAI Press, July 2012. to appear.

G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, and L. De Raedt. Lifted Probabilistic Inference by First-Order Knowledge Compilation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 2178–2185, 2011.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*, chapter 8, pages 239–269. Morgan Kaufmann, 2003.

Leveraging Side Observations in Stochastic Bandits

Stéphane Caron
Technicolor
735 Emerson St
Palo Alto, CA 94301

Branislav Kveton
Technicolor
735 Emerson St
Palo Alto, CA 94301

Marc Lelarge
INRIA–ENS
45 rue d'Ulm
75005 Paris

Smriti Bhagat
Technicolor
735 Emerson St
Palo Alto, CA 94301

Abstract

This paper considers stochastic bandits *with side observations*, a model that accounts for both the exploration/exploitation dilemma and relationships between arms. In this setting, after pulling an arm i, the decision maker also observes the rewards for some other actions related to i. We will see that this model is suited to content recommendation in social networks, where users' reactions may be endorsed or not by their friends. We provide efficient algorithms based on upper confidence bounds (UCBs) to leverage this additional information and derive new bounds improving on standard regret guarantees. We also evaluate these policies in the context of movie recommendation in social networks: experiments on real datasets show substantial learning rate speedups ranging from 2.2× to 14× on dense networks.

1 INTRODUCTION

In the classical *stochastic multi-armed bandit* problem [4, 6], a decision maker repeatedly chooses among a finite set of K actions. At each time step t, the action i chosen yields a random reward $X_{i,t}$ drawn from a probability distribution proper to action i and unknown to the decision maker. Her goal is to maximize her cumulative expected reward over the sequence of chosen actions. This problem has received well-deserved attention from the online learning community for the simple model it provides of a tradeoff between exploration (trying out all actions) and exploitation (selecting the best action so far). It has several applications, including content recommendation, Internet advertising and clinical trials.

The decision maker's performance after n steps is typically measured in terms of the *regret* $R(n)$, defined as the difference between the reward of her strategy and that of an optimal strategy (one that would always choose actions with maximum expected reward). One of the most prominent algorithms in the stochastic bandit literature, UCB1 from Auer et al. [6], achieves a logarithmic (expected) regret

$$\mathbb{E}\left[R(n)\right] \leq A_{\mathsf{UCB1}} \ln n + B_{\mathsf{UCB1}}, \tag{1}$$

where A_{UCB1} and B_{UCB1} are two constants specific to the policy. This upper bound implies fast convergence to an optimal policy: the mean loss per decision after n rounds is only $\mathbb{E}\left[R(n)/n\right] = O(\ln n/n)$ in expectation (which scaling is known to be optimal).

This paper considers the stochastic bandit problem *with side observations* (a setting that has been considered in [16] but for adversarial bandits, see Section 2), a generalization of the standard multi-armed bandit where playing an action i at step t not only results in the reward $X_{i,t}$, but also yields information on some related actions $\{X_{j,t}\}$. We also present a direct application of this scenario is advertising in social networks: a content provider may target users with promotions (*e.g.*, "20% off if you buy this movie and post it on your wall"), get a reward if the user reacts positively, but also observe her connections' feelings toward the content (*e.g.*, friends reacting by "Liking" it or not).

Our contributions are as follows. First, we consider a generalization UCB-N of UCB1 taking side observations into account. We show that its regret can be upper bounded as in (1) with a smaller $A_{\mathsf{UCB\text{-}N}} < A_{\mathsf{UCB1}}$. Then, we provide a better algorithm UCB-MaxN achieving an improved constant term $B_{\mathsf{UCB\text{-}MaxN}} < B_{\mathsf{UCB\text{-}N}}$. We show that both improvements are significant for bandits with a large number of arms and a dense reward structure, as is for example the case of advertising in social networks. We finally evaluate our policies on real social network datasets and observe substantial learning rate speedups (from 2.2× to 14×, see Section 5).

2 RELATED WORK

Multi-armed bandit problems became popular with the seminal paper of Robbins [19] in 1952. Thirty years later, Lai and Robbins [13] provided one of the key results in this literature when they showed that, asymptotically, the expected regret for the stochastic problem has to grow at least logarithmically in the number of steps, *i.e.*,

$$\mathbb{E}\left[R(n)\right] = \Omega(\ln n).$$

They also introduced an algorithm that follows the "optimism in the face of uncertainty" principle and decides which arm to play based on *upper confidence bounds* (UCBs). Their solution asymptotically matches the logarithmic lower bound.

More recently, Auer et al. [6] considered the case of bandits with *bounded* rewards and introduced the well-known UCB1 policy, a concise strategy achieving the optimal logarithmic bound *uniformly* over time instead of asymptotically. Further work improved the constants A_{UCB1} and B_{UCB1} in their upper bound (1) using additional statistical assumptions [3].

One of the major limitations of standard bandit algorithms appears in situations where the number of arms K is large or potentially infinite; note for instance that the upper bound (1) scales linearly with K. One approach to overcome this difficulty is to add *structure* to the rewards distributions by embedding arms in a metric space and assuming that close arms share a similar reward process. This is for example the case in dependent bandits [18], where arms with close expected rewards are clustered.

$\mathcal{X}$-armed bandits [7] allow for an infinite number of arms $\mathbf{x}$ living in a measurable space $\mathcal{X}$. They assume that the mean reward function $\mu : \mathbf{x} \mapsto \mathbb{E}\left[X_{\mathbf{x}}\right]$ satisfies some Lipschitz assumptions and extend the bias term in UCBs accordingly. Bubeck et al. [7] provide a tree-based optimization algorithm that achieves, under proper assumptions, a regret independent of the dimension of the space.

Linear bandits [8, 20] are another example of structured bandit problems with infinitely many arms. In this setting, arms $\mathbf{x}$ live in a finite-dimensional vector space and mean rewards are modeled as linear functions of a system-wide parameter $\mathbf{Z} \in \mathbb{R}^r$, *i.e.*, $\mathbb{E}\left[X_{\mathbf{x}}\right] = \mathbf{Z} \cdot \mathbf{x}$. Near-optimal policies typically extend the notion of confidence intervals to *confidence ellipsoids*, estimated through empirical covariance matrices, and use the radius of these confidence regions as the bias term in their UCBs.

This last framework allows for contextual bandits and has been used as such in advertisement and content recommendation settings: [14] applied it to personalized news article recommendation, while [9] extended it to generalized linear models[1] and applied it to Internet advertisement. The approach in both these works is to reduce a large number of arms to a small set of numerical features, and then apply a linear bandit policy in the reduced space. Constructing good features is thus a crucial and challenging part of this process. In this paper, we do not make any assumption on the structure of the reward space. We handle the large number of arms in multi-armed bandits leveraging a phenomenon known as *side observations* which occurs in a variety of problems. This phenomenon has already been studied by Mannor et al. [16] in the case of *adversarial* bandits, *i.e.*, where the reward sequence $\{X_{i,t}\}$ is arbitrary and no statistical assumptions are made. They proposed two algorithms: ExpBan, a mix of experts and bandits algorithms based on a clique decomposition of the side observations graph, and ELP, an extension of the well-known EXP3 algorithm [5] taking the side observation structure into account. While the clique decomposition in ExpBan inspired our present work, our setting is that of *stochastic* bandits: statistical assumptions on the reward process allow us to derive $O(\ln n)$ regret bounds, while the best achievable bounds in the adversarial problem are $\widetilde{O}(\sqrt{n})$. It is indeed much harder to learn in an adversarial environment, and the methodology to address this family of problems is quite different from the techniques we use in our work.

Note that our side observations differ from *contextual side information*, another generalization of the standard bandit problems where some additional information is given to the decision maker *before* pulling an arm. Asymptotically optimal policies have been provided for this setting [22] in the case of two-armed bandits.

3 SIDE OBSERVATIONS

Formally, a K-armed bandit problem is defined by K distributions $\mathcal{P}_1, \ldots, \mathcal{P}_K$, one for each "arm" of the bandit, with respective means $\mu_1, \ldots, \mu_K$. When the decision maker pulls arm i at time t, she receives a reward $X_{i,t} \sim \mathcal{P}_i$. All rewards $\{X_{i,t}, i \in [\![1, K]\!], t \geq 1\}$ are assumed to be independent. We will also assume that all $\{\mathcal{P}_i\}$ have support in $[0, 1]$. The mean estimate for $\mathbb{E}\left[X_{i,\cdot}\right]$ after m observations is $\overline{X}_{i,m} := \frac{1}{m}\sum_{s=1}^{m} X_{i,s}$. The (cumulative) regret after n steps is defined by

$$R(n) := \sum_{t=1}^{n} X_{i^*,t} - \sum_{t=1}^{n} X_{I_t,t},$$

[1] in which $\mathbb{E}\left[X_{\mathbf{x}}\right] = f(\mathbf{Z} \cdot \mathbf{x})$ for some regular function f

K	# of arms
$X_{i,t}$	reward of arm i at time t
μ_i	mean reward of arm i
Δ_i	expected loss for playing arm i
i^*	index of an optimal arm
μ^*	mean reward of arm i^*
I_t	index of the arm played at time t
$T_i(n)$	# pulls to arm i after n steps
$N(i)$	neighborhood of arm i (includes i)
$O_i(n)$	# observations for arm i after n steps
$O^*(n)$	same for arm i^*

Table 1: Notations Summary

where $i^* = \arg\max\{\mu_i\}$ and I_t is the index of the arm played at time t. The gambler's goal is to minimize the *expected regret* of the policy, which one can rewrite as

$$\mathbb{E}\left[R(n)\right] = \sum_{i=1}^{K} \Delta_i \mathbb{E}\left[T_i(n)\right]$$

where $T_i(n) := \sum_{t=1}^{n} \mathbf{1}\left\{I_t = i\right\}$ denotes the number of times arm i has been pulled up to time n, and $\Delta_i := \mu^* - \mu_i$ is the expected loss incurred by playing arm i instead of an optimal arm.

In the standard multi-armed bandit problem, the only information available at time t is the sequence $(X_{I_s,s})_{s\leq t}$. We now present our setting with side observations. The *side observation (SO) graph* $G = (V, E)$ is an undirected graph over the set of arms $V = [\![1, K]\!]$, where an edge $i \leftrightarrow j$ means that pulling arm i (resp. j) at time t yields a side observation of $X_{j,t}$ (resp. $X_{i,t}$). Let $N(i)$ denote the *observation set* of arm i consisting of i and its neighbors in G. Contrary to previous work on UCB algorithms [13, 6], in our setting the number of observations made so far for arm i at time n is not $T_i(n)$ but

$$O_i(n) := \sum_{t=1}^{n} \mathbf{1}\left\{I_t \in N(i)\right\},$$

which accounts for the fact that observations come from pulling either the arm or one of its neighbors. Note that $O_i(n) \geq T_i(n)$.

A *clique* in G is a subset of vertices $C \subset V$ such that all arms in C are neighbors with each other. A *clique covering* $\mathcal{C}$ of G is a set of cliques such that $\bigcup_{C\in\mathcal{C}} C = V$. Table 1 summarizes our notations.

3.1 LOWER BOUND

Before we analyze our policies, let us note that the problem we study is at least as difficult as the standard multi-armed bandit problem in the sense that, even with additional observations, the expected regret for any strategy has to grow at least logarithmically in the number of rounds. The only exception to this would be a graph where every node is neighbor with an optimal arm, a particular and easier setting that we do not study here. This observation is stated by the following Theorem:

Theorem 1. *Let* $B^* := \arg\max\{\mu_i \,|\, i \in V\}$ *and suppose* $\bigcup_{i\in B^*} N(i) \neq V$. *Then, for any uniformly good allocation rule,*[2] $\mathbb{E}\left[R(n)\right] = \Omega(\ln n)$.

Proof. For a set of arms S, we denote $N(S) := \cup_{j\in S} N(j)$. Let $i^* \in B^*$ and $v := \arg\max\{\mu_j \,|\, j \in V\backslash N(B^*)\}$, *i.e.*, the best arm which can not be observed by pulling an optimal arm.

First assume that $N(v) \cap N(B^*) = \emptyset$. The proof follows by comparing the initial bandit problem with side observations denoted $\mathcal{A}$ with the two-armed bandit $\mathcal{B}$ *without* side observations where the reward distributions are $\mathcal{P}^*$ for the optimal arm 1 and $\mathcal{P}_v$ for the non-optimal arm 2. To any strategy for $\mathcal{A}$, we associate the following strategy for $\mathcal{B}$: if arm i is played in $\mathcal{A}$ at time t, play in $\mathcal{B}$: arm 1 if $i \in N(B^*)$ and get reward $X_{i^*,t}$; arm 2 if $i \in N(v)$ and get reward $X_{v,t}$; no arm otherwise.

Let n' denote the number of arms pulled in $\mathcal{B}$ after n steps in $\mathcal{A}$. It is clear that $n' \leq n$ and a valid strategy for $\mathcal{A}$ gives a valid strategy for $\mathcal{B}$. The expected regret incurred by arm 1 in $\mathcal{B}$ is 0, and each time arm 2 is pulled in $\mathcal{B}$, a sub-optimal arm is pulled in $\mathcal{A}$ with larger expected loss. As a consequence, $\mathbb{E}\left[R_{\mathcal{A}}(n)\right] \geq \mathbb{E}\left[R_{\mathcal{B}}(n')\right]$, where $R_{\mathcal{A}}$ (resp. $R_{\mathcal{B}}$) denotes the regret in $\mathcal{A}$ (resp. $\mathcal{B}$). By the classical result of Lai and Robbins [13], $\mathbb{E}\left[R_{\mathcal{B}}(n')\right] = \Omega(\ln n')$. Hence, if $n' = \Omega(n)$ the claim follows. If $n' = o(n)$, then sub-optimal arms are played in $\mathcal{A}$ at least $n - n'$ times so that $\mathbb{E}\left[R_{\mathcal{A}}(n)\right] = \Omega(n - n') = \Omega(n)$ and the claim follows as well.

Now assume that $N(v) \cap N(B^*) \neq \emptyset$. A valid strategy for $\mathcal{A}$ does not give a valid strategy for $\mathcal{B}$ any more, since pulling an arm in $N(v) \cap N(B^*)$ gives information on both an optimal arm and v, *i.e.*, both arms in $\mathcal{B}$. We need to modify slightly the two-armed bandit as follows. First, we define $u := \arg\max\{\mu_i \,|\, i \in N(v) \cap N(B^*)\}$ and $w := v$ if $\mu_v \geq \mu_u$ and u otherwise. The reward distribution for arm 2 in $\mathcal{B}$ is now $\mathcal{P}_w$. To any strategy for $\mathcal{A}$, we associate a strategy for $\mathcal{B}$ as follows: when arm i is played in $\mathcal{A}$ at time t, play in $\mathcal{B}$:

- $i \in N(B^*)\backslash N(v) \Rightarrow$ pull arm 1, get reward $X_{i^*,t}$;
- $i \in N(v)\backslash N(B^*) \Rightarrow$ pull arm 2, get reward $X_{w,t}$;

[2] *i.e.*, not depending on the labels of the arms, see [13]

- $i \in N(v) \cap N(B^*) \Rightarrow$ pull arms 1 and 2 in two consecutive steps, getting rewards $X_{i^*,t}$ and $X_{w,t}$;
- otherwise, do not pull any arm.

Let n' denote the number of arms pulled in $\mathcal{B}$ after n steps in $\mathcal{A}$. We now see that any valid strategy for $\mathcal{A}$ gives a valid strategy for $\mathcal{B}$. As in previous setting, the expected regret incurred by arm 1 in $\mathcal{B}$ is 0, and each time arm 2 is pulled in $\mathcal{B}$, a sub-optimal arm is pulled in $\mathcal{A}$ with larger expected loss. As a consequence, $\mathbb{E}[R_{\mathcal{A}}(n)] \geq \mathbb{E}[R_{\mathcal{B}}(n')]$, and we can conclude as above. □

3.2 UPPER CONFIDENCE BOUNDS

The UCB1 policy constructs an Upper Confidence Bound for each arm i at time t by adding a *bias term* $\sqrt{2\ln t/T_i(t-1)}$ to its sample mean. Hence, the UCB for arm i at time t is

$$\mathsf{UCB}_i(t) := \overline{X}_{i,T_i(t-1)} + \sqrt{\frac{2\ln t}{T_i(t-1)}}.$$

Auer et al. [6] have proven that the policy which picks $\arg\max_i \mathsf{UCB}_i(t)$ at every step t achieves the following upper bound after n steps:

$$\mathbb{E}[R(n)] \leq 8\left(\sum_{i=1}^{K}\frac{1}{\Delta_i}\right)\ln n + \left(1+\frac{\pi^2}{3}\right)\sum_{i=1}^{K}\Delta_i. \quad (2)$$

In the setting with side observations, we will show in Section 3.3 that a generalization of this policy yields the (improved) upper bound

$$\mathbb{E}[R(n)] \leq 8\left(\inf_{\mathcal{C}}\sum_{C\in\mathcal{C}}\frac{\max_{i\in C}\Delta_i}{\min_{i\in C}\Delta_i^2}\right)\ln n + O(K),$$

where the $O(K)$ term is the same as in (2), and the infimum is over all possible clique coverings of the SO graph. We will detail in Section 3.3 how this bound improves on the original $\sum_i 1/\Delta_i$.

We will then introduce in Section 4 an algorithm improving on the constant $O(K)$ term (remember that the number of arms K is assumed $\gg 1$). By proactively using the underlying structure of the SO graph, we will reduce it to the following finite-time upper bound:

$$\left(1+\frac{\pi^2}{3}\right)\sum_{C\in\mathcal{C}}\Delta_C + o_{n\to\infty}(1),$$

where Δ_C is the *best* individual regret in clique $C \in \mathcal{C}$. Note that, while both constant terms were linear in K in Equation (2), our improved factors are both $O(|\mathcal{C}|)$ where $|\mathcal{C}|$ is the number of cliques used to cover the SO graph. We will show that this improvement is significant for dense reward structures, as is the case with advertising in social networks (see Section 5).

3.3 UCB-N POLICY

In the multi-armed bandit problem with side observations, when the decision maker pulls an arm i after t rounds of the game, he/she gets the reward $X_{i,t}$ and observes $\{X_{j,t} \,|\, j \in N(i)\}$. We consider in this section the policy UCB-N where one always plays the arm with maximum UCB, and updates all mean estimates $\{\overline{X}_{j,t} \,|\, j \in N(i)\}$ in the observation set of the pulled arm i.

Algorithm 1 UCB-N

$\overline{\mathbf{X}}, \mathbf{O} \leftarrow \mathbf{0}, \mathbf{0}$
for $t \geq 1$ **do**
  $i \leftarrow \arg\max_i \left\{\overline{X}_i + \sqrt{\frac{2\ln t}{O_i}}\right\}$
  pull arm i
  for $k \in N(i)$ **do**
    $O_k \leftarrow O_k + 1$
    $\overline{X}_k \leftarrow X_{k,t}/O_k + (1-1/O_k)\overline{X}_k$
  end for
end for

We take the convention $\sqrt{1/0} = +\infty$ so that all arms get observed at least once. This strategy takes all the side information into account to improve the learning rate. The following Theorem quantifies this improvement as a reduction in the logarithmic factor from Equation (2).

Theorem 2. *The expected regret of policy UCB-N after n steps is upper bounded by*

$$\mathbb{E}[R(n)] \;\leq\; \inf_{\mathcal{C}}\left\{8\left(\sum_{C\in\mathcal{C}}\frac{\max_{i\in C}\Delta_i}{\Delta_C^2}\right)\ln n\right\} + \left(1+\frac{\pi^2}{3}\right)\sum_{i=1}^{K}\Delta_i,$$

where $\Delta_C = \min_{i\in C}\Delta_i$.

Proof. Consider a clique covering $\mathcal{C}$ of $G = (V, E)$, *i.e.*, a set of subgraphs such that each $C \in \mathcal{C}$ is a clique and $V = \cup_{C\in\mathcal{C}} C$. One can define the *intra-clique regret* $R_C(n)$ for any $C \in \mathcal{C}$ by

$$R_C(n) \;:=\; \sum_{t<n}\sum_{i\in C}\Delta_i \mathbf{1}\{I_t = i\}.$$

Since the set of cliques covers the whole graph, we have $R(n) \leq \sum_{C\in\mathcal{C}} R_C(n)$. From now on, we will focus on upper bounding the intra-clique regret for a given clique $C \in \mathcal{C}$.

Let $T_C(t) := \sum_{i\in C} T_i(t)$ denote the number of times (any arm in) clique C has been played up to time t.

Then, for any positive integer ℓ_C,

$$R_C(n) \leq \ell_C \max_{i\in C} \Delta_i + \sum_{\substack{i\in C\\ t\leq n}} \Delta_i \mathbf{1}\{I_t = i; T_C(t-1) \geq \ell_C\}$$

Considering that the event $\{I_t = i\}$ implies $\{\overline{X}_{i,O_i(t-1)} + c_{t-1,O_i(t-1)} \geq \overline{X}^*_{O^*(t-1)} + c_{t-1,O^*(t-1)}\}$, we can upper bound this last summation by:

$$\sum_{\substack{i\in C\\ t<n}} \Delta_i \mathbf{1}\left\{ \begin{array}{l} \overline{X}_{i,O_i(t)} + c_{t,O_i(t)} \geq \overline{X}^*_{O^*(t)} + c_{t,O^*(t)} \\ T_C(t) \geq \ell_C \end{array} \right\}$$

$$\leq \sum_{\substack{i\in C\\ t<n}} \Delta_i \mathbf{1}\left\{ \max_{\ell_C \leq s_i \leq t} \overline{X}_{i,s_i} + c_{t,s_i} \geq \min_{0\leq s\leq t} \overline{X}^*_s + c_{t,s} \right\}$$

$$\leq \sum_{\substack{i\in C\\ t<n}} \sum_{s=0}^{t} \sum_{s_i=\ell_C}^{t} \Delta_i \mathbf{1}\left\{ \overline{X}_{i,s_i} + c_{t,s_i} \geq \overline{X}^*_s + c_{t,s} \right\}$$

Now, choosing

$$\ell_C \geq \max_{i\in C} \frac{8\ln n}{\Delta_i^2} = \frac{8\ln n}{\min_{i\in C}\Delta_i^2} = \frac{8\ln n}{\Delta_C^2}$$

will ensure that $\mathbb{P}\left(\overline{X}_{i,s_i} + c_{t,s_i} \geq \overline{X}^*_s + c_{t,s}\right) \leq 2t^{-4}$ for any $i \in C$ as a consequence of the Chernoff-Hoeffding bound, and following the same argument as in [6]. Hence, the overall clique regret is bounded by:

$$\begin{aligned} R_C(n) &\leq \ell_C \max_{i\in C}\Delta_i + \sum_{i\in C}\sum_{t=1}^{\infty} 2\Delta_i t^{-2} \\ &\leq 8\frac{\max_{i\in C}\Delta_i}{\Delta_C^2}\ln n + \left(1+\frac{\pi^2}{3}\right)\sum_{i\in C}\Delta_i. \end{aligned}$$

Summing over all cliques in $\mathcal{C}$ and taking the infimum over all possible coverings $\mathcal{C}$ yields the aforementioned upper bound. □

Remark. When $\mathcal{C}$ is the trivial covering $\{\{i\}, i \in V\}$, this upper bound reduces exactly to Equation (2). Therefore, taking side observations into account systematically improves on the baseline UCB1 policy.

4 UCB-MaxN POLICY

The second term in the upper bound from Theorem 2 is still linear in the number of arms and may be large when $K \gg 1$. In this section, we introduce a new policy that makes further use of the underlying reward observations to improve performances.

Consider the two extreme scenarii that can make an arm i played at time t: it has the highest UCB, so

- either its average estimate $\overline{X}_{i,t}$ is very high, which means it is empirically the best arm to play,
- or its bias term $\sqrt{2\ln t/O_i(t-1)}$ is very high, which means one wants more information on it.

In the second case, one wants to observe a sample $X_{i,t}$ to reduce the uncertainty on arm i. But in the side observation setting, we don't have to pull this arm directly to get an observation: we may as well pull any of its neighbors, especially one with higher empirical rewards, and reduce the bias term all the same. Meanwhile, in the first case, arm i will already be the best empirical arm in its observation set.

This reasoning motivates the following policy, called UCB-MaxN, where we first pick the arm we want to *observe* according to UCBs, and then pick in its observation set the arm we want to *pull*, this time according to its empirical mean only.

Algorithm 2 UCB-MaxN

$\overline{\mathbf{X}}, \mathbf{n} \leftarrow \mathbf{0}, \mathbf{0}$
for $t \geq 1$ **do**
  $i \leftarrow \arg\max_i \left\{\overline{X}_i + \sqrt{\frac{2\ln t}{O_i}}\right\}$
  $j \leftarrow \arg\max_{j\in N(i)} \overline{X}_j$
  pull arm j
  for $k \in N(j)$ **do**
    $O_k \leftarrow O_k + 1$
    $\overline{X}_k \leftarrow X_{k,t}/O_k + (1-1/O_k)\overline{X}_k$
  end for
end for

Asymptotically, UCB-MaxN reduces the second factor in the regret upper bound (2) from $O(K)$ to $O(|\mathcal{C}|)$, where $\mathcal{C}$ is an optimal clique covering of the side observation graph G.

Theorem 3. *The expected regret of strategy UCB-MaxN after n steps is upper bounded by*

$$\begin{aligned} \mathbb{E}[R(n)] \leq \inf_{\mathcal{C}} \Bigg\{ & 8\left(\sum_{C\in\mathcal{C}} \frac{\max_{i\in C}\Delta_i}{\Delta_C^2}\right)\ln n \\ & + \left(1+\frac{\pi^2}{3}\right)\sum_{C\in\mathcal{C}}\Delta_C \Bigg\} \\ & + o_{n\to\infty}(1) \end{aligned}$$

We will make use of the following lemma to prove this theorem:

Lemma 1. *Let $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$ denote two sets of i.i.d. random variables of respective means μ and ν such that $\mu < \nu$. Let $\Delta := \mu - \nu$. Then,*

$$\mathbb{P}\left(\overline{X}_n > \overline{Y}_m\right) \leq 2e^{-\min(n,m)\Delta^2/2}.$$

Proof. Note that either $\overline{X}_n < \frac{1}{2}(\mu+\nu) < \overline{Y}_m$ or one of the two events $\overline{X}_n > \frac{1}{2}(\mu+\nu)$ or $\overline{Y}_m < \frac{1}{2}(\mu+\nu)$ occurs. As a consequence, the probability $\mathbb{P}\left(\overline{X}_n > \overline{Y}_m\right)$ is lower than

$$\begin{aligned}
&\mathbb{P}\left(\overline{X}_n > \frac{\mu+\nu}{2}\right) + \mathbb{P}\left(\overline{Y}_m < \frac{\mu+\nu}{2}\right) \\
&\leq \mathbb{P}\left(\overline{X}_n - \mu > -\frac{\Delta}{2}\right) + \mathbb{P}\left(\overline{Y}_m - \nu < \frac{\Delta}{2}\right) \\
&\leq e^{-n\Delta^2/2} + e^{-m\Delta^2/2} \\
&\leq 2e^{-\min(n,m)\Delta^2/2}.
\end{aligned}$$

□

Proof of Theorem 3. Let $k_C := \arg\min_{i\in C} \Delta_i$ denote the best arm in clique C, and define $\delta_i := \Delta_i - \Delta_C$ for each arm $i \in C$. As in the beginning of our proof for Theorem 2, we can upper bound:

$$R_C(n) \leq \ell_C \max_{i\in C} \Delta_i + \sum_{\substack{i\in C \\ t<n}} \Delta_i \mathbf{1}\left\{I_t = i; T_C(t-1) \geq \ell_C\right\} \tag{3}$$

where this last summation is upper bounded by

$$\begin{aligned}
&\sum_{\substack{i\in C \\ t<n}} (\Delta_C + \delta_i)\mathbf{1}\left\{I_t = i; T_C(t-1) \geq \ell_C\right\} \\
&\leq \sum_{t<n} \Delta_C \mathbf{1}\left\{I_t = k_C; T_C(t-1) \geq \ell_C\right\} + \\
&\quad \sum_{\substack{i\in C \\ t<n}} \Delta_i \mathbf{1}\left\{I_t = i; T_C(t-1) \geq \ell_C\right\}
\end{aligned}$$

The first summation can be bounded using the Chernoff-Hoeffding inequality as before:

$$\begin{aligned}
&\sum_{t<n} \mathbf{1}\left\{I_t = k_C; T_C(t-1) \geq \ell_C\right\} \\
&\leq \sum_{t<n} \sum_{\substack{s\leq t \\ \ell_C \leq s_k \leq t}} \mathbf{1}\left\{\overline{X}_{k_C,s_k} + c_{t,s_k} > \overline{X}^*_s + c_{t,s}\right\} \\
&\leq 2\sum_{t<n} t^{-2} \leq 1 + \frac{\pi^2}{3}
\end{aligned}$$

with an appropriate choice of $\ell_C \geq \frac{8\ln n}{\Delta_C^2}$. As to the second summation, the fact that Algorithm 2 picks i instead of k_C at step t implies that $\overline{X}_{i,O_i(t)} > \overline{X}_{k_C,O_{k_C}(t)}$, so

$$\begin{aligned}
R'_C(n) &:= \sum_{\substack{i\in C \\ t<n}} \Delta_i \mathbf{1}\left\{I_t = i; T_C(t-1) \geq \ell_C\right\} \\
&\leq \sum_{\substack{i\in C \\ t<n}} \Delta_i \mathbf{1}\left\{ \begin{array}{l} \overline{X}_{i,O_i(t-1)} > \overline{X}_{k_C,O_{k_C}(t-1)} \\ T_C(t-1) \geq \ell_C \end{array} \right\}
\end{aligned}$$

Consider the times $\ell_C \leq \tau_1 \leq \cdots \leq \tau_{T_C(n)}$ when the clique C was played (after the first ℓ_C steps). Then, one can rewrite $R'_C(n)$ as follows:

$$\begin{aligned}
R'_C(n) &\leq \sum_{u=\ell_C}^{T_C(n)} \sum_{i\in C} \Delta_i \mathbf{1}\left\{\overline{X}_{i,O_i(\tau_u)} > \overline{X}_{k_C,O_{k_C}(\tau_u)}\right\} \\
\mathbb{E}\left[R'_C(n)\right] &\leq \sum_{u=\ell_C}^{T_C(n)} \sum_{i\in C} \Delta_i \mathbb{P}\left(\overline{X}_{i,O_i(\tau_u)} > \overline{X}_{k_C,O_{k_C}(\tau_u)}\right)
\end{aligned}$$

After the clique C has been played u times, all arms in C being neighbors in the side observation graph, we know that each estimate $\overline{X}_i, i \in C$ has at least u samples, *i.e.*, $O_i(\tau_u) \geq u$. Therefore, using Lemma 1 with "$n = O_i(\tau_u)$" and "$m = O_{k_C}(\tau_u)$" in the previous expression yields

$$\begin{aligned}
\mathbb{E}\left[R'_C(n)\right] &\leq \sum_{i\in C} \sum_{u=\ell_C}^{T_C(n)} 2\Delta_i e^{-u\delta_i^2/2} \\
&\leq 2\sum_{\substack{i\in C \\ \delta_i>0}} \Delta_i \frac{1-e^{-n\delta_i^2/2}}{1-e^{-\delta_i^2/2}} e^{-\ell_C\delta_i^2/2},
\end{aligned}$$

where $\delta_i = \mu_i - \min_{j\in C}\mu_j$. Combining all these separate upper bounds in Equation (3) leads us to

$$\begin{aligned}
\mathbb{E}\left[R_C(n)\right] \leq 8\frac{\max_{i\in C}\Delta_i}{\Delta_C^2}\ln n + \left(1+\frac{\pi^2}{3}\right)\Delta_C \\
+ 2\sum_{\substack{i\in C \\ \delta_i>0}} \Delta_i \frac{1-e^{-n\delta_i^2/2}}{1-e^{-\delta_i^2/2}} \cdot \left(\frac{1}{n}\right)^{4\delta_i^2/\Delta_C^2}
\end{aligned}$$

where this last term is $o_{n\to\infty}(1)$. □

Remark. UCB-MaxN is asymptotically better than UCB-N: again, its upper bound expression boils down to Equation (2) when applied to the trivial covering $\mathcal{C} = \{\{i\}, i \in V\}$.

Note that our bound is achieved *uniformly over time* and not only asymptotically; we only used the $o(1)$ notation in Theorem 3 to highlight that the last term vanishes when $n \to \infty$. This term may actually be large for small values of n and pathological regret distributions, *e.g.*, if some δ_i are such that $\delta_i \ll \Delta_C$. However, with distributions drawn from real datasets we observed a fast decrease: in the Flixster experiment for instance (see Section 5.3), this term was below the $(1+\pi^2/3)\Delta_C$ constant for more than 80% of the cliques after $T \sim 20K$ steps.

5 EXPERIMENTS

We have seen so far that our policies improve regret bounds compared to standard UCB strategies. Let us

evaluate how these algorithms perform on real social network datasets. In this section, we perform three experiments. First, we evaluate the UCB-N and UCB-MaxN policies on a movie recommendation problem using a dataset from Flixster [2]. The policies are compared to three baseline solutions: two UCB variants with no side observations, and an ε-greedy with side observations. Second, we investigate the impact of extending side observations to friends-of-friends, a setting inspired from average user preferences on social networks that densifies the reward structure and speeds up learning. Finally, we apply the UCB-N and UCB-MaxN algorithms in a bigger social network setup with a dataset from Facebook [1].

5.1 DATASETS

We perform empirical evaluation of our algorithms on datasets from two social networks: Flixster and Facebook. Flixster is a social networking service in which users can rate movies. This social network was crawled by Jamali *et al.* [10], yielding a dataset with 1M users, 14M friendship relations, and 8.2M movie ratings that range from 0.5 to 5 stars. We clustered the graph using Graclus [17] and obtained a strongly connected subgraph. Furthermore, we eliminated users that rated less than 30 movies and movies rated by less than 30 users. This preprocessing step helps us to learn more stable movie-rating profiles (Section 5.2). The resulting dataset involves 5K users, 5K movies, and 1.7M ratings. The subgraph from Facebook we used was collected by Viswanath *et al.* [21] from the New Orleans region. It contains 60K users and 1.5M friendship relationships. Again, we clustered the graph using Graclus and obtained a strongly connected subgraph of 14K users and 500K edges.

5.2 EXPERIMENTAL SETUP

We evaluate our policies in the context of movie recommendation in social networks. The problem is set up as a repetitive game. At each turn, a new movie is sampled from a homogeneous movie database and the policy offers it at a promotional price to one user in the social network.[3] If the user rates the movie higher than 3.5 stars, we assume that he/she accepts the promotion and our reward is 1, otherwise the reward is 0. The promotion is then posted on the user's wall and we assume that all friends of that user express their opinion, *i.e.*, whether they would accept a similar offer (*e.g.*, on Facebook by "Liking" or not the promotional message). The goal is to learn a policy that gives promotions to people who are likely to accept them.

We use standard matrix factorization techniques [12] to predict users ratings from the Flixster dataset. Since the Facebook dataset does not contain movie ratings, we generated rating profiles by matching users between the Flixster and Facebook social networks. This matching is based on structural features only, such as vertex degree, the aim of this experiment being to evaluate the performances of our policies in a bigger network with similar rating distributions across vertices.

The upper bounds we derived in the analysis of UCB-N and UCB-MaxN (Theorems 2 and 3) involve the number of cliques used to cover the side observation graph; meanwhile, bigger cliques imply more observations per step, and thus a faster convergence of estimators. These observations suggest that the minimum number of cliques required to cover the graph impacts the performances of our allocation schemes, which is why we took this factor into account in our evaluation.

Unfortunately, finding a cover with the minimum number of cliques is an NP-hard problem. We addressed it suboptimally as follows. First, for each vertex i in the graph, we computed a maximal clique C_i involving i. Second, a covering using $\{C_i\}$ is found using a greedy algorithm for the SET COVER problem [11].

For each experiment, we evaluate our policies on 3 subgraphs of the social network obtained by terminating the greedy algorithm after 3%, 15%, and 100% of the graph have been covered. This choice is motivated by the following observation: the degree distribution in social networks is heavy-tailed, and the number of cliques needed to cover the whole graph tends to be on the same scale of order as the number of vertices; meanwhile, the most active regions of the network (which are of practical interest in our content recommendation scenario) are densest and thus easier to cover with cliques. Since the greedy algorithm follows a biggest-cliques-first heuristic, looking at these 3% and 15% covers allows us to focus on these densest regions.

The quality of all policies is evaluated by the *per-step regret* $r(n) := \frac{1}{n}\mathbb{E}[R(n)]$. We also computed for each plot the improvement of each policy against UCB1 after the last round T (a $k\times$ improvement means that $r(T) \approx r_{\mathsf{UCB1}}(T)/k$). This number can be viewed as a speedup in the convergence to the optimal arm. Finally, all plots include a vertical line indicating the number of cliques in the cover, which is also the number of steps needed by any policy to pull every arm at least once. Before that line, all policies perform approximately the same.

[3]In accordance with the bandit framework, we further assume that the same movie is never sampled twice.

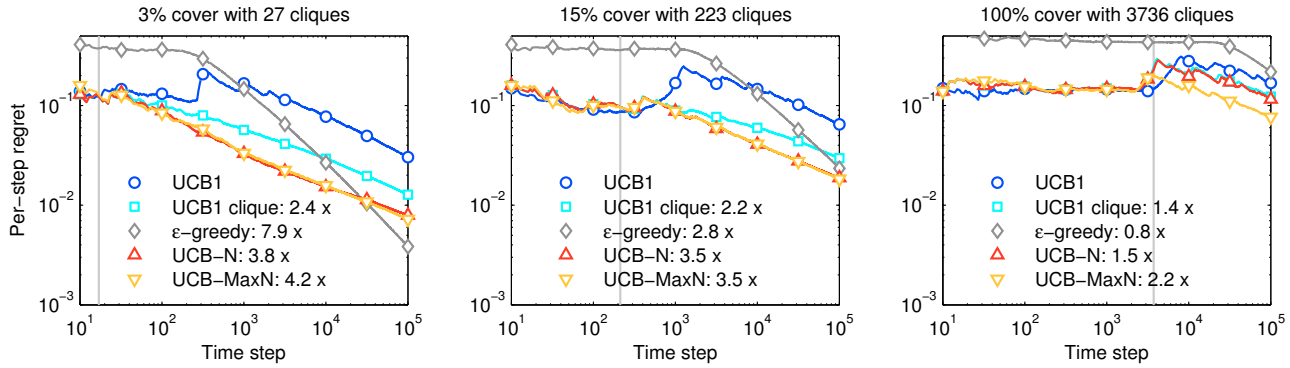


Figure 1: Per-step regret of four bandit policies on the Flixster graph.

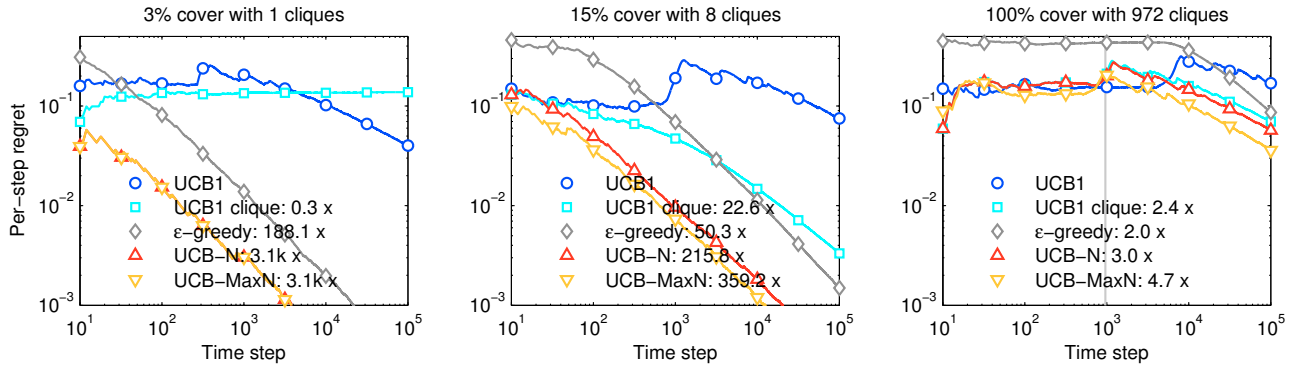


Figure 2: Per-step regret of four bandit policies on the Flixster graph with friend-of-friend side observations.

5.3 FLIXSTER

In this first experiment, we evaluate UCB-N and UCB-MaxN in the Flixster social network. These policies are compared to three baselines: UCB1 with no side observation, UCB1-on-cliques and ε-greedy. Our ε-greedy is the same as ε_n-greedy in [6] with $c = 5$, $d = 1$ and $K = |\mathcal{C}|$, which turned out to be the best empirical parametrization within our experiments. UCB1-on-cliques is similar to UCB-N, except that it updates the estimators $\{\overline{X}_k \,|\, k \in N(i)\}$ with the reward $X_{i,t}$ of the pulled arm i. This is a simple approach to make use of the network structure without access to side observations. As illustrated in Figure 1, we observe the following trends.

The regret of UCB-N and UCB-MaxN is significantly smaller than the regret of UCB1 and UCB1-on-cliques, which suggests these strategies successfully benefit from side observations to improve their learning rate. ε-greedy shows improvement as well, but its performances decrease rapidly as the size of the cover grows (*i.e.*, adding smaller cliques) compared to our strategies. Overall, the performance of all policies deteriorates with more coverage, which is consistent with the $O(K)$ and $O(|\mathcal{C}|)$ upper bounds on their regrets.

UCB-MaxN does not perform significantly better than UCB-N when the size of the cover $|\mathcal{C}|$ is small. This can be explained based on the amount of overlap between the cliques in the cover. In practice, we observed that UCB-MaxN performs better when individual arms belong to many cliques on average. For our 3%, 15%, and 100% graph cover simulations, the average number of cliques covering an arm were 1.18, 1.09, 1.76; meanwhile, the regrets of UCB-MaxN were 9%, 3%, and 33% smaller than the regrets of UCB-N, respectively.

5.3.1 FRIENDS OF FRIENDS

In the second experiment we use a denser graph where side observations come from friends and friends of friends. This setting is motivated by the observation that a majority of social network users do not restrict content sharing to their friends. For instance, more than 50% of Facebook users share all their content items with friends of friends [15].

Figure 2 shows that the gap between the baselines and our policies is even wider in this new setting. This phenomenon can be explained by larger cliques; for instance, only 8 cliques are needed to cover 15% of the graph in this instance, which is 20 times less than in Section 5.3.

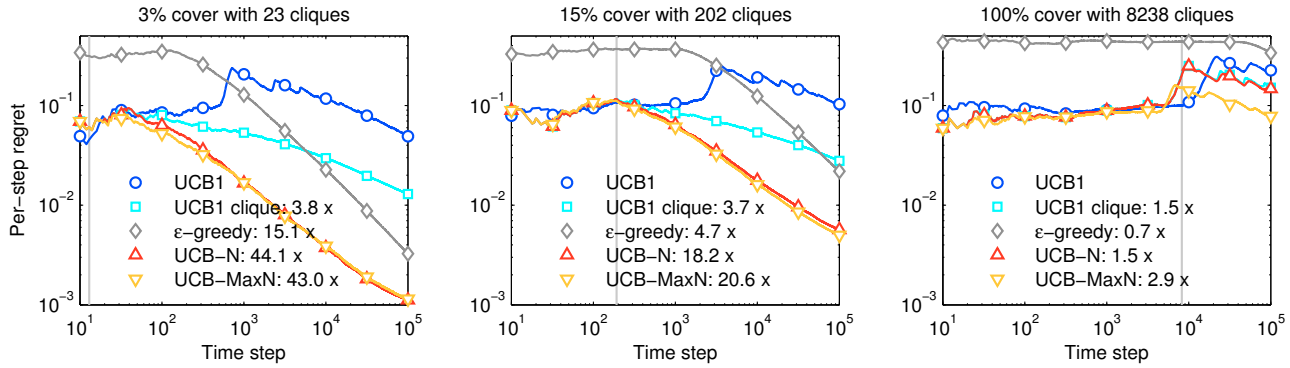


Figure 3: Per-step regret of four bandit policies on the Facebook graph.

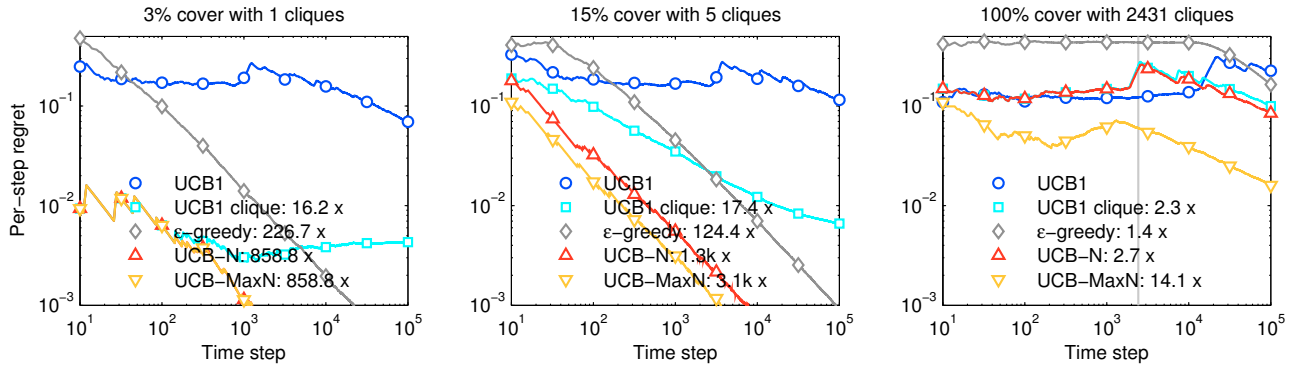


Figure 4: Per-step regret of four bandit policies on the Facebook graph with friend-of-friend side observations.

5.4 FACEBOOK

In the next experiment, we evaluate UCB-N and UCB-MaxN on a subset of the Facebook social network. This graph has three times as many vertices and twice as many edges as the Flixster graph. We experiment with both friends and friends-of-friends side observations.

As shown in Figures 3 and 4, compared to Figures 1 and 2, we observe much smaller regrets in this setting, essentially because the Facebook graph is denser. For instance, only 5 friend-of-friend cliques are needed to cover 15% of the graph. For this cover, the regret of UCB-MaxN is 10 times smaller than the regret of UCB1-on-cliques and UCB-N, respectively.

6 CONCLUSION AND FUTURE WORK

In this paper, we considered the stochastic multi-armed bandit problem with side observations. This problem generalizes the standard, independent multi-armed bandit, and has a broad set of applications including Internet advertisement and content recommendation systems. Our contribution consists in two new strategies, UCB-N and UCB-MaxN, that leverage this additional information into substantial learning rate speed-ups.

We showed that our policies reduce regret bounds from $O(K)$ to $O(|\mathcal{C}|)$, which is a significant improvement for dense reward-dependency structures. We also evaluated their performances on real datasets in the context of movie recommendation in social networks. Our experiments suggest that these strategies significantly improve the learning rate when the side observation graph is a dense social network.

So far we have focused on *cliques* as a convenient way to analyze our policies, but none of our two strategies explicitly relies on cliques (they only use the notion of neighborhood). Characterizing the most appropriate subgraph structure for this problem is still an open question that could lead to better regret bounds and inspire more efficient policies.

References

[1] Facebook. `www.facebook.com`.

[2] Flixster. `www.flixster.com`.

[3] Jean-Yves Audibert, Remi Munos, and Csaba Szepesvari. Tuning bandit algorithms in stochas-

tic environments. *Algorithmic Learning Theory*, pages 150–165, 2007.

[4] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422, March 2003.

[5] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

[6] Peter Auer, Nicol Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

[7] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

[8] Varsha Dani, Thomas P. Hayes, and Sham M. Kakade. Stochastic linear optimization under bandit feedback. In *Computational Learning Theory*, pages 355–366, 2008.

[9] Sarah Filippi and Olivier Cappé. Parametric bandits : The generalized linear case. *Advances in Neural Information Processing Systems 23*, pages 1–9, 2010.

[10] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, 2010.

[11] David Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[12] Raghunandan Keshavan and Sewoong Oh. A gradient descent algorithm on the grassman manifold for matrix completion. *arXiv:0910.5260v2*, 2009.

[13] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Adv. in Appl. Math.*, 6(1):4–22, 1985.

[14] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670, 2010.

[15] Yabing Liu, Krishna P. Gummadi, Balachander Krishnamurthy, and Alan Mislove. Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pages 61–70, New York, NY, USA, 2011. ACM.

[16] Shie Mannor and Ohad Shamir. From bandits to experts: On the value of side-observations. In *NIPS*, pages 684–692, 2011.

[17] University of Texas. Graclus. `www.cs.utexas.edu/users/dml/Software/graclus.html`.

[18] Sandeep Pandey, Deepayan Chakrabarti, and Deepak Agarwal. Multi-armed bandit problems with dependent arms. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 721–728, New York, NY, USA, 2007. ACM.

[19] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[20] Paat Rusmevichientong and John N. Tsitsiklis. Linearly parameterized bandits. *Math. Oper. Res.*, 35(2):395–411, May 2010.

[21] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *WOSN '09: Proceedings of the 2nd ACM Workshop on Online Social Networks*, 2009.

[22] Chih-Chun Wang, Sanjeev R. Kulkarni, and H. Vincent Poor. Bandit problems with side observations. *IEEE Trans. Automat. Control*, 50(3):338–355, 2005.

Interdependent Defense Games: Modeling Interdependent Security under Deliberate Attacks

Hau Chan
Department of Computer Science
Stony Brook University
hauchan@cs.stonybrook.edu

Michael Ceyko
Department of Computer Science
Harvard University
ceyko@fas.harvard.edu

Luis E. Ortiz
Department of Computer Science
Stony Brook University
leortiz@cs.stonybrook.edu

Abstract

We propose *interdependent defense (IDD) games*, a computational game-theoretic framework to study aspects of the interdependence of risk and security in multi-agent systems under deliberate external attacks. Our model builds upon *interdependent security (IDS) games*, a model due to Heal and Kunreuther that considers the source of the risk to be the result of *a fixed randomized-strategy*. We adapt IDS games to model *the attacker's deliberate behavior*. We define the attacker's pure-strategy space and utility function and derive appropriate cost functions for the defenders. We provide a complete characterization of mixed-strategy Nash equilibria (MSNE), and design a simple *polynomial-time* algorithm for computing *all* of them, for an important subclass of IDD games. In addition, we propose a random-instance generator of (general) IDD games based on a version of the real-world Internet-derived Autonomous Systems (AS) graph (with around $27K$ nodes and $100K$ edges), and present promising empirical results using a simple learning heuristics to compute (approximate) MSNE in such games.

1 INTRODUCTION

Attacks carried out by hackers and terrorists over the last few years have led to increased efforts by both government and the private sector to create and adopt mechanisms to prevent future attacks. This effort has yielded a more focused research attention to models, computational and otherwise, that facilitate and help to improve (homeland) security for both physical infrastructure and cyberspace. In particular, there has been quite a bit of recent research activity in the general area of game-theoretic models for terrorism settings (see, e.g., Bier and Azaiez [2009] and Cárceles-Poveda and Tauman [2011]).

Interdependent security (IDS) games are one of the earliest models resulting from a game-theoretic approach to model security in non-cooperative environments composed of free-will self-interested individual decision-makers. Originally introduced and studied by economists Kunreuther and Heal [2003], IDS games model general abstract security problems in which an individual within a population considers whether to voluntarily invest in some protection mechanisms or security against a risk they may face, knowing that the cost-effectiveness of the decision depends on the investment decisions of others in the population because of transfer risks (i.e., the "bad event" may be transferable from a compromised individual to another).

In their work, Kunreuther and Heal [2003] provided several examples based on their economics, finance and risk management expertise. (We refer the reader to their paper for more detailed descriptions.) As a canonical example of the real-world relevance of IDS settings and the applicability of IDS games, Heal and Kunreuther [2005] used this model to describe problems such as airline baggage security. In their setting, individual airlines may choose to invest in additional complementary equipment to screen passengers' bags and check for hazards such as bombs that could cause damage to their passengers, planes, buildings, or even reputations. However, mainly due to the large amount of traffic volume, it is impractical for an airline to go beyond applying security checks to bags incoming from passengers and include checks to baggage or cargo transferred from other airlines. On the other hand, if an airline invests in security, they can still experience a bad event if the bag was transferred from an airline that does not screen incoming bags, rendering their investment useless. [1] Thus, we can see how

[1]Note that even if full screening were performed, the Christmas Day 2009 episode in Detroit [O'Connor and

the cost-effectiveness of an investment can be highly dependent on others' investment decisions. Another recent application of the IDS model is on container shipping transportation [Gkonis and Psaraftis, 2010]. They use the IDS model to study the effect of investment decision on container screening of ports have on their neighboring ports.

In this work, we build on the literature in IDS games. In particular, we adapt the model to situations in which the abstract "bad event" results from the *deliberate* action of an attacker. The "internal agents" (e.g., airlines and computer network users), whom we also often refer to as "defenders" or "sites," have the voluntary choice to individually invest in security to defend themselves against a direct or indirect offensive attack, modulo, of course, the cost-effectiveness to do so. A side benefit of explicitly modeling the attacker, as we do in our model, is that the probability of an attack results directly from the equilibrium analysis. Building IDS games can be hard because it requires *a priori* knowledge of the likelihood of an attack. Attacks of this kind are considered rare events and thus notoriously difficult to statistically estimate in general.

Related Work. Johnson et al. [2010] and Fultz and Grossklags [2009] independently developed non-cooperative game models similar to ours. Johnson et al. [2010] extend IDS games by modeling uncertainty about the source of the risk (i.e., the attacker) using a Bayesian game over risk parameters. Fultz and Grossklags [2009] propose and study a non-graphical game-theoretic model for the interactions between attackers and nodes in a network. In their model, each node in the network can decide on whether to contribute (by investment) to the overall safety of the network and/or to individual safety. The attackers can attack any number of nodes, but with each attack there is an increased probability that the attacker might get caught and suffer penalties or fines. Hence, while their game has IDS characteristics, it is technically not within the standard IDS game framework introduced by Heal and Kunreuther.

Most of the previous related work explore the realm of information security and are application/network specific (see Roy et al. [2010] for a survey on game theory application to network security). Past literature has largely focused on two-person (an attacker and a defender) games where the nodes in the network are regarded as a single entity (or a central defender). For example, Lye and Wing [2002] look at the interactions between an attacker and the (system) administrator using a two-player stochastic game. Recent work uses a Stackelberg game model in which the defender (or leader) commits to a mixed strategy to allocate resources to defend a set of nodes in the network, and the follower (or attacker) optimally allocates resources to attack a set of "targets" in the network given the leader's commitment [Jain et al., 2011, Kiekintveld et al., 2009, Korzhyk et al., 2010, 2011a,b].

Other recent work strive to understand the motivation of the attackers. For example, Liu [2003] focus on understanding the attacker's intent, objectives, and strategies and derive a (two-player) game-theoretical model based on these, while Cremonini and Nizovtsev [2006] use cost-benefit analysis (of attackers) to address the issue of the optimal amount of security (of the nodes in the network).

Our Contribution. We adapt the standard non-cooperative framework of IDS games to settings in which the source of the risk is the result of a deliberate, strategic decision by an external attacker. In particular, we design and propose *interdependent defense (IDD) games*, a new class of games that, in contrast to standard IDS games, model the attacker *explicitly*, while maintaining a core component of IDS systems: the potential *transferability* of the risk resulting from an attack. We note that the explicit modeling of risk transfer is an aspect of our model that has not been a focus of previous game-theoretic attacker-defender models of security discussed earlier.

We formally define and study IDD games in depth in Section 3. We present several results that *fully* characterize their NE. [2] We also provide a *polynomial-time* algorithm to compute *all* MSNE for the important subclass of IDD games in which there is only one attack, the defender nodes are *fully transfer-vulnerable* (i.e., investing in security does nothing to reduce their external/transfer risk) and transfers are *one-hop*. [3]

We note that considering a single attacker is a typical assumption in security settings (see previous work discussed earlier). It is also reasonable: We can view many attackers as a single attacker. Allowing at most one attack prevents immediate representational and computational intractability problems because the number of the attacker's (pure) strategies grows exponentially with the number of attacks. Finally, because the attacker has no fixed target, it is ineffective for the attacker to consider or go beyond plans of attacks involving multiple (> 2) transfers: such plans are complex, time consuming and costly.

Our computational results are significant and surprising because computing all NE in general IDS games is

Schmitt, 2009] serves as a reminder that transfer risk still exists.

[2] Due to space limitation, we omit proofs of our main technical results and instead refer the reader to the supplementary document for details [Chan et al., 2012].

[3] We note that the original IDS games were also *fully transfer-vulnerable* and assume one-hop transfers.

hard (i.e., the Nash-extension problem in general IDS games is NP-complete [Kearns and Ortiz, 2003]). [4] We do not know of any other *non-trivial* game for which there exists a *polynomial-time* algorithm to compute *all* NE except ours and the algorithm for uniform-transfer IDS games of Kearns and Ortiz [2003].

In Section 4, we provide experimental results from applying learning-in-games heuristics to compute approximate NE to both fixed and randomly-generated instances of IDD games, with at most one simultaneous attack and one-hop transfers, on a very large Internet AS graph ($\approx 27K$ nodes and $\approx 100K$ edges).

2 IDS GAMES

Each player i in an IDS game has a choice to invest ($a_i = 1$) or not invest ($a_i = 0$). For each player i, C_i and L_i are the cost of investment and loss induce by the bad event, respectively. We define the ratio of the two parameters, the player's "cost-to-loss" ratio, as $\rho_i \equiv C_i/L_i$. Bad events can occur through both *direct* and *indirect* means. *Direct risk*, or *internal risk,* p_i is the probability that player i will experience a bad event because of direct contamination. The standard IDS model assumes that investing will completely protect the player from direct contamination; hence, internal risk is only possible when $a_i = 0$. *Indirect risk* q_{ji} is the probability that player j is directly "contaminated," does not experience the bad event but transfers it to player i who ends up experiencing the bad event. There is an implicit global constraint on these parameters, by the axioms of probability: $p_i + \sum_{j=1}^{n} q_{ij} \leq 1$ for all i.

We now formally define a (directed) graphical games [Kearns, 2007, Kearns et al., 2001] version of IDS games, as first introduced by Kearns and Ortiz [2003]. Denote by $[n] \equiv \{1, \ldots, n\}$ the set of n players. Note that the parameters q_{ij}'s induce a directed graph $G = ([n], E)$ such that $E \equiv \{(i,j) | q_{ij} > 0\}$. Let $\mathrm{Pa}(i) \equiv \{j \mid q_{ji} > 0\}$ be the set of players that are *parents* of player i in G (i.e., the set of players that player i is exposed to via transfers), and by $\mathrm{PF}(i) \equiv \mathrm{Pa}(i) \cup \{i\}$ the *parent family* of player i, which includes i. Denote by $k_i \equiv |\mathrm{PF}(i)|$ the size of the parent family of player i. Similarly, let $\mathrm{Ch}(i) \equiv \{j \mid q_{ij} > 0\}$ be the set of players that are *children* of player i (i.e., the set of players to whom player i can present a risk via transfer) and $\mathrm{CF}(i) \equiv \mathrm{Ch}(i) \cup \{i\}$ the *(children) family* of player i, which includes i. The *probability that player i is safe from player j*, as a function of player j's decision, is

$$e_{ij}(a_j) \equiv a_j + (1 - a_j)(1 - q_{ji}) = (1 - q_{ji})^{1-a_j},$$

because if j invests, then it is impossible for j to transfer the bad event, while if j does not invest, then j either experiences the bad event or transfers it to another player, but never both.

Denote by $\mathbf{a} \equiv (a_1, \ldots, a_n) \in \{0,1\}^n$ the *joint action* of all n players. Also denote by $\mathbf{a}_{-i}$ the joint-action of all players except i and for any subset $I \subset [n]$ of players, denote by $\mathbf{a}_I$ the sub-component of the joint action corresponding to those players in I only. We define i's *overall safety* from *all* other players as $s_i(\mathbf{a}_{\mathrm{Pa}(i)}) \equiv \prod_{j \in \mathrm{Pa}(i)} e_{ij}(a_j)$ and equivalently his *overall risk* from *some* other players is $r_i(\mathbf{a}_{\mathrm{Pa}(i)}) \equiv 1 - s_i(\mathbf{a}_{\mathrm{Pa}(i)})$. Note that each players' external safety (and risk) is a direct function of its parents only, *not* all other players. From these definitions, we obtain player i's overall cost, the cost of joint action $\mathbf{a} \in \{0,1\}^n$, corresponding to the (binary) investment decision of all players, is

$$\begin{aligned} M_i(a_i, \mathbf{a}_{\mathrm{Pa}(i)}) \equiv & a_i[C_i + r_i(\mathbf{a}_{\mathrm{Pa}(i)})L_i] + \\ & (1 - a_i)[p_i + (1 - p_i)r_i(\mathbf{a}_{\mathrm{Pa}(i)})]L_i \ . \end{aligned}$$

3 IDD GAMES

In the standard IDS game model, investment in security does not reduce transfer risks. However, in some IDS settings (e.g., vaccination and cyber-security), it is reasonable to expect that security investments would include mechanisms to reduce transfer risks. This motivates our first modification to the traditional IDS games: allowing the investment in protection to not only makes us safe from direct attack but also partially reduce (or even eliminate) the transfer risk. We incorporate this factor by introducing a new real-valued parameter $\alpha_i \in [0,1]$ representing the probability that a transfer of a potentially bad event will go *unblocked* by i's security, assuming i has invested. Thus, we redefine player *i's* overall cost as [5]

$$\begin{aligned} M_i(a_i, \mathbf{a}_{\mathrm{Pa}(i)}) \equiv & a_i[C_i + \alpha_i r_i(\mathbf{a}_{\mathrm{Pa}(i)})L_i] + \\ & (1 - a_i)[p_i + (1 - p_i)r_i(\mathbf{a}_{\mathrm{Pa}(i)})]L_i \ . \end{aligned}$$

Next, we introduce an additional player, the *attacker*, who *deliberately* initiates bad events. (So that now bad events are no longer "chance occurrences" without any strategic deliberation.) The attacker has a *target decision* for each player - a choice of attack ($b_i = 1$) or not attack ($b_i = 0$) player i. Hence,

[4] To put our computational contributions in context, note that deciding whether a game has a PSNE is in general NP-complete (see, e.g., Gilboa and Zemel [1989], Gottlob et al. [2005]), and computing an MSNE is PPAD-complete, even in two-player games (see, e.g., Chen et al. [2009] and Daskalakis et al. [2009]). Also, computing *all* MSNE is rarely achieved and counting-related problems are often #P-complete (see, e.g., [Conitzer and Sandholm, 2008]).

[5] A similar extension was also proposed independently by Heal and Kunreuther [2007].

the attacker's pure strategy is denoted by the vector $\mathbf{b} \in \{0,1\}^n$. The game parameter p_i implicitly "encodes" b_i because $b_i = 0$ implies $p_i = 0$. Thus, we redefine $p_i \equiv p_i(b_i) \equiv b_i\widehat{p}_i$ so that player i has *intrinsic risk* $\widehat{p}_i$, and only has *internal risk* if targeted (i.e, $b_i = 1$). The new parameter $\widehat{p}_i$ represents the (*conditional*) probability that an attack is successful at player i *given* that player i was directly targeted and did not invest in protection. The game parameter q_{ij} "encodes" $b_i = 1$, because a prerequisite is that i is targeted before it can transfer the bad event to j. We redefine $q_{ij} \equiv q_{ij}(b_i) \equiv b_i\widehat{q}_{ij}$ so that $\widehat{q}_{ij}$ is the intrinsic transfer probability from player i to player j, independent of b_i. The new parameter $\widehat{q}_{ij}$ represents the (*conditional*) probability that an attack is successful at player j *given* that it originated at player i, did not occur at i but was transferred undetected to j. Note that just as it was the case with traditional IDS games, there is an implicit constraint on the risk-related parameters: $\widehat{p}_i + \sum_{j\in \mathrm{Ch}(i)} \widehat{q}_{ij} \leq 1$, for all i. Because the p_i's and q_{ij}'s depend on the attacker's action $\mathbf{b}$, so does the safety and risk functions. In particular, we now have

$$e_{ij}(a_j, b_j) \equiv a_j + (1-a_j)(1-b_j\widehat{q}_{ji}) = (1-\widehat{q}_{ji})^{b_j(1-a_j)},$$

$s_i(\mathbf{a}_{\mathrm{Pa}(i)}, \mathbf{b}_{\mathrm{Pa}(i)}) \equiv \prod_{j\in\mathrm{Pa}(i)} e_{ij}(a_j, b_j) \equiv 1 - r_i(\mathbf{a}_{\mathrm{Pa}(i)}, \mathbf{b}_{\mathrm{Pa}(i)})$. Hence, for each player i, the *cost* function becomes

$$M_i(a_i, \mathbf{a}_{\mathrm{Pa}(i)}, b_i, \mathbf{b}_{\mathrm{Pa}(i)}) \equiv a_i[C_i + \alpha_i r_i(\mathbf{a}_{\mathrm{Pa}(i)}, \mathbf{b}_{\mathrm{Pa}(i)})L_i] + (1-a_i)[b_i\widehat{p}_i + (1-b_i\widehat{p}_i)r_i(\mathbf{a}_{\mathrm{Pa}(i)}, \mathbf{b}_{\mathrm{Pa}(i)})]L_i.$$

We assume the attacker wants to cause as much damage as possible. One possible *utility/payoff* function U quantifying the objective of the attacker is

$$U(\mathbf{a}, \mathbf{b}) \equiv \textstyle\sum_{i=1}^n M_i(\mathbf{a}_{\mathrm{PF}(i)}, \mathbf{b}_{\mathrm{PF}(i)}) - a_i C_i - b_i C_i^0.$$

which adds the expected players costs (for targeted and transferred bad events) over all players, minus C_i^0, the attacker's own *"cost" to target player i*.

3.1 MIXED STRATEGIES IN IDD GAMES

For all player i, denote by x_i the *mixed strategy of player i*: the probability that player i invests. Similarly, y denotes the joint probability mass function (PMF) corresponding to the *attacker's mixed strategy* so that for all $\mathbf{b} \in \{0,1\}^n$, $y(\mathbf{b})$ is the probability that the attacker executes joint-attack vector $\mathbf{b}$.

Denote the marginal PMF over a subset $I \subset [n]$ of the internal players by y_I such that for all $\mathbf{b}_I$, $y_I(\mathbf{b}_I) \equiv \sum_{\mathbf{b}_{-I}} y(\mathbf{b}_I, \mathbf{b}_{-I})$ is the (marginal) probability that the attacker chooses a joint-attack vector in which the subcomponent decisions corresponding to players in I are as in $\mathbf{b}_I$. Denote simply by $y_i \equiv y_{\{i\}}(1)$ the marginal probability that the attacker chooses an attack vector in which player i is directly targeted. Slightly abusing notation, we redefine the function e_{ij} (i.e., how safe i is from j), s_i and r_i (i.e., the overall transfer safety and risk, respectively) as $e_{ij}(x_j, b_j) \equiv x_j + (1-x_j)(1-b_j\widehat{q}_{ji})$, $s_i(\mathbf{x}_{\mathrm{Pa}(i)}, \mathbf{b}_{\mathrm{Pa}(i)}) \equiv \prod_{j\in\mathrm{Pa}(i)} e_{ij}(x_j, b_j)$,

$$s_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{Pa}(i)}) \equiv \sum_{\mathbf{b}_{\mathrm{Pa}(i)}} y_{\mathrm{Pa}(i)}(\mathbf{b}_{\mathrm{Pa}(i)}) s_i(\mathbf{x}_{\mathrm{Pa}(i)}, \mathbf{b}_{\mathrm{Pa}(i)}) \,,$$

and $r_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{Pa}(i)}) \equiv 1 - s_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{Pa}(i)})$.

In general, the *expected* cost of protection to site i, with respect to a joint mixed-strategy $(\mathbf{x}, y)$, can be expressed as

$$M_i(x_i, \mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{PF}(i)}) \equiv x_i[C_i + \alpha_i r_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{Pa}(i)})L_i] + (1-x_i)[\widehat{p}_i f_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{PF}(i)}) + r_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{Pa}(i)})]L_i \,,$$

where $f_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{PF}(i)}) \equiv \sum_{\mathbf{b}_{\mathrm{PF}(i)}} y_{\mathrm{PF}(i)}(\mathbf{b}_{\mathrm{PF}(i)})\, b_i s_i(\mathbf{x}_{\mathrm{Pa}(i)}, \mathbf{b}_{\mathrm{Pa}(i)})$.

The expected payoff of the attacker is

$$U(\mathbf{x}, y) \equiv \textstyle\sum_{i=1}^n M_i(\mathbf{x}_{\mathrm{PF}(i)}, y_{\mathrm{PF}(i)}) - x_i C_i - y_i C_i^0.$$

Let $\widehat{\Delta}_i \equiv \rho_i/\widehat{p}_i \equiv \frac{C_i}{L_i\widehat{p}_i}$ and $\widehat{s}_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{PF}(i)}) \equiv f_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{PF}(i)}) + \frac{1-\alpha_i}{\widehat{p}_i} r_i(x_{\mathrm{Pa}(i)}, y_{\mathrm{Pa}(i)})$. The best-response correspondence of defender i is then

$$\mathcal{BR}_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{PF}(i)}) \equiv \begin{cases} \{1\}, & \text{if } \widehat{s}_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{PF}(i)}) > \widehat{\Delta}_i, \\ \{0\}, & \text{if } \widehat{s}_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{PF}(i)}) < \widehat{\Delta}_i, \\ [0,1], & \text{if } \widehat{s}_i(\mathbf{x}_{\mathrm{Pa}(i)}, y_{\mathrm{PF}(i)}) = \widehat{\Delta}_i. \end{cases}$$

The best-response correspondence for the attacker is simply $\mathcal{BR}_0(\mathbf{x}) \equiv \arg\max_y U(\mathbf{x}, y)$.

Definition 1 *A joint mixed-strategy* $(\mathbf{x}^*, y^*)$ *is a* mixed-strategy Nash equilibrium (MSNE) *of an IDD game if (1) for all* $i \in [n]$, $x_i^* \in \mathcal{BR}_i(\mathbf{x}^*_{Pa(i)}, y^*_{\mathrm{PF}(i)})$ *and (2)* $y^* \in \mathcal{BR}_0(\mathbf{x}^*)$. *If* $(\mathbf{x}^*, y^*)$ *corresponds to a (deterministic)* joint action *then we call the MSNE a* pure-strategy Nash equilibrium (PSNE).

3.2 MODEL ASSUMPTIONS

Note that the attacker has in principle an *exponential* number of pure strategies! This affords the attacker unrealistic amount of power. Hence, we need restriction on the attacker's power. The simplest way is to allow at most a single simultaneous attack. We can weaken this assumption to allow the attacker at most K simultaneous attacks. Even then, the number of pure strategies will grow *exponentially* in the number of potential attacks, which still renders the attacker's pure-strategy space unrealistic, especially on a very large network with twenty-thousand nodes. Worst-case, we need to consider up to 2^n number of pure strategies for K attacks as K goes to n.

Assumption 1 *The set of pure strategies of the attacker is* $\mathcal{B} = \{\mathbf{b} \in \{0,1\}^n \mid \sum_{i=1}^n b_i \leq 1\}$.

The following assumptions are on the game *parameters*. The next assumption states that every site's investment cost is positive and (strictly) smaller than the *conditional* expected *direct* loss if the site were to be attacked directly ($b_i = 1$); that is, if a site knows that an attack is directed against it, the site will prefer to invest in security, unless the *external risk* is too high. This assumption is reasonable because otherwise the player will never invest regardless of what other players do (i.e., not investing would be a dominant strategy).

Assumption 2 *For all sites* $i \in [n]$, $0 < C_i < \widehat{p}_i L_i$.

The next assumption states that, for all sites i, the attacker's cost to attack i is positive and (strictly) smaller than the expected loss (i.e., gains from the perspective of the attacker) achieved if an attack initiated at site i is successful, either directly at i or at one of its children (after transfer); that is, if an attacker knows that an attack is rewarding (or able to obtain a positive utility), it will prefer to attack some nodes in the network. This assumption is reasonable; otherwise the attacker will never attack regardless of what other players do (i.e., not attacking would be a dominant strategy, leading to an easy problem to solve).

Assumption 3 *For all sites* $i \in [n]$, $0 < C_i^0 < \widehat{p}_i L_i + \sum_{j \in Ch(i)} \widehat{q}_{ij} \alpha_j L_j$.

PSNE of IDD Games. It turns out that under these three assumptions, there is no PSNE in IDD games. This is typical of attacker-defender settings. The following proposition eliminates PSNE as a universal solution concept for natural IDD games in which at most one attack is possible. The main significance of this result is that it allows us to concentrate our efforts on the much harder problem of computing MSNE.

Proposition 1 *No IDD game in which Assumptions 1, 2 and 3 hold has a PSNE.*

3.3 MSNE OF IDD GAMES

We first consider the IDD games under the assumption that no protection from transfer risk, which is used in the original IDS games. Note that, throughout this subsection, we will be using the assumptions we made earlier (in additional to the following assumption).

Assumption 4 *For all internal players* $i \in N$, *the probability that player* i*'s investment in security does not protect the player from transfers,* α_i, *is* 1.

Definition 2 *We say an IDD game is* transfer-vulnerable *if Assumption 4 holds. We say an IDD game is a* single simultaneous attack *game if Assumption 1 holds (i.e., at most one attack is possible).*

Assumption 1, in the context of mixed strategies, implies the probability of no attack $y_0 \equiv 1 - \sum_i^n y_i$. Assumptions 1 and 4 greatly simplify the best-response condition of the internal players because now $\widehat{s}_i(\mathbf{x}_{\text{Pa}(i)}, y_{\text{PF}(i)}) = y_i$. Let $L_i^0(x_i) \equiv (1 - x_i)(\widehat{p}_i L_i + \sum_{j \in \text{Ch}(i)} \widehat{q}_{ij} L_j)$. It will also be convenient to denote by $\overline{L}_i^0 \equiv L_i^0(0) = \widehat{p}_i L_i + \sum_{j \in \text{Ch}(i)} \widehat{q}_{ij} L_j$, so that we can express $L_i^0(x_i) = (1 - x_i)\overline{L}_i^0$, to highlight that L_i^0 is a linear function of x_i. Similarly, it will also be convenient to let $M_i^0(x_i) \equiv L_i^0(x_i) - C_i^0$, and denote $\overline{M}_i^0 \equiv M_i^0(0) = \overline{L}_i^0 - C_i^0$. Let $\eta_i^0 \equiv C_i^0 / \overline{L}_i^0$. The best-response condition of the attacker also simplifies under the same assumptions because now $U(\mathbf{x}, y) = \sum_{i=1}^n y_i M_i^0(x_i)$. Assumption 3 is reasonable in our new context because, under Assumption 4, if there were a player i with $\eta_i^0 > 1$, the attacker would never attack i, and as a result player i would never invest. In that case, we can safely remove j from the game, without any loss of generality.

We now characterize the space of MSNE in IDD games, which will immediately lead to a polynomial-time algorithm for computing *all* MSNE.

Characterization. The characterization starts by partitioning the space of games into three, based on whether $\sum_{i=1}^n \widehat{\Delta}_i$ is (1) $<$, (2) $=$, or (3) $>$ than 1. The rationale behind this is that now the players are indifferent between investing or not investing when $y_i = \widehat{\Delta}_i$, by the best-response correspondence the attacker's mixed strategy is restricted. The following result fully characterizes the set of MSNE in single simultaneous attack transfer-vulnerable IDD games.

Proposition 2 *The joint mixed-strategy* $(\mathbf{x}^*, y^*)$ *is an MSNE of a single simultaneous attack transfer-vulnerable IDD game in which*

1. $\sum_{i=1}^n \widehat{\Delta}_i < 1$ *if and only if (1)* $1 > y_0^* = 1 - \sum_{i=1}^n \widehat{\Delta}_i > 0$, *and (2) for all* i, $y_i^* = \widehat{\Delta}_i > 0$ *and* $0 < x_i^* = 1 - \eta_i^0 < 1$.

2. $\sum_{i=1}^n \widehat{\Delta}_i = 1$ *if and only if (1)* $y_0^* = 0$, *and (2) for all* i, $y_i^* = \widehat{\Delta}_i > 0$ *and* $x_i^* = 1 - \frac{v + C_i^0}{\overline{L}_i^0}$ *with* $0 \leq v \leq \min_{i \in [n]} \overline{M}_i^0$.

3. $\sum_{i=1}^n \widehat{\Delta}_i > 1$ *if and only if (1)* $y_0^* = 0$, *and (2) there exists a non-singleton, non-empty subset* $I \subset [n]$, *such that* $\min_{i \in I} \overline{M}_i^0 \geq \max_{k \notin I} \overline{M}_k^0$ *if* $I \neq [n]$, *and the following holds: (a) for all* $k \notin I$, $x_k^* = 0$ *and* $y_k^* = 0$, *(b) for all* $i \in J \equiv \arg\min_{i \in I} \overline{M}_i^0$, $x_i^* = 0$ *and* $0 \leq y_i^* \leq \widehat{\Delta}_i$,

and in addition, $\sum_{i\in J} y_i^* = 1 - \sum_{t\in I-J} \widehat{\Delta}_i$*; and (c) for all* $i \in I - J$*,* $y_i^* = \widehat{\Delta}_i$ *and* $0 < x_i^* = 1 - \frac{\min_{t\in I} \overline{M}_t^0 + C_i^0}{\overline{L}_i^0} < 1$.

As proof sketch, we briefly state that the proposition follows from the restrictions imposed by the model parameters and their implication to indifference and monotonicity conditions. We also mention that the third case in the proposition implies that if the $\overline{M}_l^0$'s form a complete order, then the last condition stated in that case allows us to search for a MSNE by exploring only $n-2$ sets, vs. 2^{n-2} if done naively. [6]

We now discuss properties of the characterization.

Security investment characteristics of MSNE. At equilibrium $\mathbf{x}^*$, if $x_i^* > 0$, the probability of *not* investing is proportional to C_i^0 and *inversely* proportional to $\widehat{p}_i\overline{L} + \sum_{j\in \mathrm{Ch}(i)} \widehat{q}_{ij} L_j$. It is kind of reassuring the at equilibrium, which is the (almost-surely) unique stable outcome of the system, the probability of investing increases with the potential loss a player's non-investment decision could cause to the system. Hence, behavior in a stable system implicitly "forces" all players to indirectly account for or take care of their own children. This may sound a bit paradoxical at first given that we are working within "noncooperative" setting and each player's cost function is only dependent on the investment decision of the player's *parents*. Interestingly, the existence of the attacker in the system is inducing an (almost-surely) unique stable outcome in which an implicit form of "cooperation" occurs. A defenders's best response is independent of their parents, the source of transfer risk, if investment in security does nothing to protect that player from transfers (i.e., $\alpha_i = 1$). This makes sense because the player cannot control the transfer risk.

Relation to network structure. How does the network structure and the equilibrium relate? As seen above, the values of the equilibrium strategy of each player depend on information from the attacker, the player and the player's children. From the discussion in the last paragraph, a player's probability of investing at the equilibrium increases with the expected loss sustained from a "bad event" occurring as a result of a transfer from a player to the player's children.

Let us explore this last point further by considering the case of uniform-transfer probabilities (also studied by Kunreuther and Heal [2003] and Kearns and Ortiz [2003]). In that case, transfer probabilities are only a function of the source, not the destination: $\widehat{q}_{ij} \equiv \widehat{\delta}_i$. The expression for the equilibrium probabilities of those players who have a positive probability of investing would simplify to $x_i^* = 1 - \frac{v + C_i^0}{\widehat{p}_i L_i + \delta_i \sum_{j\in \mathrm{Ch}(i)} L_j}$, for some constant v. The last expression suggests that $\sum_{j\in \mathrm{Ch}(i)} L_j$ differentiates the probability of investing between players. That would suggests the larger the number of children the larger the probability of investing. A scenario that seems to further lead us to that conclusion is when we make further assumptions (homogeneous system: first studied in the original IDS paper): $L_i \equiv L$, $\widehat{p}_i \equiv \widehat{p}$, $\delta_i \equiv \delta$, and $C_i^0 \equiv C^0$ [7] for all players. Then, we would get $x_i^* = 1 - \frac{v + C^0}{L(\widehat{p} + \delta|\mathrm{Ch}(i)|)}$. So the probability of *not* investing is inversely proportional to the *number* of children the player has.

On the attacker's equilibrium strategy. The support of the attacker, $I^* \equiv \{i \mid y_i^* > 0\}$, at equilibrium has the following properties: (1) players for which the attacker's cost-to-expected-loss is higher are "selected" first in the algorithm; (2) if the size of that set is t, and there is a lower bound on $\widehat{\Delta}_i > \widehat{\Delta}$, and $\sum_{i=1}^n \widehat{\Delta}_i > 1$, then $t < 1/\widehat{\Delta}$ is an upper-bound on the number of players that could potentially be attacked; (3) if we have a game with homogeneous parameters, then the probability of an attack will be uniform over that set I^*; and (4) all but one of the players in that set I^* invest in security with some non-zero probability (almost surely).

3.4 COMPUTING MSNE EFFICIENTLY

We now describe an algorithm to compute *all* MSNE in single simultaneous attack transfer-vulnerable IDD games that falls off the Proposition 2. We begin by noting that the equilibrium in the case of IDD games with $\sum_{i=1}^n \widehat{\Delta}_i \leq 1$, corresponding to cases 1 and 2 of the proposition, has essentially an analytic closed-form. Hence, we concentrate on the remaining and most realistic case in large-population games of $\sum_{i=1}^n \widehat{\Delta}_i > 1$. We start by sorting the indices of the internal players in descending order based on the $\overline{M}_i^0$'s. Let $\mathrm{Val}(l)$ and $\mathrm{Idx}(l)$ be the lth value and index in the resulting sorted list, respectively. Find t such that $1 - \widehat{\Delta}_{\mathrm{Idx}(t)} \leq \sum_{l=1}^{t-1} \widehat{\Delta}_{\mathrm{Idx}(l)} < 1$. Let $k = \arg\max\{l \geq t \mid \mathrm{Val}(l) = \mathrm{Val}(t)\}$ (i.e., continue down the sorted list of values until a change occurs). For $i = 1, \ldots, t-1$, let $l = \mathrm{Idx}(i)$ and set $x_l^* = 1 - \frac{\mathrm{Val}(t) + C_l^0}{\overline{L}_l^0}$ and $y_l^* = \widehat{\Delta}_l$. For $i = k+1, \ldots, n$, let $l = \mathrm{Idx}(i)$ and set $x_l^* = 0$ and $y_l^* = 0$. For $i = t, \ldots, k$, let $l = \mathrm{Idx}(i)$ and set $x_l^* = 0$. Finally, represent the simplex defined by the following constraints: for $i = t, \ldots, k$, let $l = \mathrm{Idx}(i)$ and $0 \leq y_l^* \leq \widehat{\Delta}_l$;

[6] In fact, it turns out a complete order is not actually necessary because we can safely move all defenders with the same value of $\overline{M}_i^0$ in a group as a whole inside or outside the set I referred to in that case of the proposition.

[7] Note that this does not mean that the expected loss caused by a player that does not invest but is attacked, $L(\widehat{p} + \delta|\mathrm{Ch}(i)|)$, is the same for all players.

$\sum_{i=t}^{k} y^*_{\mathrm{Idx}(i)} = 1 - \sum_{i=1}^{t-1} \widehat{\Delta}_{\mathrm{Idx}(i)}$. The running time of the algorithm is $O(n \log n)$ (because of sorting). As mentioned earlier, the theorem is significant because we do not know of any other non-trivial class of games that have algorithms to compute all Nash equilibria in polynomial time, except ours and the uniform IDS games [Kearns and Ortiz, 2003].

Theorem 1 *There exists a polynomial-time algorithm to compute all MSNE of a single simultaneous attack transfer-vulnerable IDD game.*

Note that the case in which $\sum_{i=1}^{n} \widehat{\Delta}_i = 1$ has (Borel) measure zero and is quite brittle (i.e., adding or removing a player breaks the equality). For the case in which $\sum_{i=1}^{n} \widehat{\Delta}_i > 1$, if the value of the $\overline{M}_i^0$'s are distinct, [8] then there is a unique MSNE!

4 EXPERIMENTS

In the previous section, we established the theoretical characteristics and computational tractability of single simultaneous attack IDD games with the highest transfer vulnerability parameter: $\alpha_i = 1$. In this section, partly motivated by security problems in cyberspace, we concentrate instead on empirically evaluating the other extreme of transfer vulnerability: games with low α_i values (i.e., near 0), so that investing in security considerably reduces the transfer risk.

Our main objectives for the experiments presented here are (1) to demonstrate that a simple heuristic, *best-response-gradient dynamics (BRGD)*, is practically effective in computing an (approximate) MSNE in a very large class of IDD games with realistic Internet-scale network graphs in a reasonable amount of time for cases in which the transfer vulnerabilities α_i's are low; and (2) to explore the general structural and computational characteristics of (approximate) MSNE in such IDD games, including their dependence on the underlying network structure of the game (and approximation quality).

BRGD is a well-known technique from the work on learning in games [Fudenberg and Levine, 1999, Singh et al., 2000, Kearns and Ortiz, 2003, Heal and Kunreuther, 2005, Kearns, 2005]. Here, we use BRGD as a tool to *compute* an ϵ-approximate MSNE, which is a joint mixed strategy with the property that the gain in utility (or reduction in cost) of any individual from unilaterally deviating from their prescribed mixed-strategy is no larger than ϵ; a 0-approximate MSNE is an exact MSNE.

[8] Distinct $\overline{M}_i^0$'s for the set of defenders at which the sum goes over one is sufficient to guarantee unique MSNE.

Table 1: Internet Games' Model Parameters

Model Parameters	Fixed: U = 0.5 Random: U ∼ Uniform([0,1])
α_i	$U/20$
L_i	$10^8 + (10^9) * U$
C_i	$10^5 + (10^6) * U$
$\widehat{p}_i$	$0.9 * \frac{\widetilde{p}_i}{\widetilde{p}_i + \sum_{k \in \mathrm{Ch}(i)} \widetilde{q}_{ik}}$
$\widehat{q}_{ij}$	$0.9 * \frac{\widetilde{q}_{ij}}{\widetilde{p}_i + \sum_{k \in \mathrm{Ch}(i)} \widetilde{q}_{ik}}$
z_i	$0.2 + U/5$
$\widetilde{p}_i$	$0.8 + U/10$
$\widetilde{q}_{ij}$	$z_i \frac{\lvert\mathrm{Ch}(j)\rvert + \lvert\mathrm{Pa}(j)\rvert}{\sum_{k \in \mathrm{Ch}(i)} \lvert\mathrm{Ch}(k)\rvert + \lvert\mathrm{Pa}(k)\rvert}$
C_i^0	10^6

We obtained the latest version (March 2010 at the time) of the real structure and topology of the Autonomous Systems (AS) in the Internet from DIMES (netdimes.org) [Shavitt and Shir, 2005]. The AS-level network has 27,106 nodes (683 isolated) and 100,402 directed edges; the graph length (diameter) is 6,253, the density (number of edges divided by number of possible edges) is 1.9920×10^{-5}, and the average (in and out) degree is 3.70, with $\approx$ 76.93% and 2.59% of the nodes having zero indegree and outdegree, respectively. *All the IDD games in the experiments presented in this section have this network structure.*

For simplicity, we call *Internet games* the class of IDD games with the AS-level network graph and low α_i values. We considered various settings for model parameters of Internet games: a single instance with specific fixed values; and several instances generated at random (see Table 1 for details). The attacker's cost-to-attack parameter for each node i is always held constant: $C_i^0 = 10^6$. For each run of each experiment, we ran BRGD with randomly-generated initial conditions (i.e., random initializations of the players' mixed strategies): $x_i \sim \mathrm{Uniform}([0,1])$, i.i.d. for all i, and y is a probability distribution generated uniformly at random, and independent of $\mathbf{x}$, from the set of all probability mass functions over $n+1$ events. [9] The initialization of the transfer-probability parameters of a node essentially gives higher transfer probability to children with high (total) degree (because they are potentially "more popular"). The initialization also enforces $\widehat{p}_i + \sum_{j \in \mathrm{Ch}(i)} \widehat{q}_{ij} = 0.9$. Other initializations are possible but we did not explore then here.

4.1 COMPUTING ϵ-MSNE USING BRGD

Given the lack of theoretical guarantees on the convergence rate of BRGD, we began our empirical study by evaluating the convergence and computation/running-time behavior of BRGD on Internet games. We ran

[9] Recall the probability of no attack $y_0 = 1 - \sum_{i=1}^{n} y_i$.

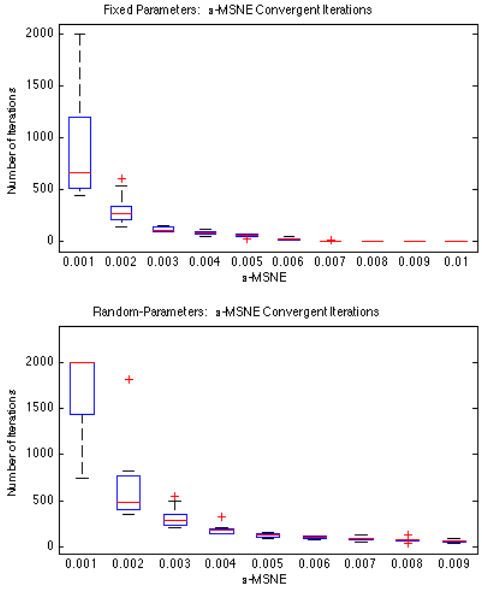


Figure 1: **Convergence Rate of Learning Dynamics:** The plots above present the number of iterations of BRGD as a function of ϵ under the two experimental conditions: Internet games with fixed (top) and randomly-generated parameters (bottom). Applying MSE regression to the top and bottom graphs, we obtain a functional expression for the number of iterations $N^F(\epsilon) = 0.00003\epsilon^{-2.547}$ ($R^2 = 0.90415$) and $N^R(\epsilon) = 0.0291\epsilon^{-1.589}$ ($R^2 = 0.9395$), respectively (i.e., low-degree polynomials of $1/\epsilon$).

ten simulations for each ϵ value and recorded the number of iterations until convergence (up to $2,000$ iterations). Figure 1 presents the number of iterations taken by BRGD to compute an ϵ-MSNE as a function of ϵ. All simulations in this experiment converged (except for $\epsilon = 0.001$: 2 and all of the runs for single and randomly-generated instances, respectively, did not). Each iteration took roughly 1-2 sec. (wall clock). Hence, we can use BRGD to consistently compute an ϵ-MSNE of a 27K-players Internet game in a few seconds!

4.2 CHARACTERISTICS OF THE ϵ-MSNE

We now concentrate on the empirical study of the *structural* characteristics of the ϵ-MSNE found by BRGD. We experimented on both the single and randomly-generated Internet game instances. We discuss the typical behavior of the attacker and the sites in an ϵ-MSNE, and the typical relationship between ϵ-MSNE and network structure.

4.2.1 A Single Internet Game

We first studied the characteristics of the ϵ-MSNE of a single Internet game instance. The only source of randomness in these experiments comes from BRGD's

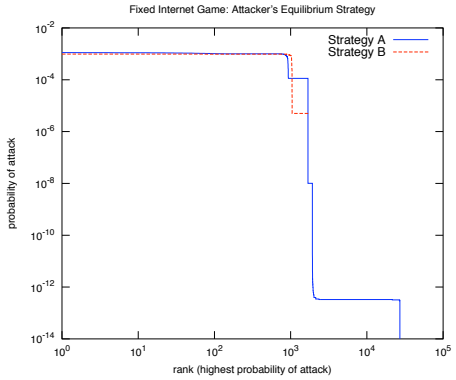


Figure 2: **Attacker's Equilibrium Strategy on an Internet Game Instance (Fixed):** The graph shows the values of $y_i^* > 0$ for each node i, sorted in decreasing order (in log-log scale), for attacker's Strategy A (blue/denser-dots line) and Strategy B (red/sparser-dots line) at an MSNE of the single instance of the Internet game.

initial conditions (i.e., the initialization of the mixed strategies $\mathbf{x}$ and $\mathbf{y}$). BRGD consistently found *exact* MSNE (i.e., $\epsilon = 0$) in *all* runs.

Players' equilibrium behavior. In fact, we consistently found that the attacker always displays only two types of "extreme" equilibrium behavior, corresponding to the two kinds of MSNE BRGD found for the single Internet game: place positive probability of a direct attack to either *almost all* nodes (Strategy A) or a *small subset* (Strategy B). Figure 2 shows a plot of the typical probability of direct attack for those two equilibrium strategies for the attacker when BRGD stops. In both strategies, a relatively small number of nodes (about 1K out of 27K) have a reasonably *high* (and near *uniform*) probability of direct attack. In Strategy A, however, *every* node has a positive probability of being the target of a direct attack, albeit relatively very low for most; this is contrast to Strategy B where *most* nodes are fully immune from a direct attack. Interestingly, *none* of the nodes invest in either MSNE: $x_i^* = 0$ for all nodes i. Thus, in this particular Internet game instance, *all* site nodes are willing to risk an attack!

Relation to network structure. We found that the nodes with (relatively) high probability of direct attack are at the "fringe" of the graph (i.e., have low or no degree). In Strategy A, fringe nodes (with mostly 0 or 1 outdegree) have relatively higher probability of direct attack than nodes with higher outdegree. Similarly, in Strategy B, the small subset of nodes that are potential target of a direct attack have relatively low outdegree (mostly 0, and 0.0067 on average; this is in contrast to the average outdegree of 3.9639 for the nodes immune from direct attack). In short, we consistently found that the nodes with low outdegree

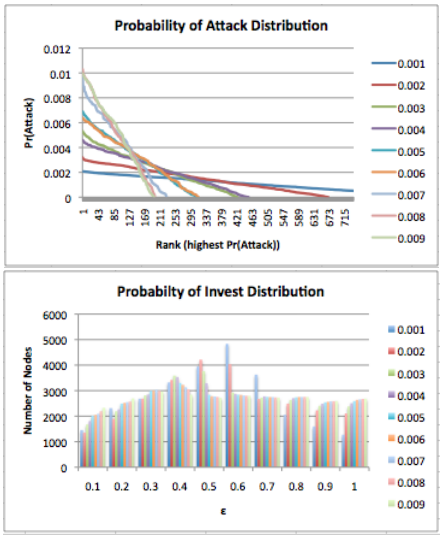


Figure 3: **Attacker's and Site's ϵ-MSNE Strategies for Randomly-Generated Internet Games:** The graphs show the empirical distributions of the probability of attack (top) and histograms of the probability of investment (bottom), for different ϵ-value conditions encoded in the right-hand side of the plots (i.e., from 0.001 to 0.009). In both graphs, the distributions and histograms found for each ϵ value considered are drawn in the same corresponding graph superimposed. The top graph plots the distribution of y_i where the nodes are ordered decreasingly based on the y_i value. The bottom bar graph shows histograms of the probability of investing in defense/security measures based on the following sequence of 10 ranges partitioning the unit interval: $([0, 0.1], (0.1, 0.2], ..., (0.9, 1])$.

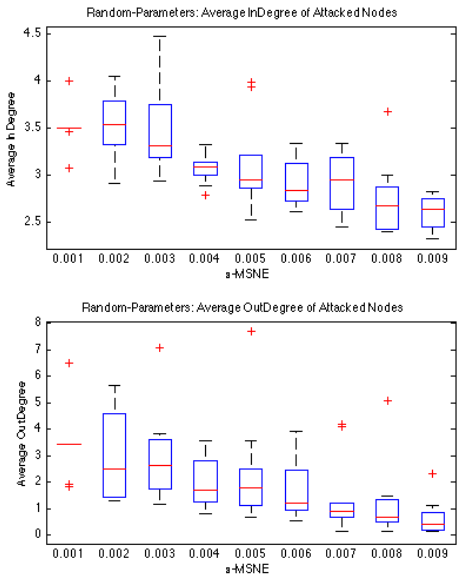


Figure 4: **Attacker's ϵ-MSNE Strategy vs. Node Degree:** Average indegree (top) and outdegree (bottom) of nodes potentially attacked in terms of the ϵ-MSNE.

are more likely to get attacked directly in the single Internet game instance.

4.2.2 Randomly-Generated Internet Games

We now present results from experiments on randomly-generated instances of the Internet game, a single instance for each $\epsilon \in \{0.001, 0.002, \ldots, 0.009\}$. For simplicity, we present the result of a single BRGD run on each instance. [10]

Behavior of the players. Figure 3 shows plots of the attacker's probability of direct attack and histograms of the nodes's probability of investment in a typical run of BRDG on each Internet game instance randomly-generated for each ϵ value. The plots suggest that approximate MSNE found by BRGD is quite sensitive to the ϵ value: as ϵ decreases, the attacker tends to "spread the risk" by selecting a larger set of nodes as potential targets for a direct attack, thus lowering the probability of a direct attack on any individual node; the nodes, on the other hand, tend to deviate from (almost) fully investing and (almost) not investing to a more uniform mixed strategy (i.e., investing or not investing with roughly equal probability). A possible reason for this is that as more nodes become potential targets of a direct attack, more nodes with initial mixed strategies close to the "extreme" (i.e., very high or very low probabilities of investing) will move closer to a more uniform (and thus less "predictable") investment distribution.

Relation to network structure. Figure 4 presents some experimental results on the relationship between network structure and the attacker's equilibrium behavior. The graphs show, for each ϵ value, the average indegree and outdegree, across the ten BRGD runs, of those nodes that are potential targets of a direct attack at an ϵ-MSNE. In general, both the average indegree and outdegree of the nodes that are potential targets of a direct attack tend to increase as ϵ decreases. One possible reason for this finding could be the fact that the values of α_i generated for each player are relatively low (i.e., uniformly distributed over $\left[0, \frac{1}{40}\right]$); yet, interestingly, such behavior and pattern, is exact opposite of the theory for the case $\alpha_i = 1$.

[10] While the results presented here are for a single instance of the Internet game for each ϵ, the results are typical of multiple instances. Our observations are robust to the experimental randomness in both the Internet game parameters and the initialization of BRGD. For the sake of simplicity of presentation, we discuss results based on a single instance of the Internet game, and in some cases based on a single BRGD run. Note that, for each ϵ value we considered, the Internet game parameters remain constant within different BRGD runs. BRGD always converged within $2,000$ iterations (except 6 runs for $\epsilon = 0.001$).

References

Vicki M. Bier and M. Naceur Azaiez, editors. *Game Theoretic Risk Analysis of Security Threats*. Springer, 2009.

Eva Cárceles-Poveda and Yair Tauman. Strategic aspects of terrorism. *Games and Economic Behavior*, 2011. Forthcoming.

Hau Chan, Michael Ceyko, and Luis E. Ortiz. Interdependent defense games: Modeling interdependent security under deliberate attacks (Supplementary Material). `http://www.cs.stonybrook.edu/~leortiz/papers/dg_UAI_SM_final.pdf`, June 2012.

Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *J. ACM*, 56(3):1–57, 2009.

Vincent Conitzer and Tuomas Sandholm. New complexity results about Nash equilibria. *Games and Economic Behavior*, 63(2):621 – 641, 2008. Second World Congress of the Game Theory Society.

Marco Cremonini and Dmitri Nizovtsev. Understanding and influencing attackers' decisions: Implications for security investment strategies. In *The Fifth Workshop on the Economics of Information Security (WEIS 2006)*, 2006.

Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *Commun. ACM*, 52(2): 89–97, 2009.

D. Fudenberg and D. Levine. *The Theory of Learning in Games*. MIT Press, 1999.

Neal Fultz and Jens Grossklags. Blue versus red: Towards a model of distributed security attacks. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security*, pages 167–183. Springer-Verlag, Berlin, Heidelberg, 2009.

I. Gilboa and E. Zemel. Nash and correlated equilibria: some complexity considerations. *Games and Economic Behavior*, 1:80–93, 1989.

Konstantinos Gkonis and Harilaos Psaraftis. Container transportation as an interdependent security problem. *Journal of Transportation Security*, 3:197–211, 2010.

Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Pure Nash equilibria: Hard and easy games. *J. Artif. Int. Res.*, 24(1):357–406, 2005.

Geoffrey Heal and Howard Kunreuther. IDS models of airline security. *Journal of Conflict Resolution*, 49 (2):201–217, April 2005.

Geoffrey Heal and Howard Kunreuther. Modeling interdependent risks. *Risk Analysis*, 27:621–634, July 2007.

Manish Jain, Dmytro Korzhyky, Ondrej Vanek, Vincent Conitzery, Michal Pechoucek, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, 2011.

Benjamin Johnson, Jens Grossklags, Nicolas Christin, and John Chuang. Uncertainty in interdependent security games. In *Proceedings of the First International Conference on Decision and Game Theory for Security*, GameSec'10, pages 234–244, Berlin, Heidelberg, 2010. Springer-Verlag.

M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 253–260, 2001.

Michael Kearns. Economics, computer science, and policy. *Issues in Science and Technology*, Winter 2005.

Michael Kearns. Graphical games. In Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*, chapter 7, pages 159–180. Cambridge University Press, 2007.

Michael Kearns and Luis E. Ortiz. Algorithms for interdependent security games. In *Neural Information Processing Systems (NIPS)*, 2003.

Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, pages 689–696, 2009.

Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *AAAI*, 2010.

Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Security games with multiple attacker resources. In *IJCAI*, pages 273–279, 2011a.

Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Solving Stackelberg games with uncertain observability. In *AAMAS*, pages 1013–1020, 2011b.

Howard Kunreuther and Geoffrey Heal. Interdependent security. *Journal of Risk and Uncertainty*, 26 (2-3):231–249, March 2003.

Peng Liu. Incentive-based modeling and inference of attacker intent, objectives, and strategies. In *Proc. of the 10th ACM Computer and Communications Security Conference (CCS'03)*, pages 179–189, 2003.

Kong-Wei Lye and Jeannette Wing. Game strategies in network security. In *Proceedings of the Workshop on Foundations of Computer Security*, pages 1–2, 2002.

Anahad O'Connor and Eric Schmitt. Terror attempt seen as man tries to ignite device on jet. The

New York Times, 25 December 2009. Cited 31 August 2010. Avaliable at `http://www.nytimes.com/2009/12/26/us/26plane.html`.

Sankardas Roy, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. A survey of game theory as applied to network security. In *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*, HICSS '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

Yuval Shavitt and Eran Shir. DIMES: Let the Internet measure itself. *ACM SIGCOMM Computer Communication Review*, 35(5):71–74, October 2005.

Satinder P. Singh, Michael J. Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *UAI*, pages 541–548, 2000.

Decentralized Data Fusion and Active Sensing with Mobile Sensors for Modeling and Predicting Spatiotemporal Traffic Phenomena

Jie Chen[†], **Kian Hsiang Low**[†], **Colin Keng-Yan Tan**[†], **Ali Oran**[§]
Dept. Computer Science, National University of Singapore[†]
Singapore-MIT Alliance for Research and Technology[§]
Republic of Singapore

Patrick Jaillet[‡], **John Dolan**[¶], **Gaurav Sukhatme**[*]
Dept. Electrical Engineering and Computer Science, MIT[‡]
The Robotics Institute, Carnegie Mellon University[¶]
Dept. Computer Science, University of Southern California[*]
USA

Abstract

The problem of modeling and predicting spatiotemporal traffic phenomena over an urban road network is important to many traffic applications such as detecting and forecasting congestion hotspots. This paper presents a decentralized data fusion and active sensing (D^2FAS) algorithm for mobile sensors to actively explore the road network to gather and assimilate the most informative data for predicting the traffic phenomenon. We analyze the time and communication complexity of D^2FAS and demonstrate that it can scale well with a large number of observations and sensors. We provide a theoretical guarantee on its predictive performance to be equivalent to that of a sophisticated centralized sparse approximation for the Gaussian process (GP) model: The computation of such a sparse approximate GP model can thus be parallelized and distributed among the mobile sensors (in a Google-like MapReduce paradigm), thereby achieving efficient and scalable prediction. We also theoretically guarantee its active sensing performance that improves under various practical environmental conditions. Empirical evaluation on real-world urban road network data shows that our D^2FAS algorithm is significantly more time-efficient and scalable than state-of-the-art centralized algorithms while achieving comparable predictive performance.

1 Introduction

Knowing and understanding the traffic conditions and phenomena over road networks has become increasingly important to the goal of achieving smooth-flowing, congestion-free traffic, especially in densely-populated urban cities. According to a 2011 urban mobility report (Schrank *et al.*, 2011), traffic congestion in the USA has caused 1.9 billion gallons of extra fuel, 4.8 billion hours of travel delay, and \$101 billion of delay and fuel cost. Such huge resource wastage can potentially be mitigated if the spatiotemporally varying traffic phenomena (e.g., speeds and travel times along road segments) are predicted accurately enough in real time to detect and forecast the congestion hotspots; network-level (e.g., ramp metering, road pricing) and user-level (e.g., route replanning) measures can then be taken to relieve this congestion, so as to improve the overall efficiency of road networks.

In practice, it is non-trivial to achieve real-time, accurate prediction of a spatiotemporally varying traffic phenomenon because the quantity of sensors that can be deployed to observe an entire road network is cost-constrained. Traditionally, static sensors such as loop detectors (Krause *et al.*, 2008a; Wang and Papageorgiou, 2005) are placed at designated locations in a road network to collect data for predicting the traffic phenomenon. However, they provide sparse coverage (i.e., many road segments are not observed, thus leading to data sparsity), incur high installation and maintenance costs, and cannot reposition by themselves in response to changes in the traffic phenomenon. Low-cost GPS technology allows the collection of traffic data using passive mobile probes (Work *et al.*, 2010) (e.g., taxis/cabs). Unlike static sensors, they can directly measure the travel times along road segments. But, they provide fairly sparse coverage due to low GPS sampling frequency (i.e., often imposed by taxi/cab companies) and no control over their routes, incur high initial implementation cost, pose privacy issues, and produce highly-varying speeds and travel times while traversing the same road segment due to inconsistent driving behaviors. A critical mass of probes is needed on each road segment to ease the severity of the last drawback (Srinivasan and Jovanis, 1996) but is often hard to achieve on non-highway segments due to sparse coverage. In contrast, we propose the use of active mobile probes (Turner *et al.*, 1998) to overcome the limitations of static and passive mobile probes. In particular, they can be directed to explore any segments of a road network to gather traffic data at a desired GPS sampling rate while enforcing consistent driving behavior.

How then do the mobile probes/sensors actively explore a road network to gather and assimilate the most informative

observations for predicting the traffic phenomenon? There are three key issues surrounding this problem, which will be discussed together with the related works:

Models for predicting spatiotemporal traffic phenomena. The spatiotemporal correlation structure of a traffic phenomenon can be exploited to predict the traffic conditions of any unobserved road segment at any time using the observations taken along the sensors' paths. To achieve this, existing Bayesian filtering frameworks (Chen *et al.*, 2011; Wang and Papageorgiou, 2005; Work *et al.*, 2010) utilize various handcrafted parametric models predicting traffic flow along a highway stretch that only correlate adjacent segments of the highway. So, their predictive performance will be compromised when the current observations are sparse and/or the actual spatial correlation spans multiple segments. Their strong Markov assumption further exacerbates this problem. It is also not shown how these models can be generalized to work for arbitrary road network topologies and more complex correlation structure. Existing multivariate parametric traffic prediction models (Kamarianakis and Prastacos, 2003; Min and Wynter, 2011) do not quantify uncertainty estimates of the predictions and impose rigid spatial locality assumptions that do not adapt to the true underlying correlation structure.

In contrast, we assume the traffic phenomenon over an urban road network (i.e., comprising the full range of road types like highways, arterials, slip roads) to be realized from a rich class of Bayesian non-parametric models called the *Gaussian process* (GP) (Section 2) that can formally characterize its spatiotemporal correlation structure and be refined with growing number of observations. More importantly, GP can provide formal measures of predictive uncertainty (e.g., based on variance or entropy criterion) for directing the sensors to explore highly uncertain areas of the road network. Krause *et al.* (2008a) used GP to represent the traffic phenomenon over a network of only highways and defined the correlation of speeds between highway segments to depend only on the geodesic (i.e., shortest path) distance of these segments with respect to the network topology; their features are not considered. Neumann *et al.* (2009) maintained a mixture of two independent GPs for flow prediction such that the correlation structure of one GP utilized road segment features while that of the other GP depended on manually specified relations (instead of geodesic distance) between segments with respect to an undirected network topology. Different from the above works, we propose a relational GP whose correlation structure exploits the geodesic distance between segments based on the topology of a directed road network with vertices denoting road segments and edges indicating adjacent segments weighted by dissimilarity of their features, hence tightly integrating the features and relational information.

Data fusion. The observations are gathered distributedly by each sensor along its path in the road network and have to be assimilated in order to predict the traffic phenomenon. Since a large number of observations are expected to be collected, a centralized approach to GP prediction cannot be performed in real time due to its cubic time complexity.

To resolve this, we propose a decentralized data fusion approach to efficient and scalable approximate GP prediction (Section 3). Existing decentralized and distributed Bayesian filtering frameworks for addressing non-traffic related problems (Chung *et al.*, 2004; Coates, 2004; Olfati-Saber, 2005; Rosencrantz *et al.*, 2003; Sukkarieh *et al.*, 2003) will face the same difficulties as their centralized counterparts described above if applied to predicting traffic phenomena, thus resulting in loss of predictive performance. Distributed regression algorithms (Guestrin *et al.*, 2004; Paskin and Guestrin, 2004) for static sensor networks gain efficiency from spatial locality assumptions, which cannot be exploited by mobile sensors whose paths are not constrained by locality. Cortes (2009) proposed a distributed data fusion approach to approximate GP prediction based on an iterative Jacobi overrelaxation algorithm, which incurs some critical limitations: (a) the past observations taken along the sensors' paths are assumed to be uncorrelated, which greatly undermines its predictive performance when they are in fact correlated and/or the current observations are sparse; (b) when the number of sensors grows large, it converges very slowly; (c) it assumes that the range of positive correlation has to be bounded by some factor of the communication range. Our proposed decentralized algorithm does not suffer from these limitations and can be computed exactly with efficient time bounds.

Active sensing. The sensors have to coordinate to actively gather the most informative observations for minimizing the uncertainty of modeling and predicting the traffic phenomenon. Existing centralized (Low *et al.*, 2008, 2009a, 2011) and decentralized (Low *et al.*, 2012; Stranders *et al.*, 2009) active sensing algorithms scale poorly with a large number of observations and sensors. We propose a partially decentralized active sensing algorithm that overcomes these issues of scalability (Section 4).

This paper presents a novel *Decentralized Data Fusion and Active Sensing* (D^2FAS) algorithm (Sections 3 and 4) for sampling spatiotemporally varying environmental phenomena with mobile sensors. Note that the decentralized data fusion component of D^2FAS can also be used for static and passive mobile sensors. The practical applicability of D^2FAS is not restricted to traffic monitoring; it can be used in other environmental sensing applications such as mineral prospecting (Low *et al.*, 2007), monitoring of ocean and freshwater phenomena (Dolan *et al.*, 2009; Podnar *et al.*, 2010; Low *et al.*, 2009b, 2011, 2012) (e.g., plankton bloom, anoxic zones), forest ecosystems, pollution (e.g., oil spill), or contamination (e.g., radiation leak). The specific contributions of this paper include:

- Analyzing the time and communication overheads of

D^{2}FAS (Section 5): We prove that D^{2}FAS can scale better than existing state-of-the-art centralized algorithms with a large number of observations and sensors;
- Theoretically guaranteeing the predictive performance of the decentralized data fusion component of D^{2}FAS to be equivalent to that of a sophisticated centralized sparse approximation for the GP model (Section 3): The computation of such a sparse approximate GP model can thus be parallelized and distributed among the mobile sensors (in a Google-like MapReduce paradigm), thereby achieving efficient and scalable prediction;
- Theoretically guaranteeing the performance of the partially decentralized active sensing component of D^{2}FAS, from which various practical environmental conditions can be established to improve its performance;
- Developing a relational GP model whose correlation structure can exploit both the road segment features and road network topology information (Section 2.1);
- Empirically evaluating the predictive performance, time efficiency, and scalability of the D^{2}FAS algorithm on a real-world traffic phenomenon (i.e., speeds of road segments) dataset over an urban road network (Section 6): D^{2}FAS is more time-efficient and scales significantly better with increasing number of observations and sensors while achieving predictive performance close to that of existing state-of-the-art centralized algorithms.

2 Relational Gaussian Process Regression

The *Gaussian process* (GP) can be used to model a spatiotemporal traffic phenomenon over a road network as follows: The traffic phenomenon is defined to vary as a realization of a GP. Let V be a set of road segments representing the domain of the road network such that each road segment $s \in V$ is specified by a p-dimensional vector of features and is associated with a realized (random) measurement z_s (Z_s) of the traffic condition such as speed if s is observed (unobserved). Let $\{Z_s\}_{s\in V}$ denote a GP, that is, every finite subset of $\{Z_s\}_{s\in V}$ follows a multivariate Gaussian distribution (Rasmussen and Williams, 2006). Then, the GP is fully specified by its *prior* mean $\mu_s \triangleq \mathbb{E}[Z_s]$ and covariance $\sigma_{ss'} \triangleq \mathrm{cov}[Z_s, Z_{s'}]$ for all $s, s' \in V$. In particular, we will describe in Section 2.1 how the covariance $\sigma_{ss'}$ for modeling the correlation of measurements between all pairs of segments $s, s' \in V$ can be designed to exploit the road segment features and the road network topology.

A chief capability of the GP model is that of performing probabilistic regression: Given a set $D \subset V$ of observed road segments and a column vector z_D of corresponding measurements, the joint distribution of the measurements at any set $Y \subseteq V \setminus D$ of unobserved road segments remains Gaussian with the following *posterior* mean vector and covariance matrix

$$\mu_{Y|D} \triangleq \mu_Y + \Sigma_{YD}\Sigma_{DD}^{-1}(z_D - \mu_D) \quad (1)$$

$$\Sigma_{YY|D} \triangleq \Sigma_{YY} - \Sigma_{YD}\Sigma_{DD}^{-1}\Sigma_{DY} \quad (2)$$

where μ_Y (μ_D) is a column vector with mean components μ_s for all $s \in Y$ ($s \in D$), Σ_{YD} (Σ_{DD}) is a covariance matrix with covariance components $\sigma_{ss'}$ for all $s \in Y, s' \in D$ ($s, s' \in D$), and Σ_{DY} is the transpose of Σ_{YD}. The posterior mean vector $\mu_{Y|D}$ (1) is used to predict the measurements at any set Y of unobserved road segments. The posterior covariance matrix $\Sigma_{YY|D}$ (2), which is independent of the measurements z_D, can be processed in two ways to quantify the uncertainty of these predictions: (a) the trace of $\Sigma_{YY|D}$ yields the sum of posterior variances $\Sigma_{ss|D}$ over all $s \in Y$; (b) the determinant of $\Sigma_{YY|D}$ is used in calculating the Gaussian posterior joint entropy

$$\mathbb{H}[Z_Y|Z_D] \triangleq \frac{1}{2}\log(2\pi e)^{|Y|}\left|\Sigma_{YY|D}\right| . \quad (3)$$

In contrast to the first measure of uncertainty that assumes conditional independence between measurements in the set Y of unobserved road segments, the entropy-based measure (3) accounts for their correlation, thereby not overestimating their uncertainty. Hence, we will focus on using the entropy-based measure of uncertainty in this paper.

2.1 Graph-Based Kernel

If the observations are noisy (i.e., by assuming additive independent identically distributed Gaussian noise with variance σ_n^2), then their prior covariance $\sigma_{ss'}$ can be expressed as $\sigma_{ss'} = k(s, s') + \sigma_n^2\delta_{ss'}$ where $\delta_{ss'}$ is a Kronecker delta that is 1 if $s = s'$ and 0 otherwise, and k is a kernel function measuring the pairwise "similarity" of road segments. For a traffic phenomenon (e.g., road speeds), the correlation of measurements between pairs of road segments depends not only on their features (e.g., length, number of lanes, speed limit, direction) but also the road network topology. So, the kernel function is defined to exploit both the features and topology information, which will be described next.

Definition 1 (Road Network) *Let the road network be represented as a weighted directed graph $G \triangleq (V, E, m)$ that consists of*

- *a set V of vertices denoting the domain of all possible road segments,*
- *a set $E \subseteq V \times V$ of edges where there is an edge (s, s') from $s \in V$ to $s' \in V$ iff the end of segment s connects to the start of segment s' in the road network, and*
- *a weight function $m : E \to \mathbb{R}^+$ measuring the standardized Manhattan distance (Borg and Groenen, 2005) $m((s, s')) \triangleq \sum_{i=1}^{p} |[s]_i - [s']_i|/r_i$ of each edge (s, s') where $[s]_i$ ($[s']_i$) is the i-th component of the feature vector specifying road segment s (s'), and r_i is the range of the i-th feature. The weight function m serves as a dissimilarity measure between adjacent road segments.*

The next step is to compute the shortest path distance $d(s, s')$ between all pairs of road segments $s, s' \in V$ (i.e., using Floyd-Warshall or Johnson's algorithm) with respect to the topology of the weighted directed graph G. Such a distance function is again a measure of dissimilarity, rather than one of similarity, as required by a kernel function. Fur-

thermore, a valid GP kernel needs to be positive semidefinite and symmetric (Schölkopf and Smola, 2002), which are clearly violated by d.

To construct a valid GP kernel from d, multi-dimensional scaling (Borg and Groenen, 2005) is applied to embed the domain of road segments into the p'-dimensional Euclidean space $\mathbb{R}^{p'}$. Specifically, a mapping $g : V \rightarrow \mathbb{R}^{p'}$ is determined by minimizing the squared loss $g^* = \arg\min_g \sum_{s,s' \in V} (d(s,s') - \|g(s) - g(s')\|)^2$. With a small squared loss, the Euclidean distance $\|g^*(s) - g^*(s')\|$ between $g^*(s)$ and $g^*(s')$ is expected to closely approximate the shortest path distance $d(s,s')$ between any pair of road segments s and s'. After embedding into Euclidean space, a conventional kernel function such as the squared exponential one (Rasmussen and Williams, 2006) can be used:

$$k(s,s') = \sigma_s^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{p'}\left(\frac{[g^*(s)]_i - [g^*(s')]_i}{\ell_i}\right)^2\right)$$

where $[g^*(s)]_i$ ($[g^*(s')]_i$) is the i-th component of the p'-dimensional vector $g^*(s)$ ($g^*(s')$), and the hyperparameters $\sigma_s, \ell_1, \ldots, \ell_{p'}$ are, respectively, signal variance and length-scales that can be learned using maximum likelihood estimation (Rasmussen and Williams, 2006). The resulting kernel function k[1] is guaranteed to be valid.

2.2 Subset of Data Approximation

Although the GP is an effective predictive model, it faces a practical limitation of cubic time complexity in the number $|D|$ of observations; this can be observed from computing the posterior distribution (i.e., (1) and (2)), which requires inverting covariance matrix Σ_{DD} and incurs $\mathcal{O}(|D|^3)$ time. If $|D|$ is expected to be large, GP prediction cannot be performed in real time. For practical usage, we have to resort to computationally cheaper approximate GP prediction.

A simple method of approximation is to select only a subset U of the entire set D of observed road segments (i.e., $U \subset D$) to compute the posterior distribution of the measurements at any set $Y \subseteq V \setminus D$ of unobserved road segments. Such a *subset of data* (SoD) approximation method produces the following predictive Gaussian distribution, which closely resembles that of the full GP model (i.e., by simply replacing D in (1) and (2) with U):

$$\mu_{Y|U} = \mu_Y + \Sigma_{YU}\Sigma_{UU}^{-1}(z_U - \mu_U) \quad (4)$$

$$\Sigma_{YY|U} = \Sigma_{YY} - \Sigma_{YU}\Sigma_{UU}^{-1}\Sigma_{UY} \; . \quad (5)$$

Notice that the covariance matrix Σ_{UU} to be inverted only incurs $\mathcal{O}(|U|^3)$ time, which is independent of $|D|$.

The predictive performance of SoD approximation is sensitive to the selection of subset U. In practice, random subset selection often yields poor performance. This issue can be resolved by actively selecting an informative subset U in an iterative greedy manner: Firstly, U is initialized to be an empty set. Then, all road segments in $D \setminus U$ are scored based on a criterion that can be chosen from, for example, the works of (Krause *et al.*, 2008b; Lawrence *et al.*, 2003; Seeger and Williams, 2003). The highest-scored segment is selected for inclusion in U and removed from D. This greedy selection procedure is iterated until U reaches a predefined size. Among the various criteria introduced earlier, the differential entropy score (Lawrence *et al.*, 2003) is reported to perform well (Oh *et al.*, 2010); it is a monotonic function of the posterior variance $\Sigma_{ss|U}$ (5), thus resulting in the greedy selection of a segment $s \in D \setminus U$ with the largest variance in each iteration.

[1]For spatiotemporal traffic modeling, the kernel function k can be extended to account for the temporal dimension.

3 Decentralized Data Fusion

In the previous section, two centralized data fusion approaches to exact (i.e., (1) and (2)) and approximate (i.e., (4) and (5)) GP prediction are introduced. In this section, we will discuss the decentralized data fusion component of our D^{2}FAS algorithm, which distributes the computational load among the mobile sensors to achieve efficient and scalable approximate GP prediction.

The intuition of our decentralized data fusion algorithm is as follows: Each of the K mobile sensors constructs a local summary of the observations taken along its own path in the road network and communicates its local summary to every other sensor. Then, it assimilates the local summaries received from the other sensors into a globally consistent summary, which is exploited for predicting the traffic phenomenon as well as active sensing. This intuition will be formally realized and described in the paragraphs below.

While exploring the road network, each mobile sensor summarizes its local observations taken along its path based on a common support set $U \subset V$ known to all the other sensors. Its local summary is defined as follows:

Definition 2 (Local Summary) *Given a common support set $U \subset V$ known to all K mobile sensors, a set $D_k \subset V$ of observed road segments and a column vector z_{D_k} of corresponding measurements local to mobile sensor k, its local summary is defined as a tuple $(\dot{z}_U^k, \dot{\Sigma}_{UU}^k)$ where*

$$\dot{z}_U^k \triangleq \Sigma_{UD_k}\Sigma_{D_kD_k|U}^{-1}(z_{D_k} - \mu_{D_k}) \quad (6)$$

$$\dot{\Sigma}_{UU}^k \triangleq \Sigma_{UD_k}\Sigma_{D_kD_k|U}^{-1}\Sigma_{D_kU} \quad (7)$$

such that $\Sigma_{D_kD_k|U}$ is defined in a similar manner to (5).

Remark. Unlike SoD (Section 2.2), the support set U of road segments does not have to be observed, since the local summary (i.e., (6) and (7)) is independent of the corresponding measurements z_U. So, U does not need to be a subset of $D = \bigcup_{k=1}^K D_k$. To select an informative support set U from the set V of all possible segments in the road network, an offline active selection procedure similar to that in the last paragraph of Section 2.2 can be performed just once prior to observing data to determine U. In contrast, SoD has to perform online active selection every time new road segments are being observed.

By communicating its local summary to every other sensor, each mobile sensor can then construct a globally consistent summary from the received local summaries:

Definition 3 (Global Summary) *Given a common support set $U \subset V$ known to all K mobile sensors and the local summary $(\dot{z}_U^k, \dot{\Sigma}_{UU}^k)$ of every mobile sensor $k = 1, \ldots, K$, the global summary is defined as a tuple $(\ddot{z}_U, \ddot{\Sigma}_{UU})$ where*

$$\ddot{z}_U \triangleq \sum_{k=1}^{K} \dot{z}_U^k \tag{8}$$

$$\ddot{\Sigma}_{UU} \triangleq \Sigma_{UU} + \sum_{k=1}^{K} \dot{\Sigma}_{UU}^k \ . \tag{9}$$

Remark. In this paper, we assume all-to-all communication between the K mobile sensors. Supposing this is not possible and each sensor can only communicate locally with its neighbors, the summation structure of the global summary (specifically, (8) and (9)) makes it amenable to be constructed using distributed consensus filters (Olfati-Saber, 2005). We omit these details since they are beyond the scope of this paper.

Finally, the global summary is exploited by each mobile sensor to compute a globally consistent predictive Gaussian distribution, as detailed in Theorem 1A below, as well as to perform decentralized active sensing (Section 4):

Theorem 1 *Let a common support set $U \subset V$ be known to all K mobile sensors.*

A. *Given the global summary $(\ddot{z}_U, \ddot{\Sigma}_{UU})$, each mobile sensor computes a globally consistent predictive Gaussian distribution $\mathcal{N}(\overline{\mu}_Y, \overline{\Sigma}_{YY})$ of the measurements at any set Y of unobserved road segments where*

$$\overline{\mu}_Y \triangleq \mu_Y + \Sigma_{YU} \ddot{\Sigma}_{UU}^{-1} \ddot{z}_U \tag{10}$$

$$\overline{\Sigma}_{YY} \triangleq \Sigma_{YY} - \Sigma_{YU}(\Sigma_{UU}^{-1} - \ddot{\Sigma}_{UU}^{-1})\Sigma_{UY} \ . \tag{11}$$

B. *Let $\mathcal{N}(\mu_{Y|D}^{\text{PITC}}, \Sigma_{YY|D}^{\text{PITC}})$ be the predictive Gaussian distribution computed by the centralized sparse partially independent training conditional (PITC) approximation of GP model (Quiñonero-Candela and Rasmussen, 2005) where*

$$\mu_{Y|D}^{\text{PITC}} \triangleq \mu_Y + \Gamma_{YD}\left(\Gamma_{DD} + \Lambda\right)^{-1}(z_D - \mu_D) \tag{12}$$

$$\Sigma_{YY|D}^{\text{PITC}} \triangleq \Sigma_{YY} - \Gamma_{YD}\left(\Gamma_{DD} + \Lambda\right)^{-1}\Gamma_{DY} \tag{13}$$

such that

$$\Gamma_{BB'} \triangleq \Sigma_{BU}\Sigma_{UU}^{-1}\Sigma_{UB'} \tag{14}$$

and Λ is a block-diagonal matrix constructed from the K diagonal blocks of $\Sigma_{DD|U}$, each of which is a matrix $\Sigma_{D_k D_k|U}$ for $k = 1, \ldots, K$ where $D = \bigcup_{k=1}^{K} D_k$. Then, $\overline{\mu}_Y = \mu_{Y|D}^{\text{PITC}}$ and $\overline{\Sigma}_{YY} = \Sigma_{YY|D}^{\text{PITC}}$.

Its proof is given in (Chen *et al.*, 2012). The equivalence result of Theorem 1B bears two implications:

Remark 1. The computation of PITC can be parallelized and distributed among the mobile sensors in a Google-like MapReduce paradigm (Chu *et al.*, 2007), thereby improving the time efficiency of prediction: Each of the K mappers (sensors) is tasked to compute its local summary while the reducer (any sensor) sums these local summaries into a global summary, which is then used to compute the predictive Gaussian distribution. Supposing $|Y| \leq |U|$ for simplicity, the $\mathcal{O}\big(|D|((|D|/K)^2 + |U|^2)\big)$ time incurred by PITC can be reduced to $\mathcal{O}\big((|D|/K)^3 + |U|^3 + |U|^2 K\big)$ time of running our decentralized algorithm on each of the K sensors, the latter of which scales better with increasing number $|D|$ of observations.

Remark 2. We can draw insights from PITC to elucidate an underlying property of our decentralized algorithm: It is assumed that $Z_{D_1}, \ldots, Z_{D_K}, Z_Y$ are conditionally independent given the measurements at the support set U of road segments. To potentially reduce the degree of violation of this assumption, an informative support set U is actively selected, as described earlier in this section. Furthermore, the experimental results on real-world urban road network data[2] (Section 6) show that D^{2}FAS can achieve predictive performance comparable to that of the full GP model while enjoying significantly lower computational cost, thus demonstrating the practicality of such an assumption for predicting traffic phenomena. The predictive performance of D^{2}FAS can be improved by increasing the size of U at the expense of greater time and communication overhead.

4 Decentralized Active Sensing

The problem of active sensing with K mobile sensors is formulated as follows: Given the set $D_k \subset V$ of observed road segments and the currently traversed road segment $s_k \in V$ of every mobile sensor $k = 1, \ldots, K$, the mobile sensors have to coordinate to select the most informative walks $w_1^*, \ldots, w_K^*$ of length (i.e., number of road segments) L each and with respective origins $s_1, \ldots, s_K$ in the road network G:

$$(w_1^*, \ldots, w_K^*) = \underset{(w_1, \ldots, w_K)}{\arg\max}\ \mathbb{H}\Big[Z_{\bigcup_{k=1}^{K} Y_{w_k}} \Big| Z_{\bigcup_{k=1}^{K} D_k}\Big] \tag{15}$$

where Y_{w_k} denotes the set of unobserved road segments induced by the walk w_k. To simplify notation, let a joint walk be denoted by $w \triangleq (w_1, \ldots, w_K)$ (similarly, for w^*) and its induced set of unobserved road segments be $Y_w \triangleq \bigcup_{k=1}^{K} Y_{w_k}$ from now on. Interestingly, it can be shown using the chain rule for entropy that these maximum-entropy walks w^* minimize the posterior joint entropy (i.e., $\mathbb{H}[Z_{V \setminus (D \bigcup Y_{w^*})} | Z_{D \bigcup Y_{w^*}}]$) of the measurements at the remaining unobserved segments (i.e., $V \setminus (D \bigcup Y_{w^*})$) in the road network. After executing the walk w_k^*, each mobile sensor k observes the set $Y_{w_k^*}$ of road segments and updates its local information:

$$D_k \leftarrow D_k \bigcup Y_{w_k^*}, z_{D_k} \leftarrow z_{D_k \bigcup Y_{w_k^*}}, s_k \leftarrow \text{terminus of } w_k^* \ . \tag{16}$$

[2] Quiñonero-Candela and Rasmussen (2005) only illustrated the predictive performance of PITC on a simulated toy example.

Evaluating the Gaussian posterior entropy term in (15) involves computing a posterior covariance matrix (3) using one of the data fusion methods described earlier: If (2) of full GP model (Section 2) or (5) of SoD (Section 2.2) is to be used, then the observations that are gathered distributedly by the sensors have to be fully communicated to a central data fusion center. In contrast, our decentralized data fusion algorithm (Section 3) only requires communicating local summaries (Definition 2) to compute (11) for solving the active sensing problem (15):

$$w^* = \arg\max_{w} \overline{\mathbb{H}}[Z_{Y_w}] \ , \tag{17}$$

$$\overline{\mathbb{H}}[Z_{Y_w}] \triangleq \frac{1}{2}\log(2\pi e)^{|Y_w|}\left|\overline{\Sigma}_{Y_w Y_w}\right| \ . \tag{18}$$

Without imposing any structural assumption, solving the active sensing problem (17) will be prohibitively expensive due to the space of possible joint walks w that grows exponentially in the number K of mobile sensors. To overcome this scalability issue for D^{2}FAS, our key idea is to construct a block-diagonal matrix whose log-determinant closely approximates that of $\overline{\Sigma}_{Y_w Y_w}$ (11) and exploit the property that the log-determinant of such a block-diagonal matrix can be decomposed into a sum of log-determinants of its diagonal blocks, each of which depends only on the walks of a disjoint subset of the K mobile sensors. Consequently, the active sensing problem can be partially decentralized, leading to a reduced space of possible joint walks to be searched, as detailed in the rest of this section.

Firstly, we extend an earlier assumption in Section 3: $Z_{D_1},\ldots,Z_{D_K},Z_{Y_{w_1}},\ldots,Z_{Y_{w_K}}$ are conditionally independent given the measurements at the support set U of road segments. Then, it can be shown via the equivalence to PITC (Theorem 1B) that $\overline{\Sigma}_{Y_w Y_w}$ (11) comprises diagonal blocks of the form $\overline{\Sigma}_{Y_{w_k} Y_{w_k}}$ for $k=1,\ldots,K$ and off-diagonal blocks of the form $\Sigma_{Y_{w_k}U}\ddot{\Sigma}^{-1}_{UU}\Sigma_{UY_{w_{k'}}}$ for $k,k'=1,\ldots,K$ and $k\neq k'$. In particular, each off-diagonal block of $\overline{\Sigma}_{Y_w Y_w}$ represents the correlation of measurements between the unobserved road segments Y_{w_k} and $Y_{w_{k'}}$ along the respective walks w_k of sensor k and $w_{k'}$ of sensor k'. If the correlation between some pair of their possible walks is high enough, then their walks have to be coordinated. This is formally realized by the following coordination graph over the K sensors:

Definition 4 (Coordination Graph) *Define the coordination graph to be an undirected graph $\mathcal{G}\triangleq(\mathcal{V},\mathcal{E})$ that comprises*

- *a set $\mathcal{V}$ of vertices denoting the K mobile sensors, and*
- *a set $\mathcal{E}$ of edges denoting coordination dependencies between sensors such that there exists an edge $\{k,k'\}$ incident with sensors $k\in\mathcal{V}$ and $k'\in\mathcal{V}\setminus\{k\}$ iff*

$$\max_{s\in Y_{W_k},s'\in Y_{W_{k'}}}\left|\Sigma_{sU}\ddot{\Sigma}^{-1}_{UU}\Sigma_{Us'}\right|>\varepsilon \tag{19}$$

for a predefined constant $\varepsilon>0$ where W_k denotes the set of possible walks of length L of mobile sensor k from origin s_k in the road network G and $Y_{W_k}\triangleq\bigcup_{w_k\in W_k}Y_{w_k}$.

Remark. The construction of $\mathcal{G}$ can be decentralized as follows: Since $\ddot{\Sigma}_{UU}$ is symmetric and positive definite, it can be decomposed by Cholesky factorization into $\ddot{\Sigma}_{UU}=\Psi\Psi^\top$ where Ψ is a lower triangular matrix and $\Psi^\top$ is the transpose of Ψ. Then, $\Sigma_{sU}\ddot{\Sigma}^{-1}_{UU}\Sigma_{Us'}=(\Psi\backslash\Sigma_{Us})^\top\Psi\backslash\Sigma_{Us'}$ where $\Psi\backslash B$ denotes the column vector ϕ solving $\Psi\phi=B$. That is, $\Sigma_{sU}\ddot{\Sigma}^{-1}_{UU}\Sigma_{Us'}$ (19) can be expressed as a dot product of two vectors $\Psi\backslash\Sigma_{Us}$ and $\Psi\backslash\Sigma_{Us'}$; this property is exploited to determine adjacency between sensors in a decentralized manner:

Definition 5 (Adjacency) *Let*

$$\Phi_k\triangleq\{\Psi\backslash\Sigma_{Us}\}_{s\in Y_{W_k}} \tag{20}$$

for $k=1,\ldots,K$. A sensor $k\in\mathcal{V}$ is adjacent to sensor $k'\in\mathcal{V}\setminus\{k\}$ in coordination graph $\mathcal{G}$ iff

$$\max_{\phi\in\Phi_k,\phi'\in\Phi_{k'}}\left|\phi^\top\phi'\right|>\varepsilon \ . \tag{21}$$

It follows from the above definition that if each sensor k constructs Φ_k and exchanges it with every other sensor, then it can determine its adjacency to all the other sensors and store this information in a column vector a_k of length K with its k'-th component being defined as follows:

$$[a_k]_{k'}=\begin{cases}1 & \text{if sensor } k \text{ is adjacent to sensor } k',\\ 0 & \text{otherwise.}\end{cases} \tag{22}$$

By exchanging its adjacency vector a_k with every other sensor, each sensor can construct a globally consistent adjacency matrix $A_{\mathcal{G}}\triangleq(a_1\ldots a_K)$ to represent coordination graph $\mathcal{G}$.

Next, by computing the connected components (say, $\mathcal{K}$ of them) of coordination graph $\mathcal{G}$, their resulting vertex sets partition the set $\mathcal{V}$ of K sensors into $\mathcal{K}$ disjoint subsets $\mathcal{V}_1,\ldots,\mathcal{V}_{\mathcal{K}}$ such that the sensors within each subset have to coordinate their walks. Each sensor can determine its residing connected component in a decentralized way by performing a depth-first search in $\mathcal{G}$ starting from it as root.

Finally, construct a block-diagonal matrix $\widehat{\Sigma}_{Y_w Y_w}$ to comprise diagonal blocks of the form $\overline{\Sigma}_{Y_{w_{\mathcal{V}_n}}Y_{w_{\mathcal{V}_n}}}$ for $n=1,\ldots,\mathcal{K}$ where $w_{\mathcal{V}_n}\triangleq(w_k)_{k\in\mathcal{V}_n}$ and $Y_{w_{\mathcal{V}_n}}\triangleq\bigcup_{k\in\mathcal{V}_n}Y_{w_k}$. The active sensing problem (17) is then approximated by

$$\begin{aligned}&\max_{w}\ \frac{1}{2}\log(2\pi e)^{|Y_w|}\left|\widehat{\Sigma}_{Y_w Y_w}\right|\\ &\equiv\max_{(w_{\mathcal{V}_1},\ldots,w_{\mathcal{V}_{\mathcal{K}}})}\sum_{n=1}^{\mathcal{K}}\log(2\pi e)^{|Y_{w_{\mathcal{V}_n}}|}\left|\overline{\Sigma}_{Y_{w_{\mathcal{V}_n}}Y_{w_{\mathcal{V}_n}}}\right| \qquad (23)\\ &=\sum_{n=1}^{\mathcal{K}}\max_{w_{\mathcal{V}_n}}\log(2\pi e)^{|Y_{w_{\mathcal{V}_n}}|}\left|\overline{\Sigma}_{Y_{w_{\mathcal{V}_n}}Y_{w_{\mathcal{V}_n}}}\right|,\end{aligned}$$

which can be solved in a partially decentralized manner by each disjoint subset $\mathcal{V}_n$ of mobile sensors:

$$\widehat{w}_{\mathcal{V}_n}=\arg\max_{w_{\mathcal{V}_n}}\log(2\pi e)^{|Y_{w_{\mathcal{V}_n}}|}\left|\overline{\Sigma}_{Y_{w_{\mathcal{V}_n}}Y_{w_{\mathcal{V}_n}}}\right|. \tag{24}$$

Our active sensing algorithm becomes fully decentralized if ε is set to be sufficiently large: more sensors become isolated in $\mathcal{G}$, consequently decreasing the size $\kappa\triangleq\max_n|\mathcal{V}_n|$

of its largest connected component to 1. As shown in Section 5.1, decreasing κ improves its time efficiency. On the other hand, it tends to a centralized behavior (17) by setting $\varepsilon \to 0^+$: $\mathcal{G}$ becomes near-complete, thus resulting in $\kappa \to K$.

Let
$$\xi \triangleq \max_{n, w_{\mathcal{V}_n}, i, i'} \left| \left[\left(\overline{\Sigma}_{Y_{w_{\mathcal{V}_n}} Y_{w_{\mathcal{V}_n}}} \right)^{-1} \right]_{ii'} \right| \quad (25)$$

and $\epsilon \triangleq 0.5 \log 1 \Big/ \left(1 - \left(K^{1.5} L^{2.5} \kappa \xi \varepsilon\right)^2\right)$. In the result below, we prove that the joint walk $\widehat{w} \triangleq (\widehat{w}_{\mathcal{V}_1}, \ldots, \widehat{w}_{\mathcal{V}_\mathcal{K}})$ is guaranteed to achieve an entropy $\overline{\mathbb{H}}[Z_{Y_{\widehat{w}}}]$ (i.e., by plugging $\widehat{w}$ into (18)) that is not more than ϵ from the maximum entropy $\overline{\mathbb{H}}[Z_{Y_{w^*}}]$ achieved by joint walk w^* (17):

Theorem 2 (Performance Guarantee) *If* $K^{1.5} L^{2.5} \kappa \xi \varepsilon < 1$, *then* $\overline{\mathbb{H}}[Z_{Y_{w^*}}] - \overline{\mathbb{H}}[Z_{Y_{\widehat{w}}}] \leq \epsilon$.

Its proof is given in (Chen *et al.*, 2012). The implication of Theorem 2 is that our partially decentralized active sensing algorithm can perform comparatively well (i.e., small ϵ) under the following favorable environmental conditions: (a) the network of K sensors is not large, (b) length L of each sensor's walk to be optimized is not long, (c) the largest subset of κ sensors being formed to coordinate their walks (i.e., largest connected component in $\mathcal{G}$) is reasonably small, and (d) the minimum required correlation ε between walks of adjacent sensors is kept low.

Algorithm 1 below outlines the key operations of our D^{2}FAS algorithm to be run on each mobile sensor k, as detailed previously in Sections 3 and 4:

Algorithm 1: D^{2}FAS$(U, K, L, k, D_k, z_{D_k}, s_k)$

while *true* **do**
- /* Data fusion (Section 3) */
- Construct local summary by (6) & (7)
- Exchange local summary with every sensor $i \neq k$
- Construct global summary by (8) & (9)
- Predict measurements at unobserved road segments by (10) & (11)
- /* Active Sensing (Section 4) */
- Construct Φ_k by (20)
- Exchange Φ_k with every sensor $i \neq k$
- Compute adjacency vector a_k by (21) & (22)
- Exchange adjacency vector with every sensor $i \neq k$
- Construct adjacency matrix of coordination graph
- Find vertex set $\mathcal{V}_n$ of its residing connected component
- Compute maximum-entropy joint walk $\widehat{w}_{\mathcal{V}_n}$ by (24)
- Execute walk $\widehat{w}_k$ and observe its road segments $Y_{\widehat{w}_k}$
- Update local information D_k, z_{D_k}, and s_k by (16)

5 Time and Communication Overheads

In this section, the time and communication overheads of our D^{2}FAS algorithm are analyzed and compared to that of centralized active sensing (17) coupled with the data fusion methods: Full GP (FGP) and SoD (Section 2).

5.1 Time Complexity

The data fusion component of D^{2}FAS involves computing the local and global summaries and the predictive Gaussian distribution. To construct the local summary using (6) and (7), each sensor has to evaluate $\Sigma_{D_k D_k|U}$ in $\mathcal{O}\big(|U|^3 + |U|(|D|/K)^2\big)$ time and invert it in $\mathcal{O}\big((|D|/K)^3\big)$ time, after which the local summary is obtained in $\mathcal{O}\big(|U|^2|D|/K + |U|(|D|/K)^2\big)$ time. The global summary is computed in $\mathcal{O}\big(|U|^2 K\big)$ by (8) and (9). Finally, the predictive Gaussian distribution is derived in $\mathcal{O}\big(|U|^3 + |U||Y|^2\big)$ time using (10) and (11). Supposing $|Y| \leq |U|$ for simplicity, the time complexity of data fusion is then $\mathcal{O}\big((|D|/K)^3 + |U|^3 + |U|^2 K\big)$.

Let the maximum out-degree of G be denoted by δ. Then, each sensor has to consider $\Delta \triangleq \delta^L$ possible walks of length L. The active sensing component of D^{2}FAS involves computing Φ_k in $\mathcal{O}\big(\Delta L |U|^2\big)$ time, a_k in $\mathcal{O}\big(\Delta^2 L^2 |U| K\big)$ time, its residing connected component in $\mathcal{O}\big(\kappa^2\big)$ time, and the maximum-entropy joint walk by (11) and (24) with the following incurred time: The largest connected component of κ sensors in $\mathcal{G}$ has to consider Δ^κ possible joint walks. Note that $\overline{\Sigma}_{Y_{w_{\mathcal{V}_n}} Y_{w_{\mathcal{V}_n}}} = \mathrm{diag}\Big((\Sigma_{Y_{w_k} Y_{w_k}|U})_{k \in \mathcal{V}_n}\Big) + \Sigma_{Y_{w_{\mathcal{V}_n}} U} \ddot{\Sigma}_{UU}^{-1} \Sigma_{U Y_{w_{\mathcal{V}_n}}}$ where $\mathrm{diag}(B)$ constructs a diagonal matrix by placing vector B on its diagonal. By exploiting Φ_k, the diagonal and latter matrix terms for all possible joint walks can be computed in $\mathcal{O}\big(\kappa\Delta(L|U|^2 + L^2|U|)\big)$ and $\mathcal{O}\big(\kappa^2\Delta^2 L^2 |U|\big)$ time, respectively. For each joint walk $w_{\mathcal{V}_n}$, evaluating the determinant of $\overline{\Sigma}_{Y_{w_{\mathcal{V}_n}} Y_{w_{\mathcal{V}_n}}}$ incurs $\mathcal{O}\big((\kappa L)^3\big)$ time. Therefore, the time complexity of active sensing is $\mathcal{O}\big(\kappa\Delta L|U|^2 + \Delta^2 L^2 |U|(K + \kappa^2) + \Delta^\kappa (\kappa L)^3\big)$.

Hence, the time complexity of our D^{2}FAS algorithm is $\mathcal{O}((|D|/K)^3 + |U|^2(|U| + K + \kappa\Delta L) + \Delta^2 L^2 |U|(K + \kappa^2) + \Delta^\kappa (\kappa L)^3)$. In contrast, the time incurred by centralized active sensing coupled with FGP and SoD are, respectively, $\mathcal{O}\big(|D|^3 + \Delta^K KL(|D|^2 + (KL)^2)\big)$ and $\mathcal{O}\big(|U|^3|D| + \Delta^K KL(|U|^2 + (KL)^2)\big)$. It can be observed that D^{2}FAS can scale better with large $|D|$ (i.e., number of observations) and K (i.e., number of sensors). The scalability of D^{2}FAS vs. FGP and SoD will be further evaluated empirically in Section 6.

5.2 Communication Complexity

Let the communication overhead be defined as the size of each broadcast message. Recall from the data fusion component of D^{2}FAS in Algorithm 1 that, in each iteration, each sensor broadcasts a $\mathcal{O}\big(|U|^2\big)$-sized summary encapsulating its local observations, which is robust against communication failure. In contrast, FGP and SoD require each sensor to broadcast, in each iteration, a $\mathcal{O}(|D|/K)$-sized message comprising exactly its local observations to handle communication failure. If the number of local observations grows to be larger in size than a local summary of predefined size, then the data fusion component of D^{2}FAS is more scalable than FGP and SoD in terms of communication overhead. For the partially decentralized active sensing component of D^{2}FAS, each sensor broadcasts $\mathcal{O}(\Delta L|U|)$-sized Φ_k and $\mathcal{O}(K)$-sized a_k messages.

6 Experiments and Discussion

This section evaluates the predictive performance, time efficiency, and scalability of our D^{2}FAS algorithm on a real-world traffic phenomenon (i.e., speeds (km/h) of road segments) over an urban road network (top figure) in Tampines area, Singapore during evening peak hours on April 20, 2011. It comprises 775 road segments including highways, arterials, slip roads, etc. The mean speed is 48.8 km/h and the standard deviation is 20.5 km/h.

The performance of D^{2}FAS is compared to that of centralized active sensing (17) coupled with the state-of-the-art data fusion methods: full GP (FGP) and SoD (Section 2). A network of K mobile sensors is tasked to explore the road network to gather a total of up to 960 observations. To reduce computational time, each sensor repeatedly computes and executes maximum-entropy walks of length $L = 2$ (instead of computing a very long walk), unless otherwise stated. For D^{2}FAS and SoD, $|U|$ is set to 64 . For the active sensing component of D^{2}FAS, ε is set to 0.1, unless otherwise stated. The experiments are run on a Linux PC with Intel® Core™2 Quad CPU Q9550 at 2.83 GHz.

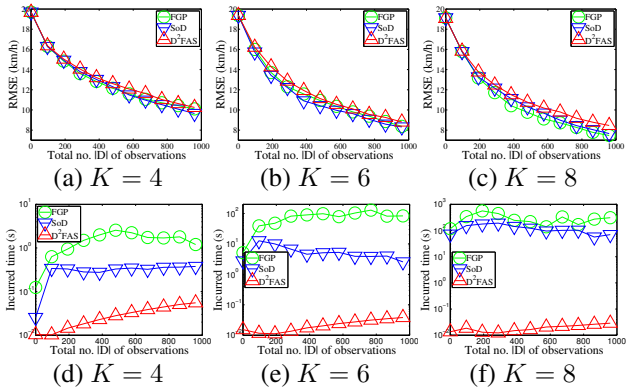
6.1 Performance Metrics

The first metric evaluates the predictive performance of a tested algorithm: It measures the *root mean squared error* (RMSE) $\sqrt{|V|^{-1}\sum_{s\in V}\left(z_s-\widehat{\mu}_s\right)^2}$ over the entire domain V of the road network that is incurred by the predictive mean $\widehat{\mu}_s$ of the tested algorithm, specifically, using (1) of FGP, (4) of SoD, or (10) of D^{2}FAS. The second metric evaluates the time efficiency and scalability of a tested algorithm by measuring its incurred time; for D^{2}FAS, the maximum of the time incurred by all subsets $\mathcal{V}_1,\ldots,\mathcal{V}_K$ of sensors is recorded.

6.2 Results and Analysis

Predictive performance and time efficiency. Fig. 1 shows results of the performance of the tested algorithms averaged over 40 randomly generated starting sensor locations with varying number $K = 4, 6, 8$ of sensors. It can be observed that D^{2}FAS is significantly more time-efficient and scales better with increasing number $|D|$ of observations (Figs. 1d to 1f) while achieving predictive performance close to that of centralized active sensing coupled with FGP and SoD (Figs. 1a to 1c). Specifically, D^{2}FAS is about $1, 2, 4$ orders of magnitude faster than centralized active sensing coupled with FGP and SoD for $K = 4, 6, 8$ sensors, respectively.

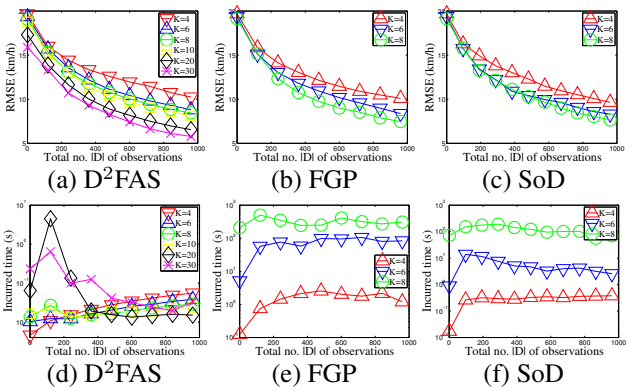
Scalability of D^{2}FAS. Using the same results as that in Fig. 1, Fig. 2 plots them differently to reveal the scalability of the tested algorithms with increasing number K of sensors. Additionally, we provide results of the performance of D^{2}FAS for $K = 10, 20, 30$ sensors; such results are not available for centralized active sensing coupled with FGP and SoD due to extremely long incurred time. It can be observed from Figs. 2a to 2c that the predictive performance of all tested algorithms improve with a larger number of sensors because each sensor needs to execute fewer walks and its performance is therefore less adversely affected by its myopic selection (i.e., $L = 2$) of maximum-entropy walks. As a result, more informative unobserved road segments are explored.



(a) $K = 4$ (b) $K = 6$ (c) $K = 8$

(d) $K = 4$ (e) $K = 6$ (f) $K = 8$

Figure 1: Graphs of (a-c) predictive performance and (d-f) time efficiency vs. total no. $|D|$ of observations gathered by varying number K of mobile sensors.

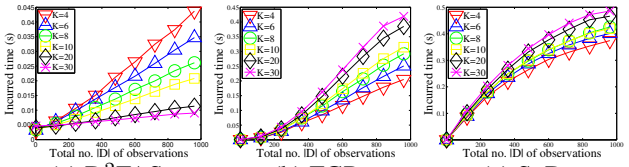


(a) D^{2}FAS (b) FGP (c) SoD

(d) D^{2}FAS (e) FGP (f) SoD

Figure 2: Graphs of (a-c) predictive performance and (d-f) time efficiency vs. total no. $|D|$ of observations gathered by varying number K of mobile sensors.

As shown in Fig. 2d, when the randomly placed sensors gather their initial observations (i.e., $|D| < 400$), the time incurred by D^{2}FAS is higher for greater K due to larger subsets of sensors being formed to coordinate their walks (i.e., larger κ). As more observations are gathered (i.e., $|D| \geq 400$), its partially decentralized active sensing component directs the sensors to explore further apart from each other in order to maximize the entropy of their walks. This consequently decreases κ, leading to a reduction in incurred time. Furthermore, as K increases from 4 to 20, the incurred time decreases due to its decentralized data fusion component that can distribute the computational load



(a) D^{2}FAS (b) FGP (c) SoD

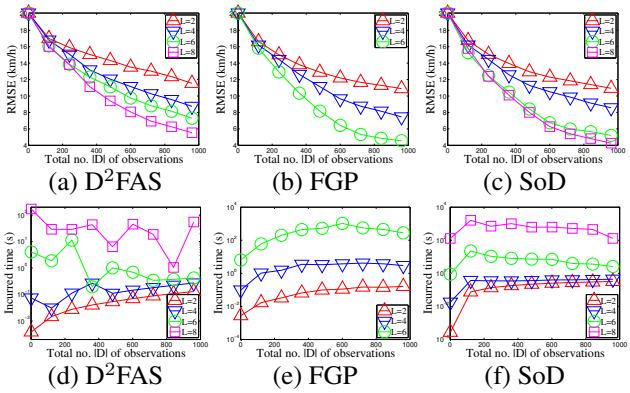
Figure 3: Graphs of time efficiency vs. total no. $|D|$ of observations gathered by varying number K of sensors.

among a greater number of sensors. When the road network becomes more crowded from $K = 20$ to $K = 30$ sensors, the incurred time increases slightly due to slightly larger κ. In contrast, Figs. 2e and 2f show that the time taken by FGP and SoD increases significantly primarily due to their centralized active sensing incurring exponential time in K. Hence, the scalability of our D^{2}FAS algorithm in the number of sensors allows the deployment of a larger-scale mobile sensor network (i.e., $K \geq 10$) to achieve more accurate traffic modeling and prediction (Figs. 2a to 2c).

Scalability of data fusion. Fig. 3 shows results of the scalability of the tested data fusion methods with increasing number K of sensors. In order to produce meaningful results for fair comparison, the same active sensing component has to be coupled with the data fusion methods and its incurred time kept to a minimum. As such, we impose the use of a fully decentralized active sensing component to be performed by each mobile sensor k: $w_k^* = \arg\max_{w_k} \mathbb{H}[Z_{Y_{w_k}}|Z_D]$. For D^{2}FAS, this corresponds exactly to (24) by setting a large enough ε (in our experiments, $\varepsilon = 2$) to yield $\kappa = 1$; consequently, computational and communicational operations pertaining to the coordination graph can be omitted.

It can be seen from Fig. 3a that the time incurred by the decentralized data fusion component of D^{2}FAS decreases with increasing K, as explained previously. In contrast, the time incurred by FGP and SoD increases (Fig. 3b and 3c): As discussed above, a larger number of sensors results in a greater quantity of more informative unique observations to be gathered (i.e., fewer repeated observations), which increases the time needed for data fusion. When $K \geq 10$, D^{2}FAS is at least 1 order of magnitude faster than FGP and SoD. It can also be observed that D^{2}FAS scales better with increasing number of observations. So, the real-time performance and scalability of D^{2}FAS's decentralized data fusion enable it to be used for persistent large-scale traffic modeling and prediction where a large number of observations and sensors (including static and passive ones) are expected to be available.

Varying length L of walk. Fig. 4 shows results of the performance of the tested algorithms with varying length $L = 2, 4, 6, 8$ of maximum-entropy joint walks; we choose to experiment with just 2 sensors since Figs. 2 and 3 reveal that a smaller number of sensors produce poorer predictive performance and higher incurred time with large number of



(a) D^{2}FAS (b) FGP (c) SoD

(d) D^{2}FAS (e) FGP (f) SoD

Figure 4: Graphs of (a-c) predictive performance and (d-f) time efficiency vs. total no. $|D|$ of observations gathered by 2 mobile sensors with varying length L of maximum-entropy joint walks.

observations for D^{2}FAS. It can be observed that the predictive performance of all tested algorithms improve with increasing walk length L because the selection of maximum-entropy joint walks is less myopic. The time incurred by D^{2}FAS increases due to larger κ but grows more slowly and is lower than that incurred by centralized active sensing coupled with FGP and SoD. Specifically, when $L = 8$, D^{2}FAS is at least 1 order of magnitude faster (i.e., average of 60 s) than centralized active sensing coupled with SoD (i.e., average of > 732 s) and FGP (i.e., not available due to excessive incurred time). Also, notice from Figs. 2a and 2d that if a large number of sensors (i.e., $K = 30$) is available, D^{2}FAS can select shorter walks of $L = 2$ to be significantly more time-efficient (i.e., average of > 3 orders of magnitude faster) while achieving predictive performance comparable to that of SoD with $L = 8$ and FGP with $L = 6$.

7 Conclusion

This paper describes a decentralized data fusion and active sensing algorithm for modeling and predicting spatiotemporal traffic phenomena with mobile sensors. Analytical and empirical results have shown that our D^{2}FAS algorithm is significantly more time-efficient and scales better with increasing number of observations and sensors while achieving predictive performance close to that of state-of-the-art centralized active sensing coupled with FGP and SoD. Hence, D^{2}FAS is practical for deployment in a large-scale mobile sensor network to achieve persistent and accurate traffic modeling and prediction. For our future work, we will assume that each sensor can only communicate locally with its neighbors (instead of assuming all-to-all communication) and develop a *distributed* data fusion approach to efficient and scalable approximate GP prediction based on D^{2}FAS and consensus filters (Olfati-Saber, 2005).

Acknowledgments. This work was supported by Singapore-MIT Alliance Research and Technology (SMART) Subaward Agreement 14 R-252-000-466-592.

References

Borg, I. and Groenen, P. J. F. (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer, NY.

Chen, H., Rakha, H. A., and Sadek, S. (2011). Real-time freeway traffic state prediction: A particle filter approach. In *Proc. IEEE ITSC*, pages 626–631.

Chen, J., Low, K. H., Tan, C. K.-Y., Oran, A., Jaillet, P., Dolan, J. M., and Sukhatme, G. S. (2012). Decentralized data fusion and active sensing with mobile sensors for modeling and predicting spatiotemporal traffic phenomena. arXiv:1206.6230.

Chu, C.-T., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G. R., Ng, A. Y., and Olukotun, K. (2007). Map-Reduce for machine learning on multicore. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 281–288, Cambridge, MA. MIT Press.

Chung, T. H., Gupta, V., Burdick, J. W., and Murray, R. M. (2004). On a decentralized active sensing strategy using mobile sensor platforms in a network. In *Proc. CDC*, pages 1914–1919.

Coates, M. (2004). Distributed particle filters for sensor networks. In *Proc. IPSN*, pages 99–107.

Cortes, J. (2009). Distributed kriged Kalman filter for spatial estimation. *IEEE Trans. Automat. Contr.*, **54**(12), 2816–2827.

Dolan, J. M., Podnar, G., Stancliff, S., Low, K. H., Elfes, A., Higinbotham, J., Hosler, J. C., Moisan, T. A., and Moisan, J. (2009). Cooperative aquatic sensing using the telesupervised adaptive ocean sensor fleet. In *Proc. SPIE Conference on Remote Sensing of the Ocean, Sea Ice, and Large Water Regions*, volume 7473.

Guestrin, C., Bodik, P., Thibaus, R., Paskin, M., and Madden, S. (2004). Distributed regression: an efficient framework for modeling sensor network data. In *Proc. IPSN*, pages 1–10.

Kamarianakis, Y. and Prastacos, P. (2003). Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches. *Transport. Res. Rec.*, **1857**, 74–84.

Krause, A., Horvitz, E., Kansal, A., and Zhao, F. (2008a). Toward community sensing. In *Proc. IPSN*, pages 481–492.

Krause, A., Singh, A., and Guestrin, C. (2008b). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, **9**, 235–284.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast sparse Gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616, Cambridge, MA. MIT Press.

Low, K. H., Gordon, G. J., Dolan, J. M., and Khosla, P. (2007). Adaptive sampling for multi-robot wide-area exploration. In *Proc. IEEE ICRA*, pages 755–760.

Low, K. H., Dolan, J. M., and Khosla, P. (2008). Adaptive multi-robot wide-area exploration and mapping. In *Proc. AAMAS*, pages 23–30.

Low, K. H., Dolan, J. M., and Khosla, P. (2009a). Information-theoretic approach to efficient adaptive path planning for mobile robotic environmental sensing. In *Proc. ICAPS*, pages 233–240.

Low, K. H., Podnar, G., Stancliff, S., Dolan, J. M., and Elfes, A. (2009b). Robot boats as a mobile aquatic sensor network. In *Proc. IPSN-09 Workshop on Sensor Networks for Earth and Space Science Applications*.

Low, K. H., Dolan, J. M., and Khosla, P. (2011). Active Markov information-theoretic path planning for robotic environmental sensing. In *Proc. AAMAS*, pages 753–760.

Low, K. H., Chen, J., Dolan, J. M., Chien, S., and Thompson, D. R. (2012). Decentralized active robotic exploration and mapping for probabilistic field classification in environmental sensing. In *Proc. AAMAS*, pages 105–112.

Min, W. and Wynter, L. (2011). Real-time road traffic prediction with spatio-temporal correlations. *Transport. Res. C-Emer.*, **19**(4), 606–616.

Neumann, M., Kersting, K., Xu, Z., and Schulz, D. (2009). Stacked Gaussian process learning. In *Proc. ICDM*, pages 387–396.

Oh, S., Xu, Y., and Choi, J. (2010). Explorative navigation of mobile sensor networks using sparse Gaussian processes. In *Proc. CDC*, pages 3851–3856.

Olfati-Saber, R. (2005). Distributed Kalman filter with embedded consensus filters. In *Proc. CDC*, pages 8179–8184.

Paskin, M. A. and Guestrin, C. (2004). Robust probabilistic inference in distributed systems. In *Proc. UAI*, pages 436–445.

Podnar, G., Dolan, J. M., Low, K. H., and Elfes, A. (2010). Telesupervised remote surface water quality sensing. In *Proc. IEEE Aerospace Conference*.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *JMLR*, **6**, 1939–1959.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.

Rosencrantz, M., Gordon, G., and Thrun, S. (2003). Decentralized sensor fusion with distributed particle filters. In *Proc. UAI*, pages 493–500.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, first edition.

Schrank, D., Lomax, T., and Eisele, B. (2011). *TTI's 2011 Urban Mobility Report*. Texas Transportation Institute, Texas A&M University.

Seeger, M. and Williams, C. (2003). Fast forward selection to speed up sparse Gaussian process regression. In C. M. Bishop and B. J. Frey, editors, *Proc. AISTATS*.

Srinivasan, K. K. and Jovanis, P. P. (1996). Determination of number of probe vehicle required for reliable travel time measurement in urban network. *Transport. Res. Rec.*, **1537**, 15–22.

Stranders, R., Farinelli, A., Rogers, A., and Jennings, N. R. (2009). Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proc. IJCAI*, pages 299–304.

Sukkarieh, S., Nettleton, E., Kim, J., Ridley, M., Goktogan, A., and Durrant-Whyte, H. (2003). The ANSER project: Data fusion across multiple uninhabited air vehicles. *IJRR*, **22**(7-8), 505–539.

Turner, S. M., Eisele, W. L., Benz, R. J., and Holdener, D. J. (1998). Travel time data collection handbook. Technical Report FHWA-PL-98-035, Federal Highway Administration, Office of Highway Information Management, Washington, DC.

Wang, Y. and Papageorgiou, M. (2005). Real-time freeway traffic state estimation based on extended Kalman filter: a general approach. *Transport. Res. B-Meth.*, **39**(2), 141–167.

Work, D. B., Blandin, S., Tossavainen, O., Piccoli, B., and Bayen, A. (2010). A traffic model for velocity data assimilation. *AMRX*, **2010**(1), 1–35.

Bayesian Structure Learning for Markov Random Fields with a Spike and Slab Prior

Yutian Chen
Department of Computer Science
University of California, Irvine
Irvine, CA 92697

Max Welling
Department of Computer Science
University of California, Irvine
Irvine, CA 92697

Abstract

In recent years a number of methods have been developed for automatically learning the (sparse) connectivity structure of Markov Random Fields. These methods are mostly based on L_1-regularized optimization which has a number of disadvantages such as the inability to assess model uncertainty and expensive cross-validation to find the optimal regularization parameter. Moreover, the model's predictive performance may degrade dramatically with a sub-optimal value of the regularization parameter (which is sometimes desirable to induce sparseness). We propose a fully Bayesian approach based on a "spike and slab" prior (similar to L_0 regularization) that does not suffer from these shortcomings. We develop an approximate MCMC method combining Langevin dynamics and reversible jump MCMC to conduct inference in this model. Experiments show that the proposed model learns a good combination of the structure and parameter values without the need for separate hyper-parameter tuning. Moreover, the model's predictive performance is much more robust than L_1-based methods with hyper-parameter settings that induce highly sparse model structures.

1 Introduction

Undirected probabilistic graphical models, also known as Markov Random Fields (MRFs), have been widely used in a large variety of domains including computer vision (Li, 2009), natural language processing (Sha and Pereira, 2003), and social networks (Robins et al., 2007). The structure of the model is defined through a set of features defined on subsets of random variables. Automated methods to select relevant features are becoming increasingly important in a time where the proliferation of sensors make it possible to measure a multitude of data-attributes. There is also an increasing interest in *sparse* model structures because they help against overfitting and are computationally more tractable than dense model structures.

In this paper we focus on a particular type of MRF, called a log-linear model, where structure learning or feature selection is integrated with parameter estimation. Although structure learning has been extensively studied for directed graphical models, it is typically more difficult for undirected models due to the intractable normalization term of the probability distribution, known as the partition function. Traditional algorithms apply only to restricted types of structures with low tree-width (Andrew and Gao, 2007; Tsuruoka et al., 2009; Hu et al., 2009) or special models such as Gaussian graphical models (Jones et al., 2005) so that accurate inference can be conducted efficiently.

For an arbitrary structure, various methods have been proposed in the literature, generally categorized into two approaches. One approach is based on separate tests on an edge or the neighbourhood of a node so that there is no need to compute the joint distribution (Wainwright et al., 2007; Bresler et al., 2008; Ravikumar et al., 2010). The other approach is based on maximum likelihood estimation (MLE) with a sparsity inducing criterion. These methods require approximate inference algorithms in order to estimate the log-likelihood such as Gibbs sampling (Della Pietra et al., 1997), loopy belief propagation (Lee et al., 2006; Zhu et al., 2010), or pseudo-likelihood (Höfling and Tibshirani, 2009). A popular choice of such a criterion is L_1 regularization (Riezler and Vasserman, 2004; Dudik et al., 2004) which enjoys several good properties such as a convex objective function and a consistency guarantee. However, L_1-regularized MLE is usually sensitive to the choice the regularization strength, and these optimization-based methods cannot provide a credible interval for the learned structure. Also, in order to learn a sparse structure, a strong penalty has to be imposed on all the edges which usually results in suboptimal parameter values.

We will follow a third approach to MRF structure learning in a fully Bayesian framework which has not been ex-

plored yet. The Bayesian approach considers the structure of a graphical model as random. Inference in a Bayesian model provides inherent regularization, and offers a fully probabilistic characterization of the underlying structure. It was shown in Park and Casella (2008) that Bayesian models with a Gaussian or Laplace prior distribution (corresponding to L_2 or L_1 regularization) do not exhibit sparse structure. Mohamed et al. (2011) proposes to use a "spike and slab" prior for learning *directed* graphical models which corresponds to the ideal L_0 regularization. This model exhibits better robustness against over-fitting than the related L_1 approaches. Unlike the Laplace/Gaussian prior, the posterior distribution over parameters for a "spike and slab" prior is no longer guaranteed to be unimodal. However, approximate inference methods have been successfully applied in the context of directed models using MCMC (Mohamed et al., 2011) and expectation propagation (Hernández-Lobato et al., 2010).

Unfortunately, Bayesian inference for MRFs is much harder than for directed networks due to the partition function. This feature renders even MCMC sampling intractable which caused some people to dub these problems "double intractability" (Murray et al., 2006). Nevertheless, variational methods (Parise and Welling, 2006; Qi et al., 2005) and MCMC methods (Murray and Ghahramani, 2004) have been successfully explored for approximate inference when the model structure is fixed.

We propose a Bayesian structure learning method with a spike and slab prior for MRFs and devise an approximate MCMC method to draw samples of both the model structure and the model parameters by combining a modified Langevin sampler with a reversible jump MCMC method. Experiments show that the posterior distribution estimated by our inference method matches the actual distribution very well. Moreover, our method offers better robustness to both under-fitting and over-fitting than L_1-regularized methods. A related but different application of the spike and slab distribution in MRFs is shown in Courville et al. (2011) for modelling hidden random variables.

This paper is organized as follows: we first introduce a hierarchical Bayesian model for MRF structure learning in section 2 and then describe an approximate MCMC method in section 3, 4, and 5 to draw samples for the model parameters, structure, and other hyper-parameters respectively. Experiments are conducted in section 6 on two simulated data sets and a real-world dataset, followed by a discussion section.

2 Learning the Structure of MRFs as Bayesian Inference

The probability distribution of a MRF is defined by a set of potential functions. Consider a widely used family of MRFs with log-linear parametrization:

$$P(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_{\alpha} \theta_\alpha f_\alpha(\mathbf{x}_\alpha)\right) \tag{1}$$

where each potential function is defined as the exponential of the product between a feature function f_α of a set of variables $\mathbf{x}_\alpha$ and an associated parameter θ_α. Z is called the partition function. All the variables in the scope of a potential function form a clique in their graphical representation. When a parameter θ_α has a value of zero, we could equivalently remove feature f_α and all the edges between variables in $\mathbf{x}_\alpha$ (if these variables are not also in the scope of other features) without changing the distribution of $\mathbf{x}$. Therefore, by learning the parameters of this MRF model we can simultaneously learn the structure of a model if we allow some parameters to go to zero.

The Bayesian learning approach to graphical models considers parameters as a random variable subject to a prior. Given observed data, we can infer the posterior distribution of the parameters and their connectivity structure. Two commonly used priors, the Laplace and the Gaussian distribution, correspond to the L_1 and L_2 penalties respectively in the associated optimization-based approach. Although a model learned by L_1-penalized MLE is able to obtain a sparse structure, the full Bayesian treatment usually results in a fully connected model with many weak edges as observed in Park and Casella (2008), without special approximate assumptions like the ones in Lin and Lee (2006). We propose to use the "spike and slab" prior to learn a sparse structure for MRFs in a fully Bayesian approach. The spike and slab prior (Mitchell and Beauchamp, 1988; Ishwaran and Rao, 2005) is a mixture distribution which consists of a point mass at zero (spike) and a widely spread distribution (slab):

$$P(\theta_\alpha) = (1 - p_0)\delta(\theta_\alpha) + p_0 \mathcal{N}(\theta_\alpha; 0, \sigma_0^2) \tag{2}$$

where $p_0 \in [0, 1]$, δ is the Dirac delta function, and σ_0 is usually large enough to be uninformative. The spike component controls the sparsity of the structure in the posterior distribution while the slab component usually applies a mild shrinkage effect on the parameters of the existing edges even in a highly sparse model. This type of *selective* shrinkage is different from the global shrinkage imposed by L_1/L_2 regularization, and enjoys benefits in parameter estimation as demonstrated in the experiment section.

The Bayesian MRF with the spike and slab prior is formulated as follows:

$$\begin{aligned} P(\mathbf{x}|\boldsymbol{\theta}) &= \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_{\alpha} \theta_\alpha f_\alpha(\mathbf{x}_\alpha)\right) \\ \theta_\alpha &= Y_\alpha A_\alpha \\ Y_\alpha &\sim \mathrm{Bern}(p_0) \quad p_0 \sim \mathrm{Beta}(a, b) \\ A_\alpha &\sim \mathcal{N}(0, \sigma_0^2) \quad \sigma_0^{-2} \sim \Gamma(c, d) \end{aligned} \tag{3}$$

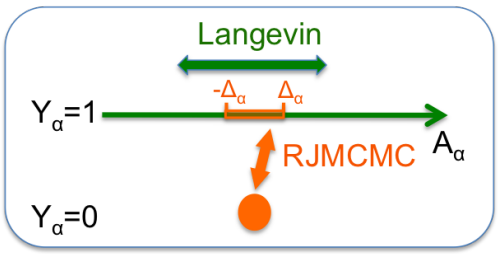


Figure 1: Illustration of the MCMC for A_α and Y_α.

where $\mathbf{x}$ is a set of state variables and a, b, c, and d are hyper-parameters. In the experiments we will use pairwise features in which case $\alpha = (i, j)$ plus bias terms given by $\sum_i \theta_i f_i(x_i)$ in the expression for the log-probability. We will use a normal prior $\theta_i \sim \mathcal{N}(0, \sigma_b^2)$ for these bias terms. σ_b is chosen to be large enough to act as an uninformative prior. Y_α is a binary random variable representing the existence of the edges in the clique $\mathbf{x}_\alpha$, and A_α is the actual value of the parameter θ_α when the edges are instantiated. It is easy to observe that given p_0 and σ_0, θ_α has the same distribution as in equation 2. We use a hierarchical model for $\boldsymbol{\theta}$ so that the inference will be insensitive to the choice of the hyper-parameters. In fact, experiments show that with a simple setting of the hyper-parameters, proper values of the sparsity parameter p_0 and the variance σ_0 are learned automatically by our model for all the data sets without the necessity of cross-validation.

Unlike the optimization-based methods which estimate a single structure, the Bayesian approach expresses uncertainty about the existence of edges through its posterior distribution, $P(\mathbf{Y}|\mathcal{D})$. We have applied a simple thresholding on $P(Y_\alpha|\mathcal{D})$ for edge detection in the experiments although more sophisticated methods can conceivably give better results.

Standard approaches to posterior inference do not work for Bayesian MRFs because it is intractable to compute the probability of a state $\mathbf{x}$ (due to the intractability of the partition function). We devised an approximate MCMC algorithm for inference, where we draw samples of the continuous variable A_α by a modified Langevin dynamics algorithm, and samples of the discrete variable Y_α jointly with A_α by a reversible jump MCMC method, as illustrated in Figure 1 and explained in the following sections.

3 Sampling Parameter Values by Langevin Dynamics

Given $\mathbf{Y}$, p_0, σ_0, and an observed data set $\mathcal{D} = \{\mathbf{x}^{(m)}\}, m = 1 \ldots N$, the conditional distribution of parameters $\{A_\alpha : Y_\alpha = 1\}$ is the posterior distribution of an MRF with a fixed edge set induced by $\{\alpha : Y_\alpha = 1\}$ and an independent Gaussian prior $\mathcal{N}(0, \sigma_0^2)$. We consider drawing samples of A_α with fixed $\mathbf{Y}$ in this section and will use θ_α and A_α interchangeably to refer to a nonzero parameter. Even for an MRF model with a fixed structure, MCMC is still intractable. Approximate MCMC methods have been discussed in Murray and Ghahramani (2004) among which Langevin Monte Carlo (LMC) with "brief sampling" to compute the required expectations in the gradients, shows good performance. (Welling and Teh, 2011) further shows that LMC with a noisy gradient can draw samples from the exact posterior distribution when the step size approaches zero.

Langevin dynamics is described as the hybrid Monte Carlo (HMC) method with one leapfrog step in section 5.5.2 of Neal (2010):

$$\begin{aligned} \boldsymbol{p}_{t+\varepsilon/2} &= \boldsymbol{p}_t + \frac{\varepsilon C}{2}\boldsymbol{g}(\boldsymbol{\theta}_t) \\ \boldsymbol{\theta}_{t+\varepsilon} &= \boldsymbol{\theta}_t + \varepsilon C \boldsymbol{p}_{t+\varepsilon/2} \\ \boldsymbol{p}_{t+\varepsilon} &= \boldsymbol{p}_{t+\varepsilon/2} + \frac{\varepsilon C}{2}\boldsymbol{g}(\boldsymbol{\theta}_{t+\varepsilon}) \end{aligned} \tag{4}$$

where $\boldsymbol{p}$ is the auxiliary momentum, ε is the step size, C is a positive definite preconditioning matrix, and $\boldsymbol{g}$ is the gradient of the log-posterior probability $\log P(\boldsymbol{\theta}|\mathcal{D})$ [1]. A new value of $\boldsymbol{p}$ is drawn at every iteration from an isotropic Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbb{I})$ and then discarded after $\boldsymbol{\theta}$ is updated. The leapfrog step is usually followed by a Metropolis-Hastings accept/reject step to ensure detailed balance. But since the rejection rate decays as ε^3, that step can be skipped for small step sizes without incurring much error.

In a MRF, the gradient term $\boldsymbol{g}$ involves computing an expectation over exponentially many states as

$$g_\alpha(\theta) = \sum_{m=1}^{N} f_\alpha(\mathbf{x}_\alpha^{(m)}) - N\mathbb{E}_{P(\mathbf{x}|\boldsymbol{\theta})} f_\alpha(\mathbf{x}_\alpha) - \frac{\theta_\alpha}{\sigma_0^2} \tag{5}$$

The expectation is estimated by a set of state samples $\{\tilde{\mathbf{x}}^{(s)}\}$ in the "brief Langevin" algorithm of Murray and Ghahramani (2004), where these samples are drawn by running a few steps of Gibbs sampling initialized from a subset of the training data. We adopt the "brief Langevin" algorithm with three modifications for faster mixing.

3.1 Persistent Gibbs Sampling

We maintain a set of persistent Markov chains for the state samples $\{\tilde{\mathbf{x}}^{(s)}\}$ by initializing Gibbs sampling at the last states of the previous iteration instead of the data. This is motivated by the persistent contrastive divergence algorithm of Tieleman (2008). When $\boldsymbol{\theta}$ changes slowly enough, the Gibbs sampler will approximately sample from the stationary distribution, even when allowed a few steps at every iteration.

[1] We omit all the other random variables that $\boldsymbol{\theta}$ is conditioned on in this section for ease of notation.

3.2 Preconditioning

When the posterior distribution of $\{\theta_\alpha\}$ has different scales along different variables, the original LMC with a common step size for all θ_α's will traverse the parameter space slowly. We adopt a preconditioning matrix C to speed up the mixing, where C satisfies $CC^T = H$ with H is the Hessian matrix of $\log P(\boldsymbol{\theta}_{\text{MAP}}|\mathcal{D})$, computed as:

$$H(\boldsymbol{\theta}_{\text{MAP}}) = \text{Cov}_{P(\mathbf{x}|\boldsymbol{\theta}_{\text{MAP}})}\boldsymbol{f}(\mathbf{x}) + \sigma_0^{-2} \tag{6}$$

This is reminiscent of the observed Fisher information matrix in Girolami and Calderhead (2011) except that we use the MAP estimate with the prior. We approximate $H(\boldsymbol{\theta}_{\text{MAP}})$ by averaging over $H(\boldsymbol{\theta}_t)$ during a burn-in period and estimate $\text{Cov}_{P(\mathbf{x}|\boldsymbol{\theta}_t)}\boldsymbol{f}$ with the set of state samples from the persistent Markov chains. The adoption of a preconditioning matrix also helps us pick a common step size parameter ε suitable for different training sets.

3.3 Partial Momentum Refreshment

The momentum term $\mathbf{p}$ in the leapfrog step represents the update direction of the parameter. Langevin dynamics is known to explore the parameter space through inefficient random walk behavior because it draws an independent sample for $\mathbf{p}$ at every iteration. We can achieve a better mixing rate with the partial momentum refreshment method proposed in Horowitz (1991). When $\mathbf{p}$ is updated at every step by:

$$\boldsymbol{p}_t \leftarrow \alpha \boldsymbol{p}_t + \beta \mathbf{n}_t \tag{7}$$

where $n_t \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$, and α, β satisfy $\alpha^2 + \beta^2 = 1$, the momentum is partially preserved from the previous iteration and thereby suppresses the random-walk behavior in a similar fashion as HMC with multiple leapfrog steps.

α controls how much momentum to be carried over from the previous iteration. With a large value of α, LMC reduces the auto-correlation between samples significantly relative to LMC without partial momentum refreshment. The improved mixing rate is illustrated in Figure 2. We also show that the mean and standard deviation of the posterior distribution does not change. However, caution should be exercised especially when the step size η is large because a value of α that is too large would increase the error in the update equation which we do not correct with a Metropolis-Hastings step because that is intractable.

4 Sampling Edges by Reversible Jump MCMC

Langevin dynamics handles the continuous change in the parameter value A_α. As for discrete changes in the model structure, Y_α, we propose an approximate reversible jump MCMC (RJMCMC) step (Green, 1995) to

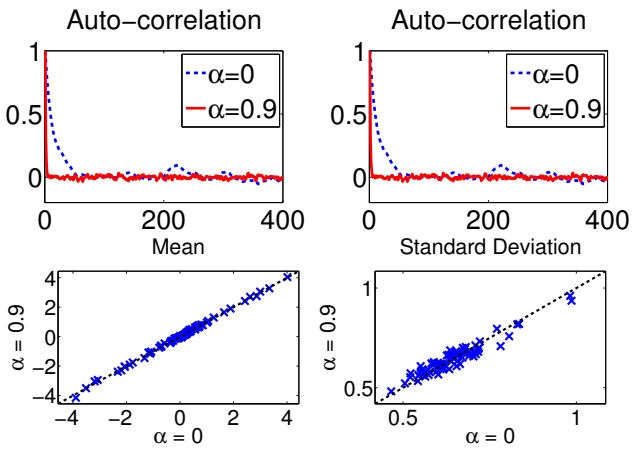


Figure 2: Comparison of Langevin dynamics on the block model in section 6.1 with partial momentum refreshment $\alpha = 0.9$ against $\alpha = 0$. Step size $\eta = 10^{-3}$. Top: the auto-correlation of two typical parameters. Bottom: the posterior mean and standard deviation of all parameters estimated with 10K samples.

sample Y_α and A_α jointly from the conditional distribution $P(\mathbf{A}, \mathbf{Y}|p_0, \sigma_0, \mathcal{D})$. The proposed Markov chain adds/deletes one clique (or simply one edge when $\alpha = (i, j)$) at a time. When an edge does not exist, i.e., $Y_\alpha = 0$, the variable A_α can be excluded from the model, and therefore we consider the jump between a full model with $Y_\alpha \neq 0$, $A_\alpha = a$ and a degenerate model with $Y_\alpha = 0$.

The proposed RJMCMC is as follows: when $Y_\alpha = 0$, propose adding an edge with probability P_{add} and sample $A_\alpha = a$ from a proposal distribution $q(A)$ with support on $[-\Delta_\alpha, \Delta_\alpha]$; when $Y_\alpha = 1$ and $|A_\alpha| \leq \Delta_\alpha$, propose deleting an edge with P_{del}. The reason of restricting the proposed move within $[-\Delta_\alpha, \Delta_\alpha]$ will be explained later. It is easy to see that the Jacobian is 1. The jump is then accepted by the Metropolis-Hastings algorithm with a probability:

$$\begin{aligned} Q_{add} &= \min\{1, Q^*(a)\}, \quad Q_{del} = \min\{1, 1/Q^*(a)\} \\ Q^*(a) &= \exp(a \sum_m f_\alpha(\mathbf{x}_\alpha^{(m)})) \left(\frac{Z(Y_\alpha = 0)}{Z(Y_\alpha = 1, A_\alpha = a)}\right)^N \\ &\quad \frac{p_0 \mathcal{N}(A_\alpha = a|0, \sigma_0^2)}{(1 - p_0)} \frac{P_{del}}{P_{add} q(A_\alpha = a)} \end{aligned} \tag{8}$$

The factors in the first line of Q^* represent the ratio of the model likelihoods while the other two are respectively the ratio of the prior distributions and the ratio of the proposal distributions.

4.1 Unbiased Estimate to Q^* and $1/Q^*$

Computing the partition functions in equation 8 is generally intractable. However, noticing that $Z(Y_\alpha = 1, A_\alpha = a) \to Z(Y_\alpha = 0)$, as $a \to 0$, the log-ratio of the two partition functions should be well approximated by a quadratic approximation at the origin when a is small. In this way

we reduce the problem of estimating the partition function to computing the first and second order derivatives of the log-partition function. We employ a second order Taylor expansion for the ratio of partition functions in Q^* as follows:

$$\left(\frac{Z(Y_\alpha=0)}{Z(Y_\alpha=1,A_\alpha=a)}\right)^N \stackrel{\text{def}}{=} R_{add}$$
$$\approx \exp\left(-Na\frac{\partial \log(Z)}{\partial A_\alpha}\mid_{A_\alpha=0} -\frac{Na^2}{2}\frac{\partial^2 \log(Z)}{\partial A_\alpha^2}\mid_{A_\alpha=0}\right) \tag{9}$$

We know that the kth order derivatives of the log-partition function of an MRF correspond to the kth order centralized moments (or cumulants) of the features, that is,

$$\frac{\partial \log(Z)}{\partial A_\alpha} = \mathbb{E}f_\alpha, \quad \frac{\partial^2 \log(Z)}{\partial A_\alpha^2} = \mathrm{Var}f_\alpha \tag{10}$$

Given a set of n state samples $\tilde{\mathbf{x}}^{(s)} \sim P(\mathbf{x}|Y_\alpha=0)$ from the persistent Markov chains, we can compute an unbiased estimate of the mean and variance of f_α as the sample mean $\bar{f}_\alpha$ and sample variance $S_\alpha^2 = \sum_s (f_\alpha(\tilde{\mathbf{x}}^{(s)}) - \bar{f}_\alpha)^2/(n-1)$ respectively. Consequently we obtain an estimate of R_{add} by plugging $\bar{f}_\alpha$ and S_α^2 into equation 9, which is unbiased in the logarithmic domain, denoted as $\hat{R}_{add}$.

An unbiased estimate in the logarithmic domain, unfortunately, is no longer unbiased once transformed to the linear domain. To correct the bias induced by the transformation, we take another Taylor expansion of $\hat{R}_{add}/R_{add}$ around $a=0$. After some derivation, we obtain an unbiased estimate of R_{add} upto the second order of a given by

$$\tilde{R}_{add}(a) = \exp\left(-Na\bar{f}_\alpha - \frac{Na^2}{2}S_\alpha^2\right)\left(1+\frac{N^2a^2}{2n}S_\alpha^2\right)^{-1}$$

with variance:

$$\mathrm{Var}(\tilde{R}_{add}/R_{add}) = \frac{N^2a^2}{n}\mathrm{Var}(f_\alpha) + o(a^3) \tag{11}$$

Similarly, we can also obtain an unbiased estimate, $\tilde{R}_{del}$, in $1/Q^*$ when considering deleting an edge, with the same formula as $\tilde{R}_{add}$ except that a is replaced by $-a$ and the sample mean and variance are now estimated with respect to $P(\mathbf{x}|Y_\alpha=1,A_\alpha=a)$. If we plug in $\tilde{R}_{add}$ (or $\tilde{R}_{del}$) into Q_{add} (or Q_{del}) we get an unbiased estimate of the acceptance probability except when Q^* (or $1/Q^*$) is close to 1 in which case the $\min$ operation causes extra bias. Since the variance can be computed as a function of a in equation 11, we can estimate how large a jump range Δ can be used in order to keep the error in the acceptance probability negligible. A larger data set requires a smaller jump range or alternatively a larger sample set that grows quadratically with the size of the data set.

4.2 Optimal Proposal Distribution q

After plugging equations 9, 10 and 5 into equation 8, we obtain the acceptance probability as a function of a:

$$Q_{add} = \min\left\{1, \frac{h(a)}{q(A_\alpha=a)}\mathrm{const}\right\}$$
$$h(a) = \exp\left(-\frac{a^2}{2}\left(\frac{1}{\sigma_0^2}+\mathrm{Var}f_\alpha\right)+aNg_\alpha\right) \tag{12}$$

where g_α and $\mathrm{Var}f_\alpha$ are defined at $\theta_\alpha=0$. Clearly, the optimal proposal distribution in terms of minimal variance is given by the following truncated Gaussian distribution

$$q_{opt}(A_\alpha=a|\boldsymbol{\theta}) \propto h(a), \quad a\in[-\Delta_\alpha,\Delta_\alpha] \tag{13}$$

When adding an edge, we have state samples $\tilde{\mathbf{x}}^{(s)} \sim P(\mathbf{x}|Y_\alpha=0)$. The expectation $\mathbb{E}f_\alpha$ in g_α can be estimated by $\bar{f}_\alpha(\{\tilde{\mathbf{x}}^{(s)}\})$ and $\mathrm{Var}f_\alpha$ by $S_\alpha^2(\{\tilde{\mathbf{x}}^{(s)}\})$. When deleting an edge, the samples are from $P(\mathbf{x}|Y_\alpha=1,A_\alpha=a)$. We apply a quadratic approximation for $\log Z(\theta_\alpha)$, use equation 10, and estimate $\mathrm{Var}f_\alpha \approx S_\alpha^2$ and $\mathbb{E}f_\alpha \approx \bar{f}_\alpha - aS_\alpha^2$.

4.3 Parallel Proposals

Since the most computationally demanding step is to obtain a set of state samples $\{\tilde{\mathbf{x}}^{(s)}\}$, we want to reuse the samples whenever possible. Given that Δ_α is small enough, the parameter value does not change much when we accept an "add or delete edge" move. We can thus assume that the distribution of A_α on an edge is not affected much by an accepted move on other edges. As a result we do not have to rerun the Gibbs sampler after every edge operation, and we can propose jumps for all $\{\alpha : |A_\alpha| < \Delta_\alpha\}$ in parallel, using the same set of samples. This reduces the computation time significantly.

5 Sampling for Hyper-parameters

Given $\mathbf{A}$ and $\mathbf{Y}$, the hyper-parameters are easy to sample when using conjugate priors:

$$p_0|\mathbf{Y} \sim \mathrm{Beta}(a+\sum_\alpha I(Y_\alpha=1), b+\sum_\alpha I(Y_\alpha=0))$$
$$\sigma_0^{-2}|\mathbf{Y},\mathbf{A} \sim \Gamma(c+\frac{1}{2}\sum_\alpha I(Y_\alpha=1), d+\frac{1}{2}\sum_{\alpha:Y_\alpha=1} A_\alpha^2) \tag{14}$$

where I is the indicator function.

The whole inference algorithm is summarized in Alg 1.

6 Experiments

6.1 Datasets

We assess the performance of our proposed method on two simulated data sets and one real-world data set. For the

Algorithm 1 MCMC for Bayesian MRFs with Langevin Dynamics and RJMCMC

(Parameters: number of iterations $ITER$, number of Gibbs sampling steps N_{Gibbs}, sample size n, step size ε, partial momentum refreshment α, RJMCMC proposal width Δ_α.)
Initialize $\mathbf{A}$, $\mathbf{Y}$, and momentum $\mathbf{p}$ randomly
for $iter = 1 \to ITER$ **do**
  Sample p_0 and σ_0 given $\mathbf{A}$ and $\mathbf{Y}$
  Run Gibbs sampling for N_{Gibbs} steps to draw $\{\mathbf{x}^{(s)}\}_{s=1}^n$.
  Run LMC to draw $\{A_\alpha : Y_\alpha \neq 0\}$.
  Run RJMCMC to propose adding an edge for $\{\alpha : Y_\alpha = 0\}$, and deleting an edge for $\{\alpha : Y_\alpha = 1, |A_\alpha| < \Delta_\alpha\}$
end for

simulated data, we randomly generate sparse Ising models with binary ± 1 states and with parameters sampled from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$ where $\sigma = 0.5$ for edges and 0.1 for node biases. These models are then converted to their equivalent Boltzmann machines with binary $\{0, 1\}$ states from which we draw exact samples. Two structures are considered: (1) a block model with 12 nodes equally divided into 3 groups. Edges are added randomly within a group with a probability of 0.8, and across groups with 0.1. Edges within a group are strong and positively coupled. There are 66 candidate edges with 20 edges in the ground truth model. (2) a 10×10 lattice with 4950 candidate edges and with 180 edges in the ground truth model.

For the real data, we use the MNIST digits image data. We convert the gray scale pixel values to binary values by thresholding at a value of 50, resize the images to a 14×14 scale, and then pick a 9×12 patch centered in each image where the average value of each pixel is in the range of $[10^{-4}, 1 - 10^{-4}]$. The last step is necessary because the other competing models do not have regularization on their biases, which will result in divergent parameter estimates for pixels that are always 0 or 1.

6.2 Model Specification

We compare our Bayesian structure learning algorithm with two other approaches. One is proposed in Wainwright et al. (2007) which recovers the neighbourhood of nodes by training separate L_1-regularized logistic regressions on each variable. While its goal is edge detection, we can also use it as a parameter estimation method with two output models, "Wain Max" and "Wain Min", as defined in Höfling and Tibshirani (2009). We implement the "Wain Max/Min" methods with the Lasso regularized generalized linear model package of Friedman et al. (2010) [2]. The other approach is one of the several variants of L_1-regularized MLE methods which use a pseudo-likelihood approximation (Höfling and Tibshirani, 2009), denoted as "MLE" [3].

For our Bayesian model, we consider two schemes to specify a model for prediction. One is the fully Bayesian approach, referred as "Bayes", in which we random pick 100 model samples in the Markov chain and approximate the Bayesian model by a mixture model of these 100 components. The other one is to obtain a single model by applying a threshold of 0.5 to $P(Y_{i,j}|\mathcal{D})$ and estimate the posterior mean of the included edges, referred to as "Bayes PM".

The performance of the Bayesian model is insensitive to the choice of hyper-parameters. We simply set $a = b = c = d = 5$ for p_0 and σ_0, and $\sigma_b = 10$ across all experiments. For the parameters of the MCMC method, we also use a common setting. We use a diagonal approximation to the feature covariance $\mathrm{Cov}\boldsymbol{f}$ and thereby the preconditioning matrix C. We set the sample size $n = 100$, number of Gibbs sampling steps $N_{Gibbs} = 1$, LMC step size $\varepsilon = 10^{-3}$, and momentum refreshment rate $\alpha = 0.9$. The RJMCMC proposal width is set as $\Delta_\alpha = 0.01/\sqrt{N \mathrm{Var} f}$ to achieve a small estimation error as in equation 11. For each experiment, we run the MCMC algorithm to collect $10K$ samples with a subsampling interval of 1000. Since the exact partition function can be computed on the small block model, we also run an exact MCMC, "Bayes Exact", with an exact gradient and accept/reject decision as well as larger values in ε, α, and Δ than the approximate MCMC.

Different levels of sparsity have to be considered in L_1-based methods for an optimal regularization strength. For the Bayesian method, we learn a single sparsity level. However, for the sake of comparison, we also consider a method with p_0 as a parameter and vary it between $(0, 1)$ to induce different sparsity, referred to with a suffix "p_0".

6.3 Accuracy of Inference

We first evaluate the validity of the proposed MCMC method on the block data where exact inference can be carried out. The marginal posterior distribution of an edge parameter, $\theta_{i,j}$, is a mixture of a point mass at zero and a continuous component with a single mode. Figure 3 shows the histogram of samples in the continuous component of four randomly picked $\theta_{i,j}$'s. The title above each plot is the posterior probability of the edge (i, j) being present in the model, i.e. $\theta_{i,j} \neq 0$ or $Y_{i,j} = 1$. In this figure, the marginal distribution estimated from the approximate MCMC method matches the distribution from "Bayes Exact" very well. For a more comprehensive comparison, we run "Bayes p_0" and "Bayes Exact p_0" methods, and vary the value of p_0 from 10^{-4} to 0.99 to achieve different levels of sparsity. We estimate and collect across different p_0

[2] Code provided at http://www-stat.stanford.edu/ tibs/glmnet-matlab

[3] Code provided at http://holgerhoefling.com

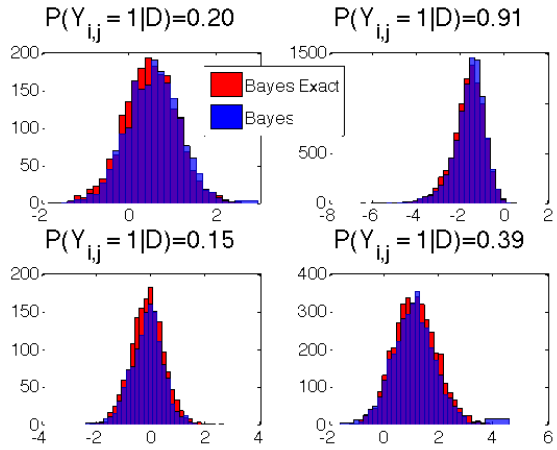


Figure 3: Histogram of the parameter samples of four randomly picked edges from "Bayes" and "Bayes Exact" when $Y_{i,j} = 1$. The training data is from the block model with $N = 100$.

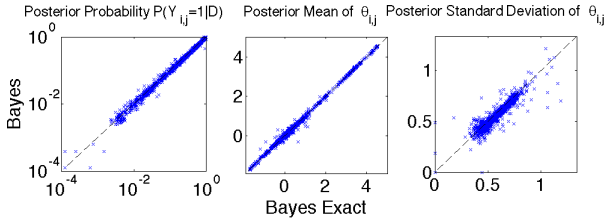


Figure 4: Comparison of "Bayes p_0" and "Bayes Exact p_0" on posterior probability of $Y_{i,j} = 1$, and the posterior mean and standard deviation of $\theta_{i,j}$ when $Y_{i,j} = 1$. The training data is from the block model with $N = 100$.

values the posterior probability of $\theta_{i,j} \neq 0$, posterior mean and standard deviation of $\theta_{i,j}$ in the continuous component, as shown in Figure 4. Each point represents a parameter under some value of p_0. We find the approximate MCMC procedure produces about the same values for these three statistics as the exact MCMC method.

6.4 Simulated Data

We then compare the performance of various methods on simulated data sets for two tasks: structure recovery and model estimation. The accuracy of recovering the true structure is measured by the precision and recall of the true edges. The quality of the estimated models is evaluated by the predictive performance on a held-out validation set. Since computing the log-likelihood of a MRF is in general intractable, we use the conditional log-likelihood (CLL) on a group of variables instead, which is a generalization of the conditional marginal log-likelihood in Lee et al. (2006). For each data case in the validation set, we randomly choose a group of variables, which is all the variables in the block model and a 3×3 grid in the lattice and MNIST models, and compute $\log P(\{x_i\}_{i \in group} | \{x_j\}_{j \notin group})$.

We train models on the simulated data ranging between 50 and 1000 items. For the Bayesian models, we remove an edge if the posterior probability $P(Y_{i,j} = 1|\mathcal{D}) < 0.5$.

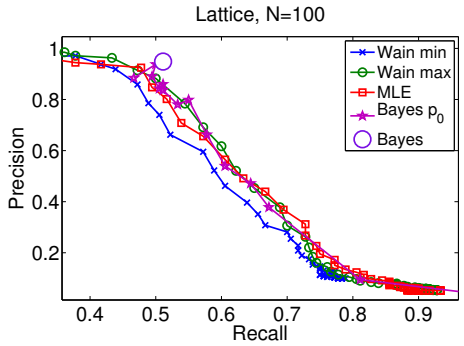


Figure 5: Precision-Recall Curves for the lattice data with 100 data cases.

Figure 5 shows typical precision-recall curves for different methods. It turns out that all the models with a sparsity tuning parameter perform similarly across all the training sets. The "Bayes" model tends to find a structure with high precision.

We then consider the average CLL as a function of the edge density of a model. The edge density is defined as the percentage of edges present in a model, i.e., $1 - sparsity$. For the fully Bayesian model, we measure the average density in the Markov chain. In fact, the variance of the edge density is usually small, suggesting that most model samples have about the same number of edges. We show the average CLL of different models trained with 100 data cases in Figure 6. The results on other data sizes are omitted because they have the same tendency. All the Bayesian models give robust prediction performance in the sparse and dense model ranges and "Bayes p_0" outperforms all the other methods with a tunable sparsity parameter for almost all settings of p_0. Moreover, the curve of "Bayes p_0" peaks at the true density level. In contrast, L_1 methods would underfit to the data for sparse models or overfit for dense models. "MLE" performs better than "Wain" models. This makes sense as the "Wain" models were not designed for MRF parameter estimation. Figure 7 compares the Bayesian models with exact and approximate inference. Again, the approximate MCMC method generates about the same posterior distribution as the "exact" method.

The difference between Bayesian models and L_1-based models could be partially explained by the different prior/regularization. As shown in Figure 8, to achieve a sparse structure, we have to use strong regularization in the L_1 models which causes global shrinkage for all parameters, resulting in under-fitting. On the other hand, to obtain a dense structure, weak regularization must be applied globally which leads to over-fitting. In contrast, with a spike and slab prior, the parameter value of existing edges is not affected directly by p_0. Instead, their variance is controlled by another random variable σ_0 which fits the data automatically with a weak hierarchical prior. The behavior of selective shrinkage in the spike and slab prior is also discussed in Mohamed et al. (2011); Ishwaran and Rao (2005).

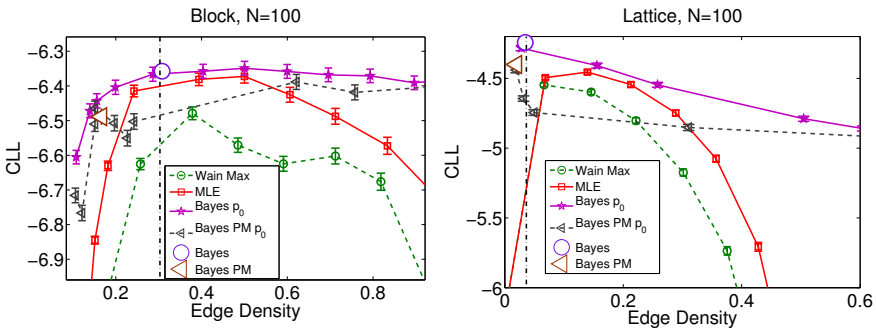


Figure 6: Mean and standard deviation of average CLL at different density levels for block (left) and lattice (right) model with 100 training data cases. "Wain Min" is not plotted as it is always inferior to "Wain Max". Vertical line: true edge density.

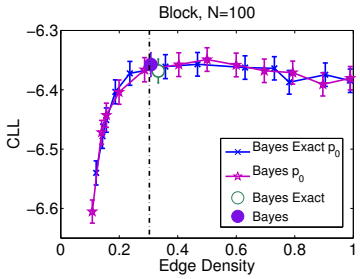


Figure 7: Average CLL at different density levels for the block model with 100 training cases.

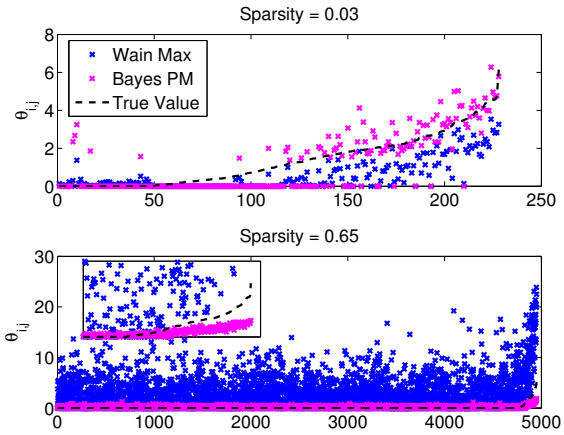


Figure 8: Absolute value of edge parameters in the true model, "Wain Max", and "Bayes PM" for lattice data with 100 samples. Parameters are plotted only if they are not zero in at least one model and sorted along the x-axis by the absolute value in the true model. The parameters in "Wain Max" are too small in sparse models (top, edge density $\approx 3\%$) and too large in dense models (bottom, 65%). Inset: zoom-in at the right-hand side. "Wain Min" and "MLE" are similar to "Wain Max".

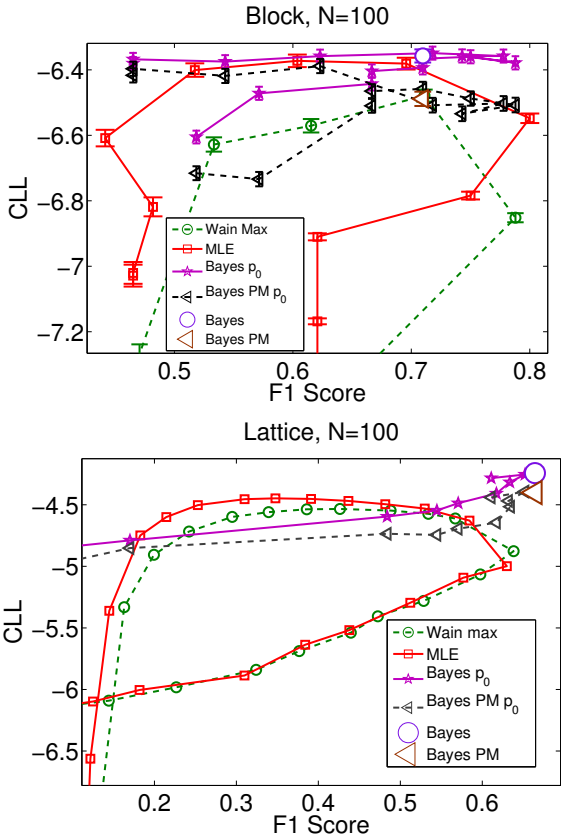


Figure 9: CLL vs F1 score for block and lattice model with 100 data cases. A good model is at the upper right corner.

The Bayesian models, "Bayes p_0" and "Bayes PM p_0" do however show an interesting "under-fitting" phenomenon at a large density levels. This is due to a misspecified value for p_0. Since the standard deviation σ_0 is shared by all the $A_{i,j}$'s whose $Y_{i,j} = 1$, when we fix p_0 at an improperly large value, it forces a lot of non-existing edges to be included in the model, which consequently brings down the posterior distribution of σ_0. This results in too small values on real edges as shown in the inset of Figure 8 and thereby a decrease in the model predictive accuracy.

However, this "misbehavior" in return just suggests the ability of our Bayesian model to learn the true structure. Once we release p_0 through a hierarchical prior, the model will abandon these improper values in p_0 and automatically find a good structure and parameters. The vertical line in Figure 6 indicates the sparsity of the true model. Both "Bayes" and "Bayes PM" find sparsity levels very close to the true value, while L_1-based methods are under-fitted at that same level as shown in the upper panel of Figure 8. We show the joint performance of edge detection and parameter learning in Figure 9 where the performance of edge detection is summarized by the F1 score (the harmonic mean of the precision and recall). The Bayesian models with a hierarchical p_0 prior achieve both a high F-1 score and CLL value near the upper right corner.

6.5 MNIST Data

Since there does not exist ground truth in the model structure of the MIST data set, we evaluate how well we can learn a sparse model for prediction. Figure 10 shows the average CLL on $10K$ test images with a model trained on

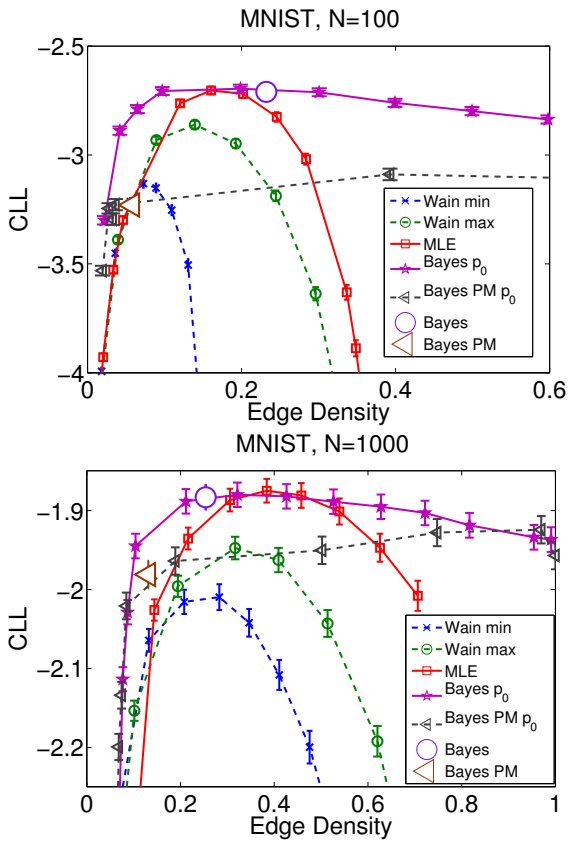


Figure 10: Mean and standard deviation of average CLL versus edge density on MNIST with 100 and 1000 data cases.

100 and 1000 images respectively. In the sparse and dense model ranges, we observe again a better performance of "Bayes" than L_1-based methods. "Bayes PM" also shows robustness to under/over-fitting although it seems that simply computing the posterior mean does not provide sufficiently good model parameters in the median density range.

To get a more intuitive comparison about the quality of learned sparse models, we train models on 1000 images by different methods with a density of 0.2 and then run Gibbs sampling to draw 36 samples from each model. The images are shown in Figure 11. While it is hard to get good reconstruction using a model without hidden variables, the Bayesian methods produce qualitatively better images than competing methods, even though "Bayes PM" does not have higher CLL than "MLE" at this level.

A common limitation of learning Bayesian models with the MCMC inference method is that it is much slower than training a model with a point estimate. However, as shown in the experiments, the Bayesian methods are able to learn a good combination of parameters and a structure without the need to tune hyper-parameters through cross validation. Also, the Bayesian methods learn sparser models than L_1-based methods without sacrificing predictive performance significantly. Because the computational complexity of inference grows exponentially with the maximum clique size of MRFs, L_1-based models at their optimal (not so

Figure 11: Samples from learned models at an edge density level of 0.2.

sparse) regularization level can in fact become significantly more computationally expensive than their Bayesian counterparts at prediction time. Turning up the regularization will result in sparser models but at the cost of under-fitting the data and thus sacrificing predictive accuracy.

7 Discussion

We propose Bayesian structure learning for MRFs with a spike and slab prior. An approximate MCMC method is proposed to achieve effective inference based on Langevin dynamics and reversible jump MCMC. As far as we known this is the first attempt to learn MRF structures in the fully Bayesian approach using spike and slab priors. Related work was presented in Parise and Welling (2006) with a variational method for Bayesian MRF model selection. However this method can only compare a given list of candidate models instead of searching in the exponentially large structure space.

The proposed MCMC method is shown to provide accurate posterior distributions at small step sizes. The selective shrinkage property of the spike and slab prior enables us to learn an MRF at different sparsity levels without noticeably suffering from under-fitting or over-fitting even for a small data set. Experiments with simulated data and real-world data show that the Bayesian method can learn both an accurate structure and a set of parameter values with strong predictive performance. In contrast the L_1-based methods could fail to accomplish both tasks with a single choice of the regularization strength. Also, the performance of our Bayesian model is largely insensitive to the choice of hyper-parameters. It provides an automated way to choose a proper sparsity level, while L_1 methods usually rely on cross-validation to find their optimal regularization setting.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 0914783, 0928427, 1018433.

References

G. Andrew and J. Gao. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM, 2007.

G. Bresler, E. Mossel, and A. Sly. Reconstruction of Markov random fields from samples: Some observations and algorithms. *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 343–356, 2008.

A. Courville, J. Bergstra, and Y. Bengio. A spike and slab restricted Boltzmann machine. *Journal of Machine Learning Research, W&CP*, 15, 2011.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):380–393, 1997.

M. Dudik, S. Phillips, and R. Schapire. Performance guarantees for regularized maximum entropy density estimation. *Learning Theory*, pages 472–486, 2004.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL `http://www.jstatsoft.org/v33/i01/`.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

P.J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

D. Hernández-Lobato, J. Hernández-Lobato, T. Helleputte, and P. Dupont. Expectation propagation for Bayesian multi-task feature selection. *Machine Learning and Knowledge Discovery in Databases*, pages 522–537, 2010.

H. Höfling and R. Tibshirani. Estimation of sparse binary pairwise Markov networks using pseudo-likelihoods. *The Journal of Machine Learning Research*, 10:883–906, 2009.

A.M. Horowitz. A generalized guided Monte Carlo algorithms. *Physics Letters B*, 268(2):247–252, 1991.

J. Hu, A. Joshi, and V.E. Johnson. Log-linear models for gene association. *Journal of the American Statistical Association*, 104(486):597–607, 2009.

H. Ishwaran and J.S. Rao. Spike and slab variable selection: frequentist and Bayesian strategies. *The Annals of Statistics*, 33(2):730–773, 2005.

B. Jones, C. Carvalho, A. Dobra, C. Hans, C. Carter, and M. West. Experiments in stochastic computation for high-dimensional graphical models. *Statistical Science*, 20(4):388–400, 2005.

S.I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1 regularization. In *In NIPS*. Citeseer, 2006.

S.Z. Li. *Markov random field modeling in image analysis*. Springer-Verlag New York Inc, 2009.

Y. Lin and D.D. Lee. Bayesian L1-norm sparse learning. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V. IEEE, 2006.

T.J. Mitchell and J.J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, pages 1023–1032, 1988.

S. Mohamed, K. Heller, and Z. Ghahramani. Bayesian and L1 approaches to sparse unsupervised learning. *Arxiv preprint arXiv:1106.1157*, 2011.

I. Murray and Z. Ghahramani. Bayesian learning in undirected graphical models: approximate MCMC algorithms. In *Proceedings of the 14th Annual Conference on Uncertainty in AI*, pages 392–399, 2004.

I. Murray, Z. Ghahramani, and D.J.C. MacKay. MCMC for doubly-intractable distributions. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, Pittsburgh, PA, 2006.

R.M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo: Methods and Applications*, page 113, 2010.

S. Parise and M. Welling. Structure learning in Markov random fields. *Advances in Neural Information Processing Systems*, 2006.

T. Park and G. Casella. The Bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.

Y. Qi, M. Szummer, and T.P. Minka. Bayesian conditional random fields. In *Artificial Intelligence and Statistics*, 2005.

P. Ravikumar, M.J. Wainwright, and J.D. Lafferty. High-dimensional ising model selection using L1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010.

S. Riezler and A. Vasserman. Incremental feature selection and L1 regularization for relaxed maximum-entropy modeling. In *Proceedings of EMNLP*, volume 4, 2004.

G. Robins, P. Pattison, Y. Kalish, and D. Lusher. An introduction to exponential random graph models for social networks. *Social networks*, 29(2):173–191, 2007.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of*

the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, pages 134–141. Association for Computational Linguistics, 2003.

T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradients. In *Proceedings of the International Conference on Machine Learning*, volume 25, pages 1064–1071, 2008.

Y. Tsuruoka, J. Tsujii, and S. Ananiadou. Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics, 2009.

M.J. Wainwright, P. Ravikumar, and J.D. Lafferty. High-dimensional graphical model selection using L1-regularized logistic regression. *Advances in neural information processing systems*, 19:1465, 2007.

M. Welling and Y.W. Teh. Bayesian learning via stochastic gradient langevin dynamics. 2011.

J. Zhu, N. Lao, and E.P. Xing. Grafting-light: fast, incremental feature selection and structure learning of Markov random fields. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 303–312. ACM, 2010.

Designing Informative Securities*

Yiling Chen
Harvard University

Mike Ruberry
Harvard University

Jennifer Wortman Vaughan
University of California, Los Angeles

Abstract

We create a formal framework for the design of informative securities in prediction markets. These securities allow a market organizer to infer the likelihood of events of interest as well as if he knew all of the traders' private signals. We consider the design of markets that are always informative, markets that are informative for a particular signal structure of the participants, and informative markets constructed from a restricted selection of securities. We find that to achieve informativeness, it can be necessary to allow participants to express information that may not be directly of interest to the market organizer, and that understanding the participants' signal structure is important for designing informative prediction markets.

1 INTRODUCTION

Prediction markets are often used to better understand the likelihood of future events. Consider, for example, a market that predicts whether a particular candidate will become the U.S. President. Traders in the market may have diverse private information, like whether the candidate will win a particular state or receive millions of dollars in donations. Pooling this information could lead to an accurate forecast of the likelihood that the candidate will be elected, but may or may not be possible depending on how well the market is designed.

A typical prediction market offers securities with payoffs associated with future events. For example, a prediction market might offer a security worth \$1 if the candidate is elected and \$0 otherwise. A risk neutral trader who believes that the probability of election is p would be willing to buy this security at any price less than \$$p$, and (short) sell the security at any price above \$$p$. For this reason, the market price of a security of this form is frequently interpreted as the traders' collective belief about the likelihood of election.

Prediction markets have been shown to produce forecasts at least as accurate as other alternatives in a wide variety of settings, including politics [6], business [11, 42], disease surveillance [38], and entertainment [34]. The New York Times has cited prices from the popular Dublin-based market Intrade in discussions of upcoming elections,[1] and the North American Derivatives Exchange has proposed running real-money prediction markets for political events.[2] Extensive prior work has studied markets' abilities to aggregate information relevant to the value of a given security in both lab and field experiments [35–37] and at theoretical equilibria such as rational expectations equilibria [39], competitive equilibria [32, 43], and game-theoretic equilibria [10, 24, 31]. However, there has been little research studying the problem of designing securities to aggregate information relevant to externally specified events.

We develop a formal framework in which to study the design of *informative* securities for predicting the likelihood of events of interest. Loosely speaking, we consider a set of securities informative if, given their prices, it is possible to calculate the posterior probabilities of the events as if we were given the traders' private information. This suggests two requirements:

1. The price of each security offered should converge to the expected value of the security conditioned on the traders' pooled information.

*This research was partially supported by the National Science Foundation under grants IIS-1054911 and CCF-0953516. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors alone.

[1] http://thecaucus.blogs.nytimes.com/2012/01/20/gingrich-gets-a-boost-on-intrade/

[2] http://www.freakonomics.com/2012/02/02/the-politics-of-political-prediction-markets/

2. It should be possible to uniquely map the final security prices to estimates of the likelihood of each event of interest.

Prior work [31] has described when a given security meets the first condition in perfect Bayesian equilibria. We focus on the design of securities that meet both conditions with respect to a specified set of events. This theoretical investigation offers insight on how to design prediction markets in practice to produce more accurate forecasts.

Poor security design can stop participants from aggregating their information, as the following example derived from Geanakoplos and Polemarchakis [18] shows.

Example 1. *Consider a market offering a single security worth \$1 if a particular candidate wins the presidential election and \$0 otherwise. The market has two participants: a political analyst in Washington and an Iowa caucus-goer who is well-informed on local politics. The analyst understands the importance of Iowa on the campaign and knows whether a win or loss there will mean the candidate is elected. The caucus-goer, on the other hand, knows whether the candidate will win or lose the caucus, but not its broader effect.*

This situation can be described by defining four states of the world, ω_1, ω_2, ω_3, and ω_4:

		Iowa	
		Wins	Loses
General Election	Wins	ω_1	ω_2
	Loses	ω_3	ω_4

The analyst knows if the true state of the world is on the diagonal or not (the effect of the caucus) and the caucus-goer knows which column the true state is in (the results of the caucus). If they could reveal their private information they would learn the true state of the world, ω^*. But with a uniform prior over the state space, both think the likelihood of election is $1/2$ and value the security at \$0.50 no matter what their private information is, since every signal contains a state where the candidate wins the election and another where the candidate loses. This prevents either from inferring the other's signal, and no aggregation occurs.

The traders in Example 1 are unable to effectively express their private information and determine the correct value for the security, even though they have between them the knowledge to do so. But even when a market determines the correct value for its securities nothing may be learned. A constant-valued security worth \$1 regardless of the outcome is, for example, always priced correctly, but tells us nothing about the likelihood of future events.

In practice many prediction markets offer multiple securities, like a security that pays \$1 if and only if a candidate wins Iowa and another that pays \$1 if and only if the candidate wins the general election. These two securities together would allow the analyst and caucus-goer to aggregate their information. The caucus-goer could first participate in the Iowa market. Then, having observed the price change in the Iowa market, the analyst would have the information he needs to participate in the general election market, revealing his own private information. Furthermore, the market organizer would be able to infer the probability of election by observing their trades. In short, these securities together are what we call *informative* on the candidate's election. We show that multiple securities are necessary for informativeness in some cases. Using too many securities, however, is bad design.

The design questions we examine are also relevant for *combinatorial prediction markets* over exponentially large outcome spaces, like the 9.2 quintillion outcomes for the NCAA tournament [44], the over 2^{50} ways for states to vote in the U.S. presidential election, and the $n!$ rankings for a competition with n candidates. Offering a security for each state would be technically informative but practically unmanageable. Prior work has frequently considered markets with securities that correspond to natural events of interest, such as horse A beating horse B in a race [3, 9], but these securities may or may not be informative.

This paper considers the design of informative markets in three settings:

- In Section 5.2, we characterize securities that are *always informative*, revealing traders' information regardless of their signal structure. Complete markets, which offer one Arrow-Debreu security associated with each state of the world, are always informative for any set of events because they reveal traders' posterior distribution over the state space, but are typically too large to run in practice. We show that even if only one event is of interest, prohibitively large numbers of securities may be required to guarantee informativeness.
- Offering a large number of securities is undesirable from a practical point of view. In Section 5.3, we show that if the market designer knows the traders' signal structure, a *single* informative security can be constructed to reveal the traders' posterior distribution.
- Real markets have multiple securities, and a single security acting as a summary statistic for a complex market is unlikely to be considered natural. In Section 5.4 we consider design as an optimization problem where a designer attempts to

find the fewest securities that are informative for some given events of interest but is constrained to select its securities from a predefined set. This predefined set of securities could be interpreted as the set of securities that are natural. Solving this optimization problem perfectly is NP-hard, even when the set of securities is restricted to those paying \$0 or \$1 in each state of the world (like the securities in the above examples).

These results suggest, unsurprisingly, that deployed prediction markets are likely not revealing *all* their participants' information and that better design may improve upon their observed efficacy. The idea of partial informativeness, which may better describe prediction markets in practice, is discussed further in the conclusion. However, only by understanding the limits and opportunities of total informativeness, as discussed in this paper, can we understand the value and interest of partial informativeness.

2 RELATED WORK

There is a rich literature on the informational efficiency of markets, including theoretical work on the existence and characteristics of rational expectations equilibria [4, 21, 39] and empirical studies of experimental markets [35, 36]. Here we review only the most relevant theoretical work that either focuses on the dynamic process of information aggregation or takes a security design perspective.

The early theoretical foundations of information aggregation were laid by Aumann [5], who initiated a line of research on common knowledge, establishing a solid foundation to the understanding of the phenomenon of consensus. An event E is said to be *common knowledge* among a set of agents if every agent knows E, and every agent knows that every other agent knows E, ad infinitum. Aumann proved that if two rational agents have the same prior and their posterior probabilities for some event are common knowledge, then their posterior probabilities must be equal. This result was repeatedly refined and extended [18, 28, 29], and Nielsen et al. [30] showed that if n agents with the same prior but possibly different information announce their beliefs about the expectation of some random variable, their conditional expectations eventually are equal.

This line of work suggests that agents will reach consensus, but says nothing about whether such a consensus fully reveals agents' information. Feigenbaum et al. [14] studied a particular model of prediction markets (a Shapley-Shubik market game [41]) in which traders' information determines the value of a security. They characterized the conditions under which the market price of the security converges to its true value under the assumption that traders are non-strategic and honestly report their expectations.

The assumption that traders are non-strategic is arguably unrealistic. Ostrovsky [31] examined information aggregation in markets with strategic, risk-neutral traders. He considered two market models, market scoring rules [22, 23] and Kyle's model [25]. For both, he showed that a separability condition is necessary for the market price of a security to always converge to the expected value of the security conditioned on all information in every perfect Bayesian equilibrium. This condition is discussed extensively in Section 4. Iyer et al. [24] extended this model to risk-averse agents and identified a smoothness condition on the price in the market that ensures full information aggregation.

The work of Feigenbaum et al. [14], Ostrovsky [31], and Iyer et al. [24] focuses on understanding the aggregation of information relevant to the value of a given, fixed security. In contrast, this paper studies how to design securities to infer the likelihood of some events of interest. There have been other papers on security design. Pennock and Wellman [33], for example, examine the conditions under which an incomplete market with a compact set of securities allows traders to hedge any risk they have (and hence is "operationally complete"). Their work considers competitive equilibria, while our work focuses on information aggregation at game-theoretic equilibria of the market.

3 THE MODEL

In this section, we describe our model of traders' information and the market mechanism. Our model closely follows Ostrovsky [31], but is generalized to handle a *vector* of securities (often simply referred to as a set of securities) instead of a single security.

3.1 Modeling Traders' Information

We consider n traders, $1, \cdots, n$, and a finite set Ω of mutually exclusive and exhaustive states of the world. Traders share a common knowledge prior distribution P_0 over Ω. Before the market opens Nature draws a state ω^* from Ω according to P_0 and traders learn some information about ω^* that, following Aumann [5], is based on partitions of Ω. A *partition* of a set Ω is a set of nonempty subsets of Ω such that every element of Ω is contained in exactly one subset. For example, $\{\{A, B\}, \{C\}, \{D\}\}$ and $\{\{A, D\}, \{B, C\}\}$ are both partitions of $\{A, B, C, D\}$. We assume that every trader i receives $\Pi_i(\omega^*)$ as their private signal, where $\Pi_i(\omega)$ denotes the element of the partition Π_i that contains ω. In other words, trader i learns that the true

state of the world lies in the set $\Pi_i(\omega^*)$.

We refer to the vector $\Pi = (\Pi_1, \cdots, \Pi_n)$ as the traders' *signal structure*, which is assumed to be common knowledge for all traders. The *join* of the signal structure, denoted join(Π), is the coarsest common refinement of Π, that is, the partition with the smallest number of elements satisfying the property that for any ω_1 and ω_2 in the same element of the partition, $\Pi_i(\omega_1) = \Pi_i(\omega_2)$ for all i. For example, the join of the partitions $\{\{A, D\}, \{B, C\}\}$ and $\{\{A, C, D\}, \{B\}\}$ is $\{\{A, D\}, \{B\}, \{C\}\}$. The join is unique. We use $\Pi(\omega)$ to denote the element of the join containing ω. Note that if two states appear in the same element of the join, no trader can distinguish between these states.

3.2 Market Scoring Rules

The market mechanism that we consider is a market scoring rule [22, 23]. We will describe a market scoring rule as a mechanism that allows traders to sequentially report their probability distributions or expectations. While focusing on market scoring rules may seem restrictive, market scoring rules are surprisingly general. In particular, any market scoring rule that allows traders to report probability distributions over Ω has an equivalent implementation as a cost-function-based market where the mechanism acts as an automated market maker who sets prices for $|\Omega|$ Arrow-Debreu securities, one for each state and taking value 1 in that state and 0 otherwise, and is willing to buy and sell securities at the set prices [8, 22]. This result can easily be extended to general scoring rules by applying the results of Abernethy and Frongillo [1, 2]. In particular, their results imply that any market scoring rule that allows traders to report their expectations has an equivalent implementation as a cost-function-based market that allows traders to trade securities with the market maker. Thus, without loss of generality, our model and analysis are presented for market scoring rules.

Before describing the market scoring rule mechanism, we first review the idea of a strictly proper scoring rule. *Scoring rules* are most frequently used to evaluate and incentivize probabilistic forecasts [16, 19], but can also be used to elicit the mean or other statistics of a random variable [26]. The scoring rules that we consider will be used to elicit the mean of a *vector* of random variables [40]. Let $\mathbf{X} = (x_1, \cdots, x_m)$ be a vector of bounded real-valued random variables. A scoring rule s maps a forecast $\vec{y}$ in some convex region $\mathcal{K} \subseteq \mathbb{R}^m$ (e.g., the probability simplex in the case of probabilistic forecasts) and a realization of $\mathbf{X}$ to a score $s(\vec{y}, \mathbf{X}(\omega))$ in $\mathbb{R}$.[3] A scoring rule for eliciting an expectation is said to be *proper* if a risk neutral forecaster who believes that the true distribution over states Ω is P maximizes his expected score by reporting $\vec{y} = \mathrm{E}_P[\mathbf{X}]$, that is, if $\mathrm{E}_P[\mathbf{X}] \in \arg\max_{\vec{y} \in \mathcal{K}} \sum_{\omega \in \Omega} P(\omega) s(\vec{y}, \mathbf{X}(\omega))$. (For random vectors $\mathbf{X}$, we use $\mathrm{E}_P[\mathbf{X}]$ to denote the expected value $\Sigma_{\omega \in \Omega} P(\omega)\mathbf{X}(\omega)$.) A scoring rule is *strictly proper* if $\mathrm{E}_P[\mathbf{X}]$ is the unique maximizer.

One common example of a strictly proper scoring rule is the Brier scoring rule [7], which is based on Euclidean distance and can be written, for any $b > 0$, as $s(\vec{y}, \mathbf{X}(\omega)) = -b \sum_{j=1}^{m} (y_j - x_j(\omega))^2 = -b||\vec{y} - \mathbf{X}(\omega)||^2$.

Strictly proper scoring rules incentivize myopic traders to report truthfully, but do not provide a mechanism for aggregating predictions from multiple traders. Hanson [22, 23] introduced *market scoring rules* to address this problem. A market scoring rule is a sequentially shared strictly proper scoring rule.

Formally, let $\mathbf{X}$ be a vector of random variables.[4] The market operator specifies a strictly proper scoring rule s and chooses an initial prediction $\vec{y}_0$ for the expected value of $\mathbf{X}$; when there is a known common prior P_0, it is most natural to set $\vec{y}_0 = \mathrm{E}_{P_0}[\mathbf{X}]$. The market opens with initial prediction $\vec{y}_0$, and traders take turns submitting predictions. The order in which traders make predictions is common knowledge. Without loss of generality, we assume that traders $1, 2, \cdots, n$ take turns, in order, submitting predictions $\vec{y}_1, \vec{y}_2, \cdots, \vec{y}_n$, then the process repeats and the traders, in the same order, submit predictions $\vec{y}_{n+1}, \vec{y}_{n+2}, \cdots, \vec{y}_{2n}$. Traders repeat this process an infinite number of times before the market closes and Nature reveals ω^*. Each trader then receives a score $s(\vec{y}_t, \mathbf{X}(\omega^*))$ for each prediction made at some time t, but must pay $s(\vec{y}_{t-1}, \mathbf{X}(\omega^*))$, the score of the previous trader. The total payment to trader i (which may be negative) is then $\sum_{t=0}^{\infty} s(\vec{y}_{tn+i}, \mathbf{X}(\omega^*)) - s(\vec{y}_{tn+i-1}, \mathbf{X}(\omega^*))$.

3.3 Modeling Traders' Behavior

Together, the traders, state space, signal structure, security vector, and market scoring rule mechanism define an extensive form game with incomplete information. We consider Bayesian traders either acting in perfect Bayesian equilibrium or behaving myopically in this game. A perfect Bayesian equilibrium is a subgame perfect Bayesian Nash equilibrium. Loosely speaking, at a perfect Bayesian equilibrium, it must

[3] Technically, the region $\mathcal{K}$ should include the convex hull of the possible realizations of $\mathbf{X}$, a set equivalent to the possible expected values of $\mathbf{X}$. A full discussion of this and other properties of scoring rules is beyond the scope of this paper, but interested readers can see Savage [40].

[4] Typically market scoring rules are used for probabilistic forecasts in which case $\mathbf{X}$ would be a vector of indicator random variables, but this need not be the case.

be the case that each player's strategy is optimal (i.e., maximizes expected utility) given the player's beliefs and the strategies of other players at any stage of the game, and that players' beliefs are derived from strategies using Bayes' rule whenever possible. See González-Díaz and Meléndez-Jiménez [20] for a more formal description.

Perfect Bayesian equilibria can be difficult to compute and it is an open question whether they always exist in prediction markets, although in some special cases they do [10]. An alternative is to consider *myopic* Bayesian traders who simply maximize their expected payoff for the current round. Since strictly proper scoring rules myopically incentivize honest reports, these traders report their current posteriors each time they make a prediction.

4 AGGREGATION

Separability is used to characterize the conditions under which securities aggregate information about their own values. Building on ideas from DeMarzo and Skiadas [12, 13], Ostrovsky [31] characterized separability for a single security. He showed that in every perfect Bayesian equilibrium market prices will, in the limit, reflect the value of the security as if traders had revealed their private signals if and only if the security is separable. If a security is not separable, then there always exist priors and equilibrium strategies where no information aggregation occurs.

In this section, we generalize these prior definitions to multiple securities and arbitrary signal structures. Ostrovsky assumed a restricted class of signal structures without loss of generality, and these generalizations are uninteresting when only considering aggregation. They will be necessary to discuss informativeness, however, as the results of the next section demonstrate. We then restate Ostrovsky's equilibrium aggregation result in this setting. As previously discussed, perfect Bayesian equilibrium may or may not exist in prediction markets, and we also adapt and formalize prior work on information aggregation to show separability is also the necessary and sufficient condition for myopic traders to always aggregate their information.

Informative markets require separable securities. If a market uses separable securities then both traders in perfect Bayesian equilibrium and Bayesian traders acting myopically will, in the limit, value the security as if their private signals were revealed, and this allows a market designer to directly infer the likelihood of his events of interest from the securities' value. If a set of non-separable securities were used then the market designer could be required instead to perform additional inference and know the prior and traders' strategies.

As mentioned, we say a market aggregates information if, in the limit as time goes to infinity, the value of the securities approaches their value conditional on all the traders' private signals. Since each trader i receives the signal $\Pi_i(\omega^*)$, their pooled signal is $\bigcap_i \Pi_i(\omega^*) = \Pi(\omega^*)$.

Definition 1 (Aggregation)**.** *Information is* aggregated *with respect to a set of securities* $\mathbf{X}$*, signal structure* Π*, and common prior* P_0*, if the sequence of predictions* $\vec{y}_0, \vec{y}_1, \vec{y}_2, \cdots$ *converges in probability to the random vector* $E_{P_0}[\mathbf{X}|\Pi(\omega^*)]$.

A set of securities is separable if and only if the traders only agree on their value when it reflects their pooled information. That is, for any prior distribution there must be at least one trader whose private information causes them to dissent from a consensus, unless that consensus is the traders' collective best estimate.

Definition 2 (Separability)**.** *A set of securities* $\mathbf{X}$ *is* non-separable *under partition structure* Π *if there exists a distribution* P *over* Ω *and vector* $\vec{v}$ *such that* $P(\omega) > 0$ *on at least one state* $\omega \in \Omega$ *in which* $E_P[\mathbf{X}|\Pi(\omega)] \neq \vec{v}$*, and for every trader* i *and state* $\omega, P(\omega) > 0$*,*

$$E_P[\mathbf{X}|\Pi_i(\omega)] = \frac{\sum_{\omega' \in \Pi_i(\omega)} P(\omega')\mathbf{X}(\omega')}{\sum_{\omega' \in \Pi_i(\omega)} P(\omega')} = \vec{v}. \quad (1)$$

If a security is not non-separable then it is separable.

Here the vector $\vec{v}$ represents a possible consensus, only agreed upon if there is no alternative when the securities are separable. Separability is a property of the entire set of securities, as Example 2 demonstrates.

Example 2. *Let* $\Omega = \{\omega_1', \omega_2^*, \omega_3, \omega_4', \omega_5^*, \omega_6\}$*. Two traders have partitions as follows:*

$$\Pi_1 = \{\{\omega_1', \omega_2^*, \omega_3\}, \{\omega_4', \omega_5^*, \omega_6\}\}$$
$$\Pi_2 = \{\{\omega_1', \omega_5^*\}, \{\omega_3, \omega_4'\}, \{\omega_2^*, \omega_6\}\}$$

and there are two securities: x^* *with value one when* ω_2^* *or* ω_5^* *occurs and zero otherwise, and* x' *with value one when* ω_1' *or* ω_4' *occurs and zero otherwise.*

Both securities are individually non-separable with respect to Π*. If the prior* P *is uniform over* $\omega_1', \omega_2^*, \omega_5^*$*, and* ω_6*, then* $E_P[x^*|\Pi_i(\omega)] = 1/2$ *for* $i \in \{1, 2\}$ *and all* ω *such that* $P(\omega) > 0$*. Similarly, if* P *is uniform over* $\omega_1', \omega_3, \omega_4'$*, and* ω_5^**, then* $E_P[x'|\Pi_i(\omega)] = 1/2$ *for* $i \in \{1, 2\}$ *and all* ω *such that* $P(\omega) > 0$*. The join of traders' partitions, however, consists of singletons. Hence, both* $E_P[x^*|\Pi(\omega)]$ *and* $E_P[x'|\Pi(\omega)]$ *have value 0 or 1, not* $1/2$*, for all* ω*.*

But taken together the set of securities is separable with respect to Π*. Given any prior distribution* P

and a state ω, trader 2 either identifies ω with certain, which happens when P assigns 0 probability to the other state in its signal $\Pi_2(\omega)$, or assigns positive probability to both states in $\Pi_2(\omega)$. In the former case, $E_P[\mathbf{X}|\Pi_2(\omega)] = E_P[\mathbf{X}|\Pi(\omega)]$. In the latter case, trader 2's expected value for the securities is positive for both when $\omega \in (\omega_1', \omega_5^*)$, positive for only x' when $\omega \in (\omega_3, \omega_4')$, and positive for only x^* when $\omega \in (\omega_2', \omega_6)$. If the set of securities is non-separable there must exist a distribution $\tilde{P}$ and a vector $\vec{v}$ such that $\vec{v} \neq E_{\tilde{P}}[\mathbf{X}|\Pi(\tilde{\omega})]$ for some state $\tilde{\omega} \in \{\omega|P(\omega) > 0\}$ and $E_{\tilde{P}}[X|\Pi_2(\omega)] = \vec{v}$ for any state $\omega \in \{\omega|P(\omega) > 0\}$. This is possible only when $\tilde{P}$ assigns positive probability to the two states in $\Pi_2(\tilde{\omega})$ and 0 probability for all other states because each signal of player 2 has a distinct expectation of the securities. Given such a $\tilde{P}$, however, trader 1 always uniquely identifies the true state and has the correct expectation of the securities. Hence, the set of securities is separable with respect to Π.

4.1 Aggregation

Separability is a necessary and sufficient property for aggregation in two natural cases.

Theorem 1 (Equilibrium Aggregation, Ostrovsky [31])**.** *Consider a market with securities* $\mathbf{X}$ *and traders with signal structure* Π*. Information is aggregated in every perfect Bayesian equilibrium of this market if and only if the securities* $\mathbf{X}$ *are separable under* Π*.*

Theorem 2 (Myopic Aggregation)**.** *Consider a market with securities* $\mathbf{X}$ *and myopic traders with signal structure* Π*. Information is aggregated in finite rounds if and only if the securities* $\mathbf{X}$ *are separable under* Π*.*

Ostrovsky [31] proved a special case of Theorem 1 for markets with one security. Theorem 1 stated above accommodates any finite set of securities and is proved using a simple extension of Ostrovsky's proof. Specifically, the proof shows that traders' sequences of predictions at any perfect Bayesian equilibrium are bounded martingales and must converge. Separability implies that if information is not aggregated in the limit, there exists an agent who can make an arbitrarily large profit by deviating from his equilibrium strategy, a contradiction to traders being in equilibrium.

The proof of Theorem 2 makes use of prior work on convergence to common knowledge (particularly Geanakoplos [17]) and shows not only that myopic traders' sequences of predictions are bounded martingales but also that they must converge to the same random vector in a finite number of periods. Then, by separability, it is shown that this consensus prediction must equal $\mathrm{E}[X|\Pi(\omega^*)]$, implying aggregation. A full proof appears in the appendix of the long version of this paper, available on the authors' websites.

If the securities are not separable then there exists a distribution P satisfying (1) in the definition of separability. Letting this distribution be the prior, a perfect Bayesian equilibrium is simply for traders to report the common consensus value, not allowing any meaningful Bayesian updating and preventing aggregation from occurring. Myopic traders are constrained to report this same value.

5 SECURITY DESIGN

In this section we discuss the design of informative markets. While separability is a sufficient and necessary condition for aggregation in two natural settings, it only implies the value of the securities reflects all the traders' private information, not that the market designer can use this value to infer that private information or the likelihood of the events of interests. We define informative securities as securities that are both separable and allow for the likelihood of the events of interest to be inferred directly from their value.

As we will show, complete markets are always informative, but deployed prediction markets are rarely complete. These markets require too many securities to be practical, and their securities present challenges for traders. A prediction market for the U.S. presidential election, for example, may need one state per outcome in the electoral college. This is over 2^{50} states and requires traders to bid on securities like "The President wins Ohio, not Florida, Illinois, not Indiana ..." Even if alternative bidding methods were developed, traders would still be required to review the value of each security for aggregation to be formally implied. This is impractical, and so we consider good designs as those using a few natural securities. We first discuss the design requirements of markets that are always informative, and markets that are informative for a particular signal structure. The latter market allows a single security to be informative on any set of events, but arguably appears "unnatural." To describe the challenges of designing using only natural securities we then consider a constrained design process instead, where the market designer is restricted to an arbitrary subset of (possibly natural) securities.

5.1 Informative Securities

Informally, we would like to say that a market's securities are *informative* on a set of events with respect to a signal structure if the market organizer learns the likelihood of the events as if it knew all the traders' private signals. Assuming the values of the securities reflect traders' pooled information, if the likelihood of

the events is unambiguously implied from these values then functionally all the private signals are revealed. We call this latter property *distinguishability*.

Definition 3 (Distinguishability)**.** *Let* Π *be a signal structure over states* Ω *and* $P_{\text{join}(\Pi)}$ *be the set of all probability distributions over* Ω *that assign positive probability only to a subset of states in one element of* join(Π) *(i.e., a trader's possible posteriors after aggregation). A set of securities* $\mathbf{X}$ *on* Ω distinguishes *a set of events* $\mathcal{E}$ *with respect to* Π *if and only if for any* $P, P' \in P_{\text{join}(\Pi)}, E_P[\mathbf{X}] = E_{P'}[\mathbf{X}]$ *implies* $P(E) = P'(E), \forall E \in \mathcal{E}$.

Equivalently a set of securities distinguishes a set of events if there exists a function from the securities' values to the likelihood of the events. When a set of securities is both separable and distinguishable we describe it as *informative*.

Definition 4 (Informativeness)**.** *A set of securities* $\mathbf{X}$ *is* informative *on a set of events* $\mathcal{E}$ *with respect to a signal structure* Π *if and only if* $\mathbf{X}$ *both distinguishes* $\mathcal{E}$ *and is separable with respect to* Π.

Informativeness is a strong condition. Even if securities are not informative it might be possible for a market designer to infer some information from the market, or for the market to be described as partially informative. Generalizing our framework to account for partial aggregation would be an interesting line of future work.

5.2 Always Informative Securities

We first address the problem of designing a set of securities that is informative on a set of events with respect to *any* signal structure. We call such securities *always informative*. These securities may be of practical interest if the market designer is unsure of the traders' signal structure; using a set of always informative securities implies aggregation will occur no matter what the true signal structure is.

A market is said to be *complete* if by trading securities, agents can freely transfer wealth across states [27]. Rigorously, consider the set of securities that contains a constant payoff security plus all of the securities offered by a market. The market is complete if and only if this set includes $|\Omega|$ linearly independent securities. The most common is a market with $|\Omega|$ Arrow-Debreu securities, each associated with a different state of the world, taking value 1 on that state and 0 everywhere else. For an overview of complete markets, see Flood [15] or Mas-Colell et al. [27].

Complete markets are theoretically appealing because they allow traders to express any information about their beliefs. We formalize this well-known idea in our framework in the following proposition.

Proposition 1. *A market over state space* Ω *with securities* $\mathbf{X}$ *is complete if and only if for all distinct probability distributions* P *and* P' *over* Ω, $E_P[\mathbf{X}] \neq E_{P'}[\mathbf{X}]$.

Proof. Let M be a matrix containing the payoffs of $\mathbf{X}$, with one row for each outcome and one column for each security. The element at row i and column j of M takes value $\mathbf{x}_j(\omega_i)$. Consider a probability distribution P represented as a row vector so, $PM = \mathrm{E}_P[\mathbf{X}]$. The system of linear equations

$$P'M = \mathrm{E}_P[\mathbf{X}] \qquad \sum_{\omega \in \Omega} P'(\omega) = 1$$

has a unique solution $P' = P$ if and only if the matrix M', which is M augmented by a column of 1s to represent the summation constraint, has rank $|\Omega|$.

If the market is complete, M' has this rank so any distinct probability distribution has distinct expectation.

Now assume $\mathrm{E}_P[\mathbf{X}] \neq \mathrm{E}_{P'}[\mathbf{X}], \forall P \neq P'$, and, for a contradiction, that the market is not complete. Then the system of equations has at least two solutions, one of which is the probability distribution P and a distinct solution Q, such that $PM = QM = \mathrm{E}_P[X]$. Let U be the uniform distribution over Ω. Then there exists $c > 0$ such that $(1-c)U + cQ$ is a probability distribution (since Q satisfies $\sum_{\omega \in \Omega} Q(\omega) = 1$). Moreover, $(1-c)U + cP$ is also a probability distribution and

$$\big((1-c)U + cP\big)M = \big((1-c)U + cQ\big)M,$$

contradicting $\mathrm{E}_P[\mathbf{X}] \neq \mathrm{E}_{P'}[\mathbf{X}], \forall P \neq P'$. Thus, the market must be complete. □

This expressiveness is a necessary and sufficient condition for the likelihood of every event to be inferred, and suggests an alternative characterization of complete markets as those markets that are always informative on every event.

Theorem 3. *A market is always informative on every event* E *with respect to every signal structure* Π *if and only if it is complete.*

Proof. Distinguishing every event E is equivalent to distinguishing each state of the world $\omega \in \Omega$; the latter are also events and so must be distinguished, and if each is distinguished then the likelihood of any event can be inferred. Proposition 1 shows that completeness is a necessary and sufficient condition for distinguishing each state of the world.

It remains to show that complete markets are also separable with respect to any signal structure Π. Assume,

for a contradiction, there exists a signal structure Π and a complete market with securities $\mathbf{X}$ such that $\mathbf{X}$ is non-separable with respect to Π. Since $\mathbf{X}$ is non-separable there must exist distinct probability distributions P and P' over Ω such that $\mathrm{E}_P[\mathbf{X}] = \mathrm{E}_{P'}[\mathbf{X}]$; but by Proposition 1, in a complete market this equality only holds if $P = P'$, a contradiction. So complete markets are separable with respect to any signal structure and always distinguish every event, implying they are always informative on every event. □

Complete markets are often impractical, but rarely is every event of interest. Even if a single event is of interest, however, as many securities as almost half the states in the market may be required to create an always informative market. We let $\bar{E}$ denote the complement of E.

Theorem 4. *Any market that is always informative on an event E must have at least* $\min(|E|, |\bar{E}|) - 1$ *linearly independent securities.*

Proof. Let $\mathbf{X}$ be a set of securities, fewer than $\min(|E|, |\bar{E}|)) - 1$ of which are linearly independent, and assume, for a contradiction, that $\mathbf{X}$ is always informative on E. Restricting attention to states in E, the argument from Proposition 1 implies this market has too few securities to distinguish every probability distribution over E and there exist probability distributions P_E and P'_E such that $E_{P_E}[\mathbf{X}] = E_{P'_E}[\mathbf{X}]$. Let the difference between these distributions be the vector $\Delta_E = P_E - P'_E$, and define vectors Δ_E^+ and Δ_E^- such that $\Delta_E^+(\omega) = \max(0, \Delta_E(\omega))$ and $\Delta_E^-(\omega) = \min(0, \Delta_E(\omega))$. Since Δ_E is the difference of two probability distributions with the same expected value,

$$\sum_{\omega \in E} \frac{\Delta_E^+(\omega)}{||\Delta_E^+||_1} \mathbf{X}(\omega) = \sum_{\omega \in E} \frac{-\Delta_E^-(\omega)}{||\Delta_E^-||_1} \mathbf{X}(\omega). \qquad (2)$$

That is, $\frac{\Delta_E^+(\omega)}{||\Delta_E^+||_1}$ and $\frac{-\Delta_E^-(\omega)}{||\Delta_E^-||_1}$ are disjoint probability distributions over states in E with the same expected value, and the same argument can be made, *mutatis mutandi* for two such probability distributions over states in $\bar{E}$. Let these distributions over E be Q_E and Q'_E, and the ones over $\bar{E}$ be $Q_{\bar{E}}$ and $Q'_{\bar{E}}$. Although we have been referring to these as distributions over E and $\bar{E}$ we consider them to be distributions over Ω that assign zero probability to all states not previously included in the distributions, and we will use these names to stand for both these distributions and the states they assign positive probability to to reduce notation.

Now suppose there are two traders with signal structure

$$\Pi_1 = \{\{Q_E, Q_{\bar{E}}\}, \{Q'_E, Q'_{\bar{E}}\}\}$$
$$\Pi_2 = \{\{Q_E, Q'_{\bar{E}}\}, \{Q'_E, Q_{\bar{E}}\}\}$$

and prior

$$P_0 = \frac{Q_E + Q'_E + Q_{\bar{E}} + Q'_{\bar{E}}}{4}.$$

Each trader's expectation conditional on any signal is the same since $\mathrm{E}_{Q_E}[\mathbf{X}] = \mathrm{E}_{Q'_E}[\mathbf{X}]$ and $\mathrm{E}_{Q_{\bar{E}}}[\mathbf{X}] = \mathrm{E}_{Q'_{\bar{E}}}[\mathbf{X}]$ and each signal contains one distribution over states in E and another over states in $\bar{E}$. But the join of the signal structure is $\{\{Q_E\}, \{Q'_E\}, \{Q_{\bar{E}}\}, \{Q'_{\bar{E}}\}\}$, and if $\mathbf{X}$ is separable with respect to Π the expectation conditional on any such element must also, then, be the same. This implies $E_{Q_E}[\mathbf{X}] = E_{Q_{\bar{E}}}[\mathbf{X}]$, but by construction $Q_E(E) \neq Q_{\bar{E}}(E)$, so if $\mathbf{X}$ is separable with respect to Π it does not distinguish E, contradicting our assumption that $\mathbf{X}$ is always informative on E. □

This result demonstrates the need for a market designer to allow traders to express information it finds uninteresting. It also suggests that, in practice, few markets are acquiring all of their participants' information. This is unsurprising, but we think better designs will extract more information, and that this result shows knowledge of or assumptions about the traders' signal structure may be necessary to inform those designs.

5.3 Fixed Signal Structures

If the join of the traders' signal structure is known and has singleton sets for its elements, then there exists a single security that is informative on every event.

Theorem 5. *For any signal structure Π such that* $\mathrm{join}(\Pi)$ *consists only of singleton sets there exists a security $\mathbf{x}$ that is informative on every event E with respect to Π.*

The proof uses a result from Ostrovsky [31].

Theorem 6 (Ostrovsky [31])**.** *Let Π be a signal structure such that* $\mathrm{join}(\Pi)$ *consists of singleton sets of states, and let x be a security that can be expressed as $x(\omega) = \Sigma_i f(\Pi_i(\omega))$ for an arbitrary function f mapping signals to reals. Then x is separable under Π.*

Proof of Theorem 5. To construct the security, first assign a unique identifier $s_0, s_1, s_2, \ldots$ to every signal of every trader, and define $f(s_j) = 10^j$ for all j. Let S_ω denote the set of indices of the identifiers corresponding to the signals of each trader for state ω, i.e, corresponding to $\Pi_i(\omega)$ for each trader i. The security

$x(\omega) = \Sigma_{j \in S_\omega} f(s_j)$ is separable by Theorem 6. Additionally, the sum $\Sigma_{j \in J} f(s_j)$ for any $J \subset \{0, 1, 2, \ldots\}$ is unique, and each state ω has a unique associated set of signals since we assumed the join consists of singletons. This implies the value of the security for each element of the join is unique, so the security also distinguishes every event. □

The assumption that the join of traders' signal structure consists only of singleton sets is not without loss of generality. If the signal structure is known, however, the market designer can treat elements of the join as states of the world, identify the correct element of the join by running the market with a single security, then apply the prior to that element to learn the likelihood of each state as if he knew all the traders' private signals. If the prior is unknown this distribution can also be solicited from any single trader using a scoring rule.

5.4 Constrained Design

A single security acting as a summary statistic for an entire market is unlikely to be considered natural by any criterion. Real markets, like those on Intrade, use multiple securities. Instead of imposing our own definition of natural, in this section we consider adding a design constraint that the market's securities must be picked from a predefined set. The market designer is then challenged to find the fewest securities from this set that are informative on the events of interest with respect to the given signal structure. We call this the INFORMATIVE SET optimization problem. If the set of predefined securities is empty or has no informative subset then the problem is simply infeasible, so we assume there exists at least one such subset.

Demonstrating INFORMATIVE SET is hard would not be very interesting if exotic and unnatural securities were required for the proof. One commonly used class of securities are *event securities* which pay \$1 if an event occurs and \$0 otherwise. The corresponding optimization problem is INFORMATIVE EVENT SET, a restriction of INFORMATIVE SET, and even solving this restricted version of the problem is NP-hard.

Theorem 7. INFORMATIVE EVENT SET *is* NP*-hard.*

This immediately implies that the more general INFORMATIVE SET problem is also hard.

Corollary 1. INFORMATIVE SET *is* NP*-hard.*

The proof appears in the appendix and demonstrates a one-to-one correspondence between set cover instances and a minimal informative set of securities for a single fully informed trader.

The complexity of these problems suggests that while knowledge of the traders' signal structure allows for better designs, a perfect design will be intractable to compute or require additional assumptions about the relationship between traders' signal structure and the set of possible securities. Practically we can only ever hope to offer better (but not perfect) designs that extract more information from traders than current markets do. These results confirm we will always have to settle for some degree of error in our designs even if the traders' signal structure could be perfectly observed.

6 CONCLUSION

We developed a formal framework for the design of informative prediction markets. These markets reveal the posterior probabilities of a set of events of interest as accurately as if the traders had directly revealed their information, a commonly cited goal of prediction markets. These markets require that traders have an incentive to be accurate, that they can aggregate their information, and that the market designer can use this information to infer the likelihood of the events of interest.

Ideally informative markets would use a few natural securities. Complete markets, usually too large to be used in practice, are, however, the only markets to always be informative, regardless of the traders' signal structure. When the signal structure is known, a single security can be informative, but this security may appear strange and unintuitive to traders. Finding the smallest informative set of natural securities is computationally hard in general.

Real-world prediction markets do typically offer small numbers of simple and natural securities, and have been shown to aggregate information effectively in practice. This is not in contrast to our results, which only consider whether a market reveals *all* of the traders' private information. Our results demonstrate, however, the importance of security design and suggest that better designs that extract *more* of the traders' information are possible.

We hope this paper will allow future research on *partial* aggregation, like that which occurs in practice. Future work might also consider the effect of alternative communication channels outside the market's securities, like comments on Intrade, that allow traders to explain the reasoning behind their predictions. The development of this line of research is crucial to understanding why prediction markets work and how to make them work better.

References

[1] J. Abernethy and R. Frongillo. A collaborative mechanism for crowdsourcing prediction problems. In *Neural Information Processing Systems*, 2011.

[2] J. Abernethy and R. Frongillo. A characterization of scoring rules for linear properties. In *Proceedings of the 25th Conference on Learning Theory*, 2012.

[3] J. Abernethy, Y. Chen, and J. W. Vaughan. An optimization-based framework for automated market-making. In *Proceedings of the Twelfth ACM Conference on Electronic Commerce*, 2011.

[4] B. Allen. Strict rational expectation equilibria with diffuseness. *Journal of Economic Theory*, 27 (1):20–46, 1982.

[5] R. J. Aumann. Agreeing to disagree. *The Annals of Statistics*, 4(6):1236–1239, 1976.

[6] J. E. Berg, R. Forsythe, F. D. Nelson, and T. A. Rietz. Results from a dozen years of election futures markets research. In C. A. Plott and V. Smith, editors, *Handbook of Experimental Economic Results*. 2001.

[7] G. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

[8] Y. Chen and J. W. Vaughan. A new understanding of prediction markets via no-regret learning. In *Proceedings of the Eleventh ACM Conference on Electronic Commerce*, 2010.

[9] Y. Chen, L. Fortnow, N. Lambert, D. M. Pennock, and J. Wortman. Complexity of combinatorial market makers. In *Proceedings of the Nineth ACM Conference on Electronic Commerce*, 2008.

[10] Y. Chen, S. Dimitrov, R. Sami, D. M. Reeves, D. M. Pennock, R. D. Hanson, L. Fortnow, and R. Gonen. Gaming prediction markets: Equilibrium strategies with a market maker. *Algorithmica*, 58(4):930–969, 2010.

[11] B. Cowgill, J. Wolfers, and E. Zitzewitz. Using prediction markets to track information flows: Evidence from Google. Working paper, 2008.

[12] P. DeMarzo and C. Skiadas. Aggregation, determinacy, and informational efficiency. *Journal of Economic Theory*, 80:123–152, 1998.

[13] P. DeMarzo and C. Skiadas. On the uniqueness of fully informative rational expectations equilibria. *Economic Theory*, 13:1–24, 1999.

[14] J. Feigenbaum, L. Fortnow, D. M. Pennock, and R. Sami. Computation in a distributed information market. *Theoretical Computer Science*, 343 (1-2):114–132, 2005.

[15] M. D. Flood. An introduction to complete markets. *The Federal Reserve Bank of St. Louis Review*, 1991.

[16] P. H. Garthwaite, J. B. Kadane, and A. O'Hagan. Statistical methods for eliciting probability distributions. *Journal of the American Statistical Association*, 100:680–701, 2005.

[17] J. D. Geanakoplos. Common knowledge. In R. Aumann and S. Hart, editors, *Handbook of Game Theory*, volume 2, pages 1437–1496. North Holland, 1994.

[18] J. D. Geanakoplos and H. M. Polemarchakis. We can't disagree forever. *Journal of Economic Theory*, 28(1):192–200, 1982.

[19] T. Gneiting and A. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477): 359–378, 2007.

[20] J. González-Díaz and M. A. Meléndez-Jiménez. On the notion of perfect bayesian equilibrium. *TOP*, November 2011.

[21] S. J. Grossman. An introduction to the theory of rational expectations under asymmetric information. *Review of Economic Studies*, 48(4):541–559, 1981.

[22] R. Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):105–119, 2003.

[23] R. Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):3–15, 2007.

[24] K. Iyer, R. Johari, and C. C. Moallemi. Information aggregation in smooth markets. In *Proceedings of the Eleventh ACM Conference on Electronic Commerce*, pages 199–205, 2010.

[25] A. S. Kyle. Continuous auctions and insider trading. *Econometrica*, 53(6):1315–1336, 1985.

[26] N. Lambert. Elicitation and evaluation of statistical forecasts. Working Paper, 2011.

[27] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomics Theory*. Oxford University Press, New York, NY, USA, 1995.

[28] R. D. McKelvey and T. Page. Common knowledge, consensus, and aggregate information. *Econometrica*, 54(1):109–127, 1986.

[29] L. T. Nielsen. Common knowledge, communication and convergence of beliefs. *Mathematical Social Sciences*, 8:1–14, 1984.

[30] L. T. Nielsen, A. Brandenburger, J. Geanakoplos, R. McKelvey, and T. Page. Common knowledge

of an aggregate of expectations. *Econometrica*, 58 (5):1235–1238, 1990.

[31] M. Ostrovsky. Information aggregation in dynamic markets with strategic traders. *Econometrica*, 2012. To Appear.

[32] D. M. Pennock and M. P. Wellman. Representing aggregate belief through the competitive equilibrium of a securities market. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997.

[33] D. M. Pennock and M. P. Wellman. Compact securities markets for pareto optimal reallocation of risk. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

[34] D. M. Pennock, S. Lawrence, C. L. Giles, and F. A. Nielsen. The real power of artificial markets. *Science*, 291:987–988, 2002.

[35] C. R. Plott and S. Sunder. Efficiency of experimental security markets with insider information: An application of rational expectations models. *Journal of Political Economy*, 90(4):663–98, 1982.

[36] C. R. Plott and S. Sunder. Rational expectations and the aggregation of diverse information in laboratory security markets. *Econometrica*, 56(5): 1085–1118, 1988.

[37] C. R. Plott, J. Wit, and W. C. Yang. Parimutuel betting markets as information aggregation devices: Experimental results. Technical Report Social Science Working Paper 986, California Institute of Technology, Apr. 1997.

[38] P. M. Polgreen, F. D. Nelson, and G. R. Neumann. Using prediction markets to forecast trends in infectious diseases. *Clinical Infectious Diseases*, 44(2):272–279, 2007.

[39] R. Radner. Rational expectations equilibrium: Generic existence and the information revealed by prices. *Econometrica*, 47(3):655–678, 1979.

[40] L. J. Savage. Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 66(336):783–801, 1971.

[41] L. Shapley and M. Shubik. Trading using one commodity as a means of payment. *Journal of Political Economy*, 85:937–968, 1977.

[42] M. Spann and B. Skiera. Internet-based virtual stock markets for business forecasting. *Management Science*, 49(10):1310–1326, 2003.

[43] J. Wolfers and E. Zitzewitz. Interpreting prediction market prices as probabilities. Technical report, NBER Working Paper No. 10359, 2005.

[44] L. Xia and D. M. Pennock. An efficient montecarlo algorithm for pricing combinatorial prediction markets for tournaments. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2011.

Lifted Relational Variational Inference

Jaesik Choi and **Eyal Amir**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61874, USA
{jaesik,eyal}@illinois.edu

Abstract

Hybrid continuous-discrete models naturally represent many real-world applications in robotics, finance, and environmental engineering. Inference with large-scale models is challenging because relational structures deteriorate rapidly during inference with observations. The main contribution of this paper is an efficient relational variational inference algorithm that factors large-scale probability models into simpler variational models, composed of mixtures of iid (Bernoulli) random variables. The algorithm takes probability relational models of large-scale hybrid systems and converts them to a close-to-optimal variational models. Then, it efficiently calculates marginal probabilities on the variational models by using a latent (or lifted) variable elimination or a lifted stochastic sampling. This inference is unique because it maintains the relational structure upon individual observations and during inference steps.

1 Introduction

Many real-world systems can be described using continuous and discrete variables with relations among them. Such examples include measurements in environmental sensor networks, localizations in robotics, and economic forecasting in finance. In such large systems, efficient and precise inference is essential. As an example from environmental engineering, an inference algorithm can predict a posterior of unobserved groundwater levels and contamination levels at different locations, and making such an inference precisely is critical to decision makers.

Real-world systems have large numbers of variables including both discrete and continuous. Probabilistic first order languages, e.g. [3; 15; 24; 25; 14; 26; 28], describe probability distributions at a relational level with the purpose of capturing the structure of larger models. A key challenge of inference procedures with the languages is that they often result in intermediate density functions involving many random variables and complex relationship among them.

Lifted inference presently can address discrete models and continuous models, but not hybrid ones. For (d-valued) discrete variables, lifted inference can take an advantage of the insight which groups equivalent models into histogram representations with an order of $poly(d)$ entries [12; 22; 17] (instead of $exp(d)$ entries in traditional *ground* models). For Gaussian potentials, lifted inference can use an insight which enables maintaining compact covariance matrices during (and after) inference, e.g. [5; 7; 1].

Nonparametric variational models, e.g. NP-BLOG [4] and Latent Tree Models [32; 8], handle inference problems for discrete models and continuous Gaussian models. Here, our model and algorithms provide a solution for general (non-Gaussian) hybrid models.

In this paper, we first define **Relational Hybrid Models** (Section 2) and variational models (Section 3). We presents pragmatic algorithms (Section 4, 5 and 6) based on a new insight, **relational variational-inference lemmas** (Section 7), which accurately factors densities of relational models into mixtures of iid random variables. These lemmas enable us to build a variational approximation algorithm, which takes large-scale graphical models with hybrid variables and finds close to optimal relational variational models. Then, our inference algorithms, a variable elimination and a stochastic sampling, efficiently solve marginal inference problems on the variational models. We review the literature of statistical relational models and variational inference (Section 8). We show that the algorithm gives a better solution than solutions with existing methods (Section 9).

2 Relational Hybrid Models (RHMs)[1]

A **factor** $f = (A_f, \phi_f)$ is a pair, composed of a tuple of random variables (**rvs**) A_f and a potential function ϕ_f. Here, ϕ_f is an unnormalized probability density from the range of A_f to the nonnegative real numbers. The range of a rv can be discrete or continuous, i.e., hybrid domains. Given a **valuation v** of rvs, the **potential** of f on **v** is $\phi_f(v)$.

We define **parameterized (indexed) rvs** by using **predicates** those are functions mapping parameter values to rvs. A **relational atom** (or just **atom**) denotes a parametrized rv with free parameter variable(s). For example, an atom $\mathbf{X}(\mathbf{a})$ can be mapped to one of n rvs $\{X(a_1), \cdots, X(a_n)\}$ when the free parameter variable **a** is **substituted** by a value a_i.

A **parfactor** $g = [L, A_g, \phi_g]$ is a tuple composed of a set of parameters L, a tuple of relational atoms A_g and a potential function ϕ_g. A substitution θ is an assignment to L, and $A_g\theta$ the relational atom (possibly ground) resulting from replacing the logical variables by their values in θ. $gr(g)$ is a set of factors derived from the parfactor g by substitutions.

Following example is a parfactor:

$$[\ \underbrace{(\mathbf{a}, \mathbf{b})}_{\text{Parameter variables}},\ \underbrace{(\mathbf{X}(\mathbf{a}), \mathbf{Y}(\mathbf{b}))}_{\text{Relational atoms}}, \underbrace{f_{\mathcal{N}}(\mathbf{X}(\mathbf{a}) - \mathbf{Y}(\mathbf{b}); \mu, \sigma^2)}_{\text{A potential (linear Gaussian)}}]. \quad (1)$$

The domains of the parameter variables (**a** and **b**) are $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_m\}$. Thus, any substitution (e.g. $\mathbf{a} = a_i, \mathbf{b} = b_j$) let two rvs (e.g. $\mathbf{X}(\mathbf{a_i})$, $\mathbf{Y}(\mathbf{b_j})$) hold the linear Gaussian relationship $f_{\mathcal{N}}$.

A **Relational Hybrid Model (RHM)** is a compact representation of graphical models with discrete and continuous rvs. An RHM is composed of a **domain**, the set of possible parameter values, and a set of parfactors **G**. The joint probability of an RHM G on a valuation v of rvs is as follows:

$$\frac{1}{z} \prod_{g \in \mathbf{G}} \prod_{f \in gr(g)} \phi_f(\mathbf{v})$$

where z is the normalizing constant.

This representation is rather straightforward. However, inference procedures often result in complex models. For example, eliminating $X(a_1)$ in Equation (1) makes all other rvs fully connected. To address this problem, we focus on an important property of RHMs such that ground rvs mapped from a relational atom are **exchangeable**, defined as follows:

[1]Parts of our model representations in this section are based on the previous works ([26; 12; 23; 6]).

Definition (Exchangeable Random Variables). A sequence of rvs $X(a_1), \cdots, X(a_n)$ is **exchangeable**, when for any finite permutation $\pi()$ of the indices the joint probability of the permuted sequence $X(a_{\pi(1)})$, $\cdots, X(a_{\pi(n)})$ is the same as the joint probability of the original sequence.

Note that RHMs may include atoms with non-exchangeable rvs.[2] In this case, our variational algorithm grounds, or shatters, any atom including non-exchangeable rvs. That is, the atom degenerates into a set of propositional rvs. The detail conditions to determine exchangeable rvs, see [26; 21; 4; 12].

For convenience, we use $\mathbf{X}^n$ to refer to the set of n rvs, which are mapped from a relational atom $X(\mathbf{a})$ by substitutions, i.e., $\mathbf{X}^n = \{X(a_1), \cdots, X(a_n)\}$. The joint probability of the rvs mapped from two atoms $\mathbf{X}(\mathbf{a})$ $\mathbf{Y}(\mathbf{b})$ can be represented as follows: $P(\mathbf{X}^n, \mathbf{Y}^m) = P(X(a_1), \cdots, X(a_n), Y(b_1), \cdots, Y(b_m))$.

Potentials with a large number of rvs in RHMs introduce several difficulties in representation, learning and inference. To address these difficulties, we propose a model-factorization based on a variational method and de Finetti's theorem [11].

3 Background: Variational Inference

Variational methods are used to convert a complex problem into a simpler problem, where the simpler problem is charaterized by a decoupling of the degrees of freedom in the original problem [19]. As an example of *ground* models, one variational representation of model $P(x_1, \cdots, x_n)$ can be as follows:

$$P(x_1, \cdots, x_n) \approx \sum_{\theta} \prod_{i} P(x_i, |\theta) P(\theta)$$

where θ is a latent random variable. Inference is a procedure which computes marginal probabilities, say $P(X')$ $(X' \in \{x_1, \cdots, x_n\})$. This can be done by eliminating (or summing out) all variables except ones in X'. The variational representation (right-hand side) allows efficient inference procedures because all random variables are factorized.

De Finetti-Hewitt-Savage's Theorem: For exchangeable rvs, de Finetti's theorem [11] showed that any probability distribution $P(\mathbf{X}^n)$ of an infinite number n of binary exchangeable rvs can be represented by a mixture of independent and identically distributed

[2]Lifted inference for RHMs with non-exchangeable rvs is out of scope of this paper. There are recent developments to handle some of such cases, e.g. [17; 2].

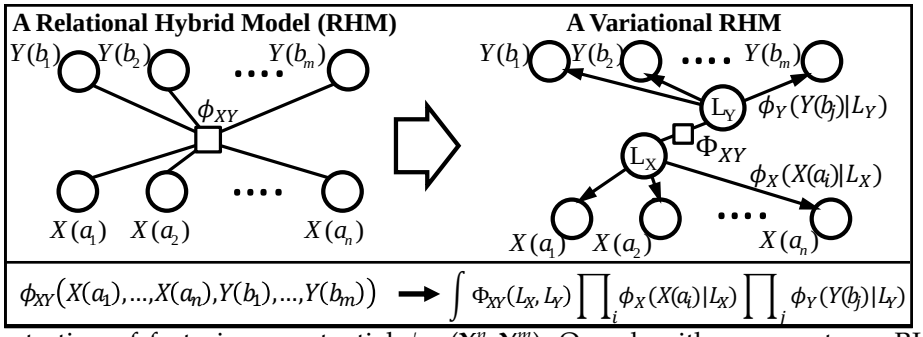


Figure 1: An illustration of factoring a potential $\phi_{XY}(\mathbf{X}^n, \mathbf{Y}^m)$. Our algorithm converts an RHM (left) into a variational (or factored) RHM (right) where the probability is represented by two latent variables L_X and L_Y.

(**iid**) Bernoulli rvs $P_{Miid}(\mathbf{X}^n)$ with a parameter θ:

$$\lim_{n\to\infty} P(\mathbf{X}^n) = \int_0^1 \theta^{t_n}(1-\theta)^{n-t_n}\Phi_X(\theta)\,\mathbf{d}\theta = P_{Miid}(\mathbf{X}^n),$$

where $t_n = \sum_i X(a_i)$, and $0 \leq \theta \leq 1$ is a **latent variable**. This observation is extended to multi-valued and continuous rvs by [16].

$$\lim_{n\to\infty} P(\mathbf{X}^n) = \int \prod_{i=1}^{n} \phi_X(X(a_i)|L_X)\Phi_X(\mathbf{d}L_X) = P_{Miid}(\mathbf{X}^n), \quad (2)$$

where L_X is a **latent variable** which chooses a distribution $\phi_X(\mathbf{X}(a)|L_X)$ of the iid rvs.[3] We represent the variational form as $P_{Miid}(\mathbf{X}^n)$ (or $\phi_{Miid}(\mathbf{X}^n)$ for unnormalized potentials).[4] Compared to the input distribution $P(\mathbf{X}^n)$, the number of parameters of the variational form, e.g. entries of conditional density tables in $P_{Miid}(\mathbf{X}^n)$, can be substantially reduced by this factorization. As shown in Figure 1, when the variational models are applied to two sets of exchangeable rvs, the variational model (the right hand side) requires parameters of $\phi_X(X(a_i)|L_X)$, $\phi_Y(Y(b_j)|L_Y)$ and $\Phi_{XY}(L_X, L_Y)$ not parameters of $\phi_{XY}(X(a_i), \cdots, Y(b_m))$.

We present variational representations for multiple sets of finite, exchangeable rvs, and new error anaysis (Section 7). Beforehand, we introduce our variational learning and inference algorithms.

4 Algorithms: Lifted Variational Inference

This section outlines our pragmatic variational inference algorithm, ***Lifted Relational Variational Inference*** **(LRVI)**. LRVI is composed of two main subroutines: learning a variational approximation, ***Find-Variational-RHM***; and eliminating latent variables (inference) ***Latent-Variable-Elimination*** which can be replaced by ***Lifted-MCMC*** (Section 6.3).

[3]Here, we consider Φ_X with a density. Thus $\Phi_X(\mathbf{d}L_X)$ in Equation (2) can be replaced by $\Phi(L_X)\,\mathbf{d}L_X$.

[4]Here, *Miid* stands for a mixture of iid rvs.

Input: G an *RHM* (a set of parfactors), Q a query (a set of relational atoms), O observations
Output: $P(Q)$ a (posterior) distribution of Q
begin
  // (One-time) Variational Learning
  if $G \notin$ {*Variational RHMs*} **then**
    $G \leftarrow$ **Find-Variational-RHM**(G);
  // Main Inference Routine
  $P(Q) \leftarrow$ **Latent-Variable-Elimination** (G, Q, O);
  return $P(Q)$;

Algorithm Lifted Relational Variational Inference: It receives an RHM, a query and observations, then returns a posterior of the query.

LRVI receives an RHM G, a query Q and observations O as inputs. It outputs the conditional probability, $P(Q|O)$. In the routine, it examines that each potential in G is the variational form, a mixture of product of iid rvs. If not, it calls *Find-Variational-RHM(G)* and receives a variational RHM G_{Miid}. The variational RHM is calculated once, and reused next time. With the G_{Miid}, *Latent-Variable-Elimination*(G_{Miid}, Q, O) solves the inference problem $P(Q|O)$. This is done by the variable elimination which iteratively eliminates non-query atoms.

Input: G, an RHM
Output: G_{Miid}, a variational RHM
begin
  for $g = (L, A, \phi) \in G$ **do**
    if *A has no continuous atom* **then**
      $\phi_{Miid} \leftarrow$ **Lifting-Discrete(**ϕ**)** (Section 5.1);
    else
      $\phi_{Miid} \leftarrow$ **Lifting-Continuous(**ϕ**)** (Section 5.2);
    $G_{Miid} \leftarrow G_{Miid} \bigcup \{(L, A, \phi_{Miid})\}$;
  return G_{Miid};

Algorithm Find-Variational-RHM: It finds a variational approximation for an input RHM (Section 5).

Find-Variational-RHM(G) converts the potential ϕ in each parfactor into a variational potential ϕ_{Miid}, a mixture of iid rvs as shown in Equation (2). For potentials with only discrete atoms, it calls *Lifting-Discrete(ϕ)* and receives a variational potential ϕ_{Miid}. For potentials including at least one continuous atom, it calls *Lifting-Continuous(ϕ)*. After iterating and converting all par-

factors in G, a variational RHM G_{Miid} is returned. Section 5 explains the procedures in detail.

Input: G_{Miid} a variational *RHM*, Q a query, O observations
Output: $P(Q)$ a (posterior) distribution of Q
begin
  $G_{Miid} \leftarrow$ Update-Obs(G_{Miid}, O) ; // For observations
  $\mathbf{A} \leftarrow$ a set of atoms in G_{Miid}; $\mathbf{\Phi} \leftarrow \{\phi_g | g \in G_{Miid}\}$;
  for $X \in \mathbf{A} \setminus Q$ **do**
    $\Phi_X \leftarrow \{\phi \in \mathbf{\Phi} | X$ is argument of $\phi\}$;
    if X *is discrete* **then**
      $\phi' \leftarrow$ **Inference-Discrete(**Φ_X**)** (Section 6.1);
    else
      $\phi' \leftarrow$ **Inference-Continuous(**Φ_X**)** (Section 6.2);
    $\mathbf{\Phi} \leftarrow (\mathbf{\Phi} \setminus \phi_X) \bigcup \{\phi'\}$;
  return $\mathbf{\Phi}$;

Algorithm Latent-Variable-Elimination: It sums out non-query latent variables, and returns a posterior of the query (Section 6).

Latent-Variable-Elimination(G_{Miid}, Q, O) call *Update-Obs*(G_{Miid}, O) to update the potentials of latent variables based on observations O. The intuition of *Update-Obs* is that each rv is conditionally independent given latent variables like the Naive Bayes models [18]. Variational RHMs allows a simple update algorithm to maintain the relational structure upon individual observations.[5] It iteratively eliminates all latent variables except the query without referring to ground variables. Section 6 includes detail procedures.

5 Variational Learning for RHMs

This section elaborates a learning algorithm which converts each potential in an RHM into a variational potential. Here, the key procedure is to extract the potential on latent variables (e.g. $\Phi_X(L_X)$).

The potential on latent variables can be derived analytically and exactly, when an input potential satisfies some conditions such as ∞-extendible (explained in Section 7). It is also known that discrete potentials[6] and some Gaussian potentials (e.g. pairwise Gaussian [5] and Gaussian processes [9; 31]) allow such derivations.

Unfortunately, it is hard to use such derivations in general hybrid models because many real-world potentials are neither ∞-extendible nor Gaussian. Here, we present our solutions for (more intuitive) discrete models first, and then for continuous models.

5.1 Lifting Discrete Potentials

For discrete potentials, we need to find the probability density $\Phi_X(L_X)$ over the iid (Bernoulli) rvs where L_X is the Bernoulli parameter. To represent an input potential $\phi(\mathbf{X}^n)$ compactly, we group equivalent value assignments according to the **value-histogram representation** [12; 23], $\phi(\mathbf{X}^n) = \phi_{\mathbf{h}}(h_X)$.[7] Learning variational parameters of Equation (2) with discrete rvs is formulated as follows:

$$\begin{aligned}
&\arg\max_{\Phi_X(p)} \left\| \phi(\mathbf{X}^n) - \int \Phi_X(p) \prod_{i=1}^{n} \phi_X(X(a_i)|p)\, dp \right\| \\
&= \arg\max_{\Phi_X(p)} \left\| \phi_{\mathbf{h}}(h_X) - \int \Phi_X(p) \cdot f_{\mathcal{B}/\mathcal{M}}(h_X; n, p)\, \mathbf{d}p \right\| \\
&\approx \arg\max_{\mathbf{w},\mathbf{p_X}} \left\| \phi_{\mathbf{h}}(h_X) - \sum_{l=1}^{k} w_l \cdot f_{\mathcal{B}/\mathcal{M}}(h_X; n, p_{X_l}) \right\|, \qquad (3)
\end{aligned}$$

where $||P - Q||$ (or $d_{TV}(P, Q)$) is the **total variation distance**[8]; $f_{\mathcal{B}/\mathcal{M}}(h_X; n, p)$ is a binomial (or multinomial) pdf; $\mathbf{w} = (w_1, \cdots, w_k)$ is a k-dimensional weight vector such that w_l=$\Phi_X(p_{X_l})$, $\sum_{l=1}^{k} w_l$=1; and $p_{\mathbf{X}} = (p_{X_1}, \cdots, p_{X_k})$ is a vector of k values chosen from $[0, 1]$, the domain of the latent variable p.

For **binary** exchangeable rvs, the iid potential $\phi_X(X(a_i)|p)$ is the Bernoulli distribution with a parameter p (i.e. $P(X(a_i)) = p$). When equivalent models in $\mathbf{X}^n$ are grouped into the histogram h_X, the variational terms can be represented as a binomial distribution (the second line in Equation (3)) because $\binom{n}{h_X} \prod_i \phi_X(X(a_i)|p) = f_{\mathcal{B}}(h_X; n, p)$. That is, the problem is reduced to learn a mixture of k binomial distributions where w_l is a weight for each binomial $f_{\mathcal{B}}(h_X; n, p_{X_l})$.

For **multi-valued** exchangeable rvs, the iid potential $\phi_X(X(a_i)|p)$ is the Categorical distribution, i.e. multi-valued Bernoulli. Thus, this problem is reduced to learn a mixture of multinomial distributions $f_{\mathcal{M}}$:

$$\arg\max_{\mathbf{w},\mathbf{p_X}} \left\| \phi_{\mathbf{h}}(h_X) - \sum_{l=1}^{k} w_l \cdot f_{\mathcal{M}}(h_X; n, p_{X_l}) \right\|. \qquad (4)$$

For potentials with two or more atoms, it can be formulated as follows:

$$\arg\max_{\mathbf{w},\mathbf{p_X},\mathbf{p_Y}} \left\| \phi_{\mathbf{h}}(h_X, h_Y) - \sum_{l=1}^{k} w_l f_{\mathcal{B}/\mathcal{M}}(h_X; n, p_{X_l}) f_{\mathcal{B}/\mathcal{M}}(h_Y; m, p_{Y_l}) \right\|,$$

where $p_{\mathbf{Y}}$ is a k-dimensional vector, $(p_{Y_1}, \cdots, p_{Y_k})$; and f is either the binomial or the multinomial depending on the domain of rvs.

We learn a mixture of binomials (or multinomials), e.g. $(\mathbf{w}, p_{\mathbf{X}})$ in Equation (3), from the original potential $\phi(\mathbf{X}^n)$ using an incremental EM algorithm.[9] Because the k is not known or given, the incremental EM algorithm increases k up to n until the variational error

[5] Existing lifted inference methods degenerate relational models upon observations. See Split [26] or Shatter [12].

[6] Section 8 includes an empirical comparison with existing lifted inference for discrete models

[7] h_X is a vector with $h_{Xv} = |\{i : X(a_i) = v\}|$.

[8] $||P-Q|| = \sup_{A \in \mathcal{B}}(P(A)-Q(A))$ when $\mathcal{B}$ is a class of Borel sets.

[9] EM algorithms are common to learn parameters for mixture models [30; 10; 27].

converges. Assuming that the EM algorithm interates up to a constant times given a fixed k, the computational complexity of the incremental EM algorithm for n binary, exchangeable rvs is bounded by $O(n^3)(=\sum_{k=1}^{n} c \cdot O(kn))$. Note that, the algorithm increases k from 1 to n. In each EM step, k components visit n histogram entries $O(kn)$.

5.2 Lifting Continuous and Hybrid Potentials

For a potential $\phi(\mathbf{X}^n)$ with continuous rvs, we use a mixture of non-parametric densities to represent variational potentials. Here, we generate samples from the input potential, then learn parameters for the mixture of non-parametric densities.

Equation (2) is used to formulate the learning problem as follows:

$$\arg\max_{\Phi_X(L_X)} \left\| \phi(\mathbf{X}^n) - \int \Phi_X(L_X) \cdot \prod_{i=1}^{n} \hat{f}_{L_X}(X(a_i))\, \mathbf{d}L_X \right\| \quad (5)$$

$$\approx \arg\max_{\mathbf{w}, \hat{f}_{\mathbf{X}}} \left\| \phi(\mathbf{X}^n) - \sum_{u=1}^{k} w_u \cdot \prod_{i=1}^{n} \hat{f}_{X_l}(X(a_i)) \right\|, \quad (6)$$

where $\hat{f}_{L_X}$ and $\hat{f}_{X_l}$ refer to (non-parametric) probability distributions. To solve the optimization problem, we generate N samples $\mathbf{v_1}, \cdots, \mathbf{v_N}$ from the input potential $\phi(\mathbf{X}^n)$ where $\mathbf{v_t} = (v_{t_1}, \cdots, v_{t_n})$ is a n-dimensional vector, value assignments for n rvs. Then, we solve the following maximum likelihood estimation (MLE) problem: $\arg\max_{\mathbf{w}, \hat{f}_{\mathbf{X}}} \sum_{t=1}^{N} \log\left(\sum_{l=1}^{k} w_l \cdot \prod_{i=1}^{n} \hat{f}_{X_l}(v_{t_i})\right)$, where we denote the kernel density estimator by $\hat{f}_{X_l}(x) = \frac{1}{S\sigma^2} \sum_{i=1}^{S} K\left(\frac{x-\mu_i}{\sigma^2}\right)$ where $(\mu_1, \cdots, \mu_S)$ are S data points that underlie the density, and σ^2 is a parameter. For simplicity, we use the Gaussian Kernel, $K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.

It is interesting to note that the kernel density estimator is analogous to the value-histogram for discrete rvs [12; 23] in a sense that frequently observed regions (or bins) have the higher probability. This new insight enables us to represent continuous models compactly.

For potentials with two or more atoms, the approach can be formulated as follows:

$$\arg\max_{\mathbf{w}, \hat{f}_X, \hat{f}_Y} \sum_{t=1}^{N} \log\left(\sum_{l=1}^{k} w_l \cdot \prod_{i=1}^{n} \hat{f}_{X_l}(v_{t_i}^X) \cdot \prod_{j=1}^{m} \hat{f}_{Y_l}(v_{t_j}^Y) \right),$$

where v_t^X and v_t^Y are respectively the t^{th} samples of $\mathbf{X}^n$ and $\mathbf{Y}^m$.

This MLE problem is also solved by an EM algorithm. N samples are used to build k densities in the maximization (M) step, and the likelihood of each sample is calculated in the expectation (E) step. variation error converges.

6 Lifted Inference with RHMs

In this section we build on the result of previous sections and present lifted inference algorithms that utilize the learned variational models to speed up relational inference. The lifted inference algorithms find solutions without referring to ground rvs.

Latent-Variable-Elimination marginalizes relational atoms with the following steps: (i) choosing an atom; (ii) finding all potentials including the atom and making the product of them; (iii) summing out the atom; and (iv) repeating the steps until only query atoms are left. We demonstrate the key step (iii) with two variational potentials, $\phi_{Miid}(\mathbf{X}^n, \mathbf{Y}^m)$ and $\phi'_{Miid}(\mathbf{Y}^m)$.

6.1 Inference with Discrete Variables

The key intuition is that the variational form is maintained after eliminating an atom. We demonstrate the intuition by an example. The marginal probability of the latent variable L_X is calculated by eliminating (or summing) $\mathbf{Y}^m$ out: $\sum_{h_y} \phi_{\mathbf{h}}(h_x, h_y) \cdot \phi'_{\mathbf{h}}(h_y)$

$$\begin{aligned}
&\approx \sum_{h_y} \sum_{l=1}^{k} w_l f_{\mathcal{B}}(h_x; n, p_{X_l}) f_{\mathcal{B}}(h_y; m, p_{Y_l}) \sum_{l'=1}^{k'} w_{l'} f_{\mathcal{B}}(h_y; m, p_{Y_{l'}}) \\
&= \sum_{l=1}^{k} \sum_{l'=1}^{k'} w_l w_{l'} \left(\sum_{h_y} f_{\mathcal{B}}(h_y; m, p_{Y_l}) f_{\mathcal{B}}(h_y; m, p_{Y_{l'}}) \right) f_{\mathcal{B}}(h_x; n, p_{X_l}) \\
&\approx \sum_{l=1}^{k} \sum_{l'=1}^{k'} w_l w_{l'} \left(\int f_{\mathcal{N}}(h_y; \mu_l, \sigma_l^2) f_{\mathcal{N}}(h_y; \mu_{l'}, \sigma_{l'}^2)\, \mathbf{d}h_y \right) f_{\mathcal{B}}(h_x; n, p_{X_l}) \\
&= \sum_{l=1}^{k} w_{\overline{Y},l} \cdot f_{\mathcal{B}}(h_x; n, p_{X_l}) = \phi''(\mathbf{X}^n) \qquad (7)
\end{aligned}$$

when $\sum_{l=1}^{k} w_{\overline{Y},l} = 1$ and $f_{\mathcal{N}}(h_y; \mu_l, \sigma_l^2)$ is the Normal approximation to binomial such that $\mu_l (= m \cdot p_{Y_l})$ and $\sigma_l^2 (= m \cdot p_{Y_l} \cdot (1 - p_{Y_l}))$. It is important to note that binomial pdfs are not closed under the product operation. That is, a product of two binomial pdfs are not a binomial pdf unless the binomial parameters p_{X_l} p_{X_l} are identical. For large n and m, the Normal approximation to Binomial is an important step to maintain the variational structure during the inference procedure. In this way, after eliminating $\mathbf{Y}^m$, the marginal potential $\phi''_{Miid}(\mathbf{X}^n)$ is still represented as the variational form. The same principle is applied for potentials with more than two atoms.

Now, we will show that the product of variational forms in Step (iii) can also be represented as a variational form. Consider the following two variational potentials $\phi_{Miid}(\mathbf{X}^n)$ $\phi'_{Miid}(\mathbf{X}^n)$. The product operation is common during the elimination step as shown in Equation (7). The product of the two potentials is represented as follows:

$$\left(\sum_{l=1}^{k} w_l \cdot f_{\mathcal{B}}(h_x; n, p_{X_l}) \right) \cdot \left(\sum_{l'=1}^{k'} w'_{l'} \cdot f'_{\mathcal{B}}(h_x; n, p_{X_{l'}}) \right)$$

$$\approx \sum_{l=1}^{k}\sum_{l'=1}^{k'} w_l{\cdot}w'_{l'} \int f_{\mathcal{N}}(h_x;\mu_l,\sigma_l^2){\cdot}f'_{\mathcal{N}}(h_x;\mu_{l'},\sigma_{l'}^2)\,\mathbf{d}h_x$$

$$= \sum_{l=1}^{k}\sum_{l'=1}^{k'} w_l{\cdot}w'_{l'}{\cdot}z_{l,l'}\, f_{\mathcal{N}}(h_x;\mu_{new},\sigma^2_{new}) = \phi'''_{Miid}(\mathbf{X}^n), \quad (8)$$

$z_{l,l'}$ is the inverse of the normalizing constant. This derivation shows that a product of variational potentials results in a variational potential as $\phi'''_{Miid}(\mathbf{X}^n)$.[10]

6.2 Inference with Continuous Variables

For continuous variables, we also demonstrate the intuition by an example with two potentials $\phi_{Miid}(\mathbf{X}^n,\mathbf{Y}^m)$ $\phi'_{Miid}(\mathbf{Y}^m)$ where $\mathbf{X}^n$ and $\mathbf{Y}^m$ are two sets of continuous rvs. Each potential is represented as shown in Section 5.2. When we eliminate $\mathbf{Y}^m$, it can be formulated as follows:

$$\int\left(\sum_{l=1}^{k} w_l\prod_{i=1}^{n}\hat{f}_{X_l}(X(a_i))\prod_{j=1}^{m}\hat{f}_{Y_l}(Y(b_j))\right)\phi'_{Miid}(\mathbf{Y}^m)\,\mathbf{d}\mathbf{Y}$$
$$=\sum_{l=1}^{k}\sum_{l'=1}^{k'} w_l w_{l'}\prod_{i=1}^{n}\hat{f}_{X_l}(X(a_i))\prod_{j=1}^{m}\left(\int\hat{f}_{Y_l}(Y(b_j))\hat{f}_{Y_{l'}}(Y(b_j))\,\mathbf{d}Y(b_j)\right)$$
$$=\sum_{l=1}^{k}\sum_{l'=1}^{k'} w_l w_{l'}\frac{1}{{z_{l,l'}}^m}\prod_{i=1}^{n}\hat{f}_{X_l}(X(a_i)) = \phi''_{Miid}(\mathbf{X}^n), \quad (9)$$

$z_{l,l'}$ is the normalizing constant calculated from the product of two mixtures of Normals: $\hat{f}_{Y_l}(Y(b_j))$; and $\hat{f}_{Y_{l'}}(Y(b_j))$.

Finally, we show that the product of two variational potentials becomes a variational form: $\left(\sum_{l=1}^{k} w_l{\cdot}\prod_{i=1}^{n}\hat{f}_{X_l}(X(a_i))\right){\cdot}\left(\sum_{l'=1}^{k'} w'_{l'}{\cdot}\prod_{i=1}^{n}\hat{f}_{X_{l'}}(X(a_i))\right)$

$$=\sum_{l=1}^{k}\sum_{l'=1}^{k'} w_l w'_{l'}\prod_{i=1}^{n}\hat{f}_{X_l}(X(a_i))\hat{f}_{X_{l'}}(X(a_i))$$
$$=\sum_{l=1}^{k}\sum_{l'=1}^{k'} w_l w'_{l'}\frac{1}{{z_{l,l'}}^n}\cdot\prod_{i=1}^{n}\hat{f}^{new}_{X_{l,l'}}(X(a_i)) = \phi'''_{Miid}(\mathbf{X}^n). \quad (10)$$

6.3 Lifted Markov chain Monte Carlo (MCMC)

When variational RHMs include a large number of mixture compoments. The latent variable elimination may take long time. In this case, we use a lifted MCMC algorithm: (i) choosing a latent variable (e.g L_X) randomly; (ii) calculating the conditional probability of the latent variable (e.g. $\Phi_X(L_X)$) using assignment of neighboring latent variables; (iii) choosing an assignment from the probability (e.g. $L_X{=}p_{X_l}$ $(1{\le}l{\le}k)$); and (iv) repeating until convergence.

[10] $\phi'''_{Miid}(\mathbf{X}^n)$ is a mixture of $|k\cdot k'|$ Normals. When $|k\cdot k'|$ is large, it is possible to collapse the mixture into a mixture of fewer components.

Here, the steps (ii) and (iii) are main steps. Step (ii) is a subset of the procedure in Equations (7) and (9), because the values of neighboring latent variables (e.g. $L_Y{=}p_{Y_{l'}}$ $(1{\le}l'{\le}k')$) can be simply assigned. Step (iii) is a straightforward sampling procedure which chooses one compoment based on its weight. For example, $w_l{\cdot}w'_{l'}{\cdot}z_{l,l'}$ in Equation (8) is a weight for one of $|k|\cdot|k'|$ Normal distributions in $\phi'''_{Miid}(\mathbf{X}^n)$.

7 Relational-Variational Lemmas

This section provides error analysis of our variational approximations for a single atom, multiple atoms and the general case (RHMs). Beforehand, we define a term, $\overline{\mathbf{n}}$**-extendible**:

Definition ($\overline{\mathbf{n}}$**-extendible).** $P(\mathbf{X}^n)$, a probability with n exchangeable rvs, is $\overline{\mathbf{n}}$**-extendible** when the followings hold: (1) there is $P(\mathbf{X}^{\overline{n}})$, a probability with $\overline{n}$ exchangeable rvs ($\overline{n} > n$); and (2) $P(\mathbf{X}^n)$ is the marginal distribution of $P(\mathbf{X}^{\overline{n}})$, i.e., eliminating $(\overline{n}-n)$ rvs.

Figure 2 explains the intuition of $\overline{\mathbf{n}}$-extendible potentials for discrete models. If a potential is not extendible, it has rugged bars, e.g. a single peak. If a potential is extendible to a large number $\overline{n}$, the potential has a smoothed histogram. If a potential is ∞-extendible, it is represented by a mixture of binomials exactly.

Lemma 1. [13] *If $P(\mathbf{X}^n)$, a probability with n exchangeable rvs, is $\overline{n}$-extendible, then the* **total variation distance** *$d_{TV}(P(\mathbf{X}^n),P_{Miid}(\mathbf{X}^n))$ of the input probability $P(\mathbf{X}^n)$ and the variational form $P_{Miid}(\mathbf{X}^n)$ in Equation* (2) *is bounded as follows: (i) when $X(a_i)$ are d-valued discrete rvs, $d_{TV}(P(\mathbf{X}^n),P_{Miid}(\mathbf{X}^n)) \le \frac{2dn}{\overline{n}}$; and (ii) when $X(a_i)$ are continuous rvs, $d_{TV}(P(\mathbf{X}^n),P_{Miid}(\mathbf{X}^n)) \le \frac{n(n-1)}{\overline{n}}$.*

7.1 Our Results: Variational RHMs

Factoring Potentials with Multiple Atoms: De Finetti-Heweitt-Savage's theorem (Section 3) is applicable only to potentials with a single atom. Here, we present our new theoretical results on variational RHMs.

Lemma 2. [Existence of a Variational Form] *For $P(\mathbf{X}^n,\mathbf{Y}^m)$, a probability with two atoms in an RHM, there are two new latent variables, L_X and L_Y, and a new potential $\Phi_{XY}(L_X,L_Y)$ such that the following holds,* $\lim_{n,m\to\infty} P(\mathbf{X}^n,\mathbf{Y}^m)$

$$=\int\Phi(L_X,L_Y)\prod_{i=1}^{n}\phi_X(X(a_i)|L_X)\prod_{j=1}^{m}\phi_Y(Y(b_j)|L_Y)\,\mathbf{d}L_X L_Y$$
$$= P_{Miid}(\mathbf{X}^n,\mathbf{Y}^m).$$

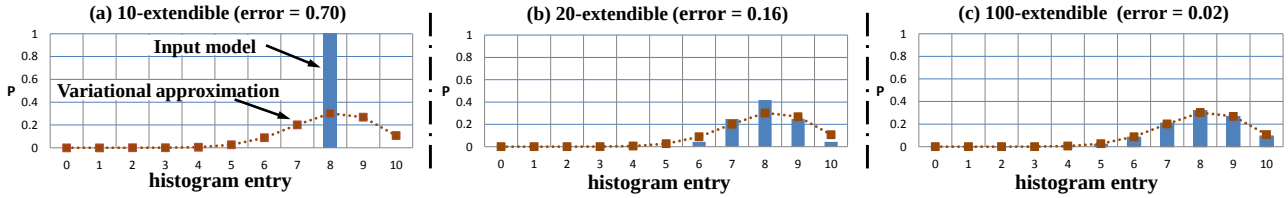


Figure 2: Illustrations of three different value-histograms of 10 exchangeable binary rvs. Dotted lines with markers represent the best possible variational approximation, i.e., a mixture of binomials. (a), (b) and (c) respectively present potentials extendible up to 10, 20 and 100 rvs. For the potential in (1), the variational approximation has a high error, total variation, and thus is a poor approximation. When a potential is extendible to the larger number, the variational approximation is smaller as shown in (2) and (3).

Sketch of proof. Assigning values to one atom is fixed (e.g. $\mathbf{Y}^m$=$\mathbf{v}$) results in a new potential with one atom, $\phi(\mathbf{X}^n)$, which can be factored into $\int \Phi(L_X|\mathbf{v}) \prod_i \phi_X(X(a_i)|L_X)\mathbf{d}L_X$. Because $\Phi(L_X|\mathbf{v})$ is conditioned on $\mathbf{v}$, $\Phi(L_X|\mathbf{Y}^m)$ can be factored into $\int \Phi(L_X, L_Y) \prod_j \phi_Y(Y(b_j)|L_Y)\mathbf{d}L_Y$. □

To analyze variational error of potentials with multiple atoms, we introduce another term $(\overline{\mathbf{n}}, \overline{\mathbf{m}})$-**extendible**.

Definition ($(\overline{\mathbf{n}}, \overline{\mathbf{m}})$**-extendible).** $P(\mathbf{X}^n, \mathbf{Y}^m)$ is $(\overline{\mathbf{n}}, \overline{\mathbf{m}})$-**extendible** when (1) there is $P(\mathbf{X}^{\overline{n}}, \mathbf{Y}^{\overline{m}})$, a probability with two sets of exchangeable rvs ($\overline{n}$>n, $\overline{m}$>m); and (2) $P(\mathbf{X}^n, \mathbf{Y}^m)$ is the marginal distribution of $P(\mathbf{X}^{\overline{\mathbf{n}}}, \mathbf{Y}^{\overline{\mathbf{m}}})$.

Lemma 3 (Error of the Variational Parfactor). *If $P(\mathbf{X}^n, \mathbf{Y}^m)$, a probability with two exchangeable rvs in an RHM, is $(\overline{n}, \overline{m})$-extendible, then the total variation $d_{TV}(P(\mathbf{X}^n, \mathbf{Y}^m), P_{Miid}(\mathbf{X}^n, \mathbf{Y}^m))$ is bounded as follows: (i) when $\mathbf{X}^n$ and $\mathbf{Y}^m$ are respectively d_x-valued and d_y-valued discrete rvs, $d_{TV}(P, P_{Miid}) \leq \frac{2d_x n}{\overline{n}} + \frac{2d_y m}{\overline{m}}$; (ii) when $\mathbf{X}^n$ are d_x-valued discrete and $\mathbf{Y}^m$ are continuous, $d_{TV}(P, P_{Miid}) \leq \frac{2d_x n}{\overline{n}} + \frac{m(m-1)}{\overline{m}}$; (iii) when $\mathbf{X}^n$ and $\mathbf{Y}^m$ are continuous, $d_{TV}(P, P_{Miid}) \leq \frac{n(n-1)}{\overline{n}} + \frac{m(m-1)}{\overline{m}}$.*

proof. The intuition is that the error of a variational model is additive with an additional atom. We used the principle in [13] such that a $\bar{n}$-extendible pdf, $P(X^n)$, can be represented by a mixture of extreme pdfs (e.g. $P(X^n) = \sum_e w_e p_e$). Here, an extreme pdf p_e is a probability of n draws made at random without replacement from an urn, U, which contains $\bar{n}$ balls marked by one of c colors. Let e a unique marking in U. The variation distance of each extreme point p_e and its variational form $P_{Miid}(X^n|e)$ is bounded $\leq \frac{2cn}{\bar{n}}$ for discrete rvs.

For a distribution with the multiple atoms, each extreme point corresponds to the joint distribution of n draws from U_X of $\bar{n}$ balls, and m draws from U_Y of $\bar{m}$ balls. Because the draws can be done independently for each urn, an extreme pdf (e.g. p_{e_x,e_y}) can be represented as the product of independent extreme pdfs (e.g. $p_{e_x} \cdot p_{e_y}$). From the Lemma 1, the variation distances of variational forms of p_{e_x} and p_{e_y} are respectively bounded. WLOG, we can represent the errors with ϵ_x and ϵ_y,

$$d_{TV}(p_{e_x}, \phi_{iidx}) \leq \epsilon_x, d_{TV}(p_{e_y}, \phi_{iidy}) \leq \epsilon_y.$$

Note that,

$$\begin{aligned} & d_{TV}(p_{e_x}p_{e_y}, \phi_{iidx}\phi_{iidy}) \\ \leq\ & d_{TV}(p_{e_x}p_{e_y}, p_{e_x}\phi_{iidy}) + d_{TV}(\phi_{iidy}p_{e_x}, \phi_{iidy}\phi_{iidx}) * \\ \leq\ & d_{TV}(p_{e_y}, \phi_{iidy}) + d_{TV}(p_{e_x}, \phi_{iidx}) * * = \epsilon_x + \epsilon_y. \end{aligned}$$

The second step (marked as $*$) is derived from the following equations (Here, A is $p_{e_x}\phi_{iidy}$):

$$\begin{aligned} p_{e_x}p_{e_y} - \phi_{iidx}\phi_{iidy} &= p_{e_x}p_{e_y} - \phi_{iidx}\phi_{iidy} - A + A \\ &= (p_{e_x}p_{e_y} - A) + (A - \phi_{iidx}\phi_{iidy}). \end{aligned}$$

The third step (marked as $**$) is derived from the fact that p_{e_x} and ϕ_{iidy} are pdfs, e.g. $\sum_{X^n} p_{e_x}(X^n) = 1$. That is, $d_{TV}(p_{e_x}p_{e_y}, p_{e_x}\phi_{iidy}) \leq d_{TV}(p_{e_y}, \phi_{iidy})$. The derivation can be applied to ϕ_{iidy} and continuous cases.

Thus, the total variation distance between two probabilities, $P(X^n, Y^m)$ and $P_{Miid}(X^n, Y^m)$ is bounded by the sum of two error bounds. □

Now, we present error analsys for RHMs with more than two atoms, e.g. $P(\mathbf{X}^n, \mathbf{Y}^m, \mathbf{Z}^u)$.

Theorem 4 (*Error of Variational RHMs*). *Let $\mathbf{X}_g$ and $\mathbf{X}_G$ is respectively the set of all rvs in a parfactor g and an RHM G. Let $P(\mathbf{X}_g)(= \frac{1}{z_g} \prod_{f \in gr(g)} w_f(\mathbf{X}_f))$; and $P(\mathbf{X}_G)(= \frac{1}{z} \prod_{g \in G} P(\mathbf{X}_g))$. The total variation $d_{TV}\left(P(\mathbf{X}_G) - P_{Miid}(\mathbf{X}_G)\right)$ is bounded by $\frac{1}{z}\sum_{g \in G} \epsilon_g$ where $\epsilon_g = d_{TV}(P(\mathbf{X}_g) - P_{Miid}(\mathbf{X}_g))$ and z is the normalizing constant.*

Sketch of proof. We can build a fully joint probability of all relational atoms with the RHM. Then, Lemma 3 can be used to prove the total variation of the variational RHM. □

8 Related Work

Nonparametric Bayesian Logic (NP-BLOG) [4] presents a new variational representation for relational discrete models using the Dirichlet Process. In principle, the NP-BLOG and the variational RHMs have in common: compact representations for exchangeable rvs. The difference is that NP-BLOG handles ∞-extendible exchangeable, discrete rvs. Here, we investigated further for several new directions: model learning, continuous domains, and approximation errors. Our error bounds provide an in-depth understanding for models with finite exchangeable discrete rvs.

For discrete models, the value-histogram [12; 23] represents potentials with polynomial numbers of histogram entries. When aggregate operators are given, [6] shows that the histgram representations can be approximately replaced by a Normal with linear constraints. We generalize and expand the concept to compress general-purpose histogram representations.

For continuous potentials, unfortunately, the histogram representation is not applicable because it is not clear how to discretize continuous domains to build up such histograms. Thus, most existing lifted inference for continuous models are limited to Gaussian potentials [5; 7; 1]. Thus, our representation is a unique lifted inference for non-Gaussian continuous potentials.

Lifted Belief Propagation (LBP) [29] solves inference problems in many practial models by grouping rvs. Here, rvs in an atom send the same messages to neighboring rvs, and are not constrained among rvs in an atom. Instead, our lifted MCMC sends a distribution, has more expressive power, and requires fewer samples until the convergence. The advance promises that our variational models and the lifted MCMC method can be a good complement to existing sampling methods such as LBP.

9 Experimental Results

We provide experimental results regarding the variational approximations and the efficiency and the accuracy of our LRVI in a real-world groundwater model.

First, we analyze variational approximations on the competing workshops models C-FOVE [23] which includes two parfactors, $\phi_1(\mathit{attends}(X), \mathit{hot}(W))$ and $\phi_2(\mathit{attends}(X), \mathit{series})$. We can exactly represent ϕ_2 with a Binomial:

$$\begin{aligned}&\phi_2(\#X[\mathit{attends}(X)], \mathit{series})\\ &\quad = \quad w(\mathit{series}) f_{\mathcal{B}}(\#X[\mathit{attends}(X)]; n, p(\mathit{series})),\end{aligned}$$

Here, n is the number of people; w and l are functions: $\{\top, \bot\} \rightarrow \mathbf{R}$; and $\#X[\mathit{attends}(X)]$ is a histogram representation in [23] such that $|\{i|\mathit{attends}(X_i) = \top\}|$. ϕ_1 is also represented by a mixture of $|W|$. ϕ_1 is the bivariate multiplicative binomial [20] and conditionally binomial. E.g., when $\#W[\mathit{hot}(W)]$ is fixed to 5, $\phi_1(\#X[\mathit{attends}(X)], \#W[\mathit{hot}(W)] = 5) = w_5 f_{\mathcal{B}}(\#X[\mathit{attends}(X)]; n, l_5)$.

When using the same parameters in [23], $\phi_1(\mathit{attends}(X), \mathit{hot}(W))$ is represented by a single binomial distribution accurately because weights of other $|W - 1|$ binomials are small with -9 orders of magnitude. We make similar observations with different parameters. When we randomly choose parameters (50 times), a single binomial was enough to represent ϕ_1 with a small total variation (< 0.001) for more than 90% (46 times). For other parameters, at most three binomials can represent ϕ_1 with a very small error (<0.0001).

Second, we apply our variational learning algorithm to a real-world groundwater dataset shown in Figure 3 (a) Republican River Compact Administration (RRCA). The dataset is composed of measurements (water levels) at over 10,000 wells and baseflow observations at 65 gages from 1918 until 2007.[11] After calibration, the training dataset is a set of partial observations in a 480 (months) by 3420 (wells) matrix. First, we cluster the 3420 wells by k-means into 10 groups based on means and variances (approximately exchangeable).[12] From the dataset, the EM algorithm directly learns a variational model until the log-likelihood converges. As a result, from 6 to 14 mixtures of Gaussians (MoGs) are learned for each cluster. Figure 3 (b) shows some learned empirical distributions, cdfs of MoGs, with high weights from two clustered area, A and B. To represent the joint distribution over the clusters, we convert the 480X3240 input matrix into a 480 (months) by 92 (MoGs) matrix.

For each test month, we compute the empirical distribution of the query variables given the partial observations. Our lifted VE returns queries in average 0.3 secs and a ground VE inference returns in average 37.9 secs. In fact, ground and variational inference algorithms use different sizes of matrices 480X3420 and 480X92, respectively. That explains the reason why the variational inference is efficient. The average total variations are 0.35 (ground) and 0.29 (variational), where smaller is more accurate.

Third, we compare the accuracy and the efficiency of

[11] Head predictions are available via the RRCA official website, http://www.republicanrivercompact.org.

[12] Although the means and variances do not guarantee the exchangeability in the clusters, here we focus on measuring the computational efficiency of our variational method.

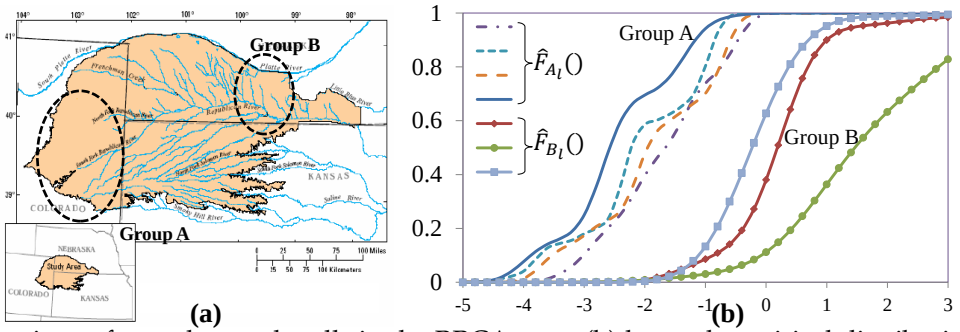


Figure 3: (a) locations of two clustered wells in the RRCA map; (b) learned empirical distributions, cdfs ($\hat{F}_{A_l}()$ and $\hat{F}_{B_l}(x)$) for two regions A and B.

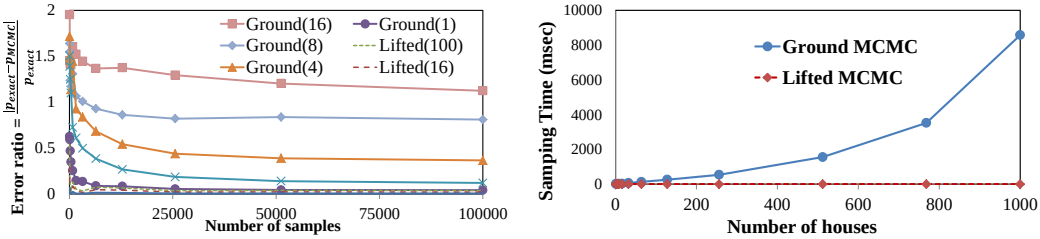


Figure 4: (a) The accuracy of the lifted MCMC and the ground MCMC with various numbers of houses. Here, () indicates the number of houses, e.g., Ground(16) is the ground MCMC with 16 houses. (b) The average time to generate samples in each MCMC step.

our lifted MCMC algorithm with a vanilla (ground) MCMC algorithm on an already factored variational model. The model is composed of two relational atoms: a binary atom $\mathbf{Job}^n$, saying whether each individual has a job, and a continuous atom $\mathbf{HP}^m$, saying the price change of each house. p_{Job} is a latent variable, which represents the Bernoulli parameters of $\phi_{Miid}(\mathbf{Job}^n)$. p_D is a latent variable (probability of market down), which represents the mixture of two Gaussians, $\phi_{Miid}(\mathbf{HP}^m)$ =

$$p_D \prod_{j=1}^{m} f_{\mathcal{N}}(HP(h_j); -0.3, \sigma_D^2) + (1-p_D) \prod_{j=1}^{m} f_{\mathcal{N}}(HP(h_j); 0.1, \sigma_{UP}^2).$$

Here, we assume that the two latent variables follows a linear Gaussian: $\Phi(p_{Job}, p_D) = f_{\mathcal{N}}(p_{Job} - p_D; 0, \sigma^2_{JobHouse})$. Figure 4 (a) shows the accuracy of the two algorithms after generating the same number of samples. That is, it measure the ratio of error to estimate a probability of a randomly chosen variable $HP(h_j)$, $|p_{exact}(HP(h_j)) - p_{MCMC}(HP(h_j))|/p_{exact}(HP(h_j))$. It shows that our lifted MCMC converges to the true density much faster than the ground MCMC.[13] Figure 4 (b) shows the average sampling time (per step) with different number of rvs, i.e. the number of houses.

[13]The lifted Belief Propagation (LBP) [29] is not directly applicable because the target distribution is a mixture of two iid Gaussians. The LBP assumes that each house sends the same message, i.e. price estimation. However, here, the messages among houses should be constrained. That is, the prices of some houses go down, and then prices of the other houses should go up.

10 Conclusions

We propose new lifted relational variational inference algorithms for relational hybrid models. Our main contributions are two folds: (1) in theory, we show that a relational model, which can represent large-scale systems, is accurately represented by a variational relational model; (2) our lifted algorithms are the first to solve inference problems without referring ground rvs for non-Gaussian continuous models. Experiments show that our method outperforms the existing possible methods in various cases including a real-world environmental problem.

Acknowledgments

We wish to thank Wen Pu, Jihye Seong and anonymous reviewers for their valuable, constructive comments. This material is supported by NSF award ECS-09-43627 - Improving Prediction of Subsurface Flow and Transport through Exploratory Data Analysis and Complementary Modeling and DARPA award AF Sub SRI 27-001337.

References

[1] Babak Ahmadi, Kristian Kersting, and Scott Sanner. Multi-evidence lifted message passing, with application to pagerank and the kalman filter. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 1152–1158, 2011.

[2] Udi Apsel and Ronen I. Brafman. Extended lifted inference with joint formulas. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2011*, pages 11–18, 2011.

[3] Fahiem Bacchus. *Representing and reasoning with probabilistic knowledge: a logical approach to probabilities*. MIT Press, Cambridge, MA, USA, 1990.

[4] Peter Carbonetto, Jacek Kisynski, Nando de Freitas, and David Poole. Nonparametric bayesian logic. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, UAI 2005*, pages 85–93, 2005.

[5] Jaesik Choi, Eyal Amir, and David Hill. Lifted inference for relational continuous models. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2010*, pages 126–134, 2010.

[6] Jaesik Choi, Rodrigo de Salvo Braz, and Hung H. Bui. Efficient methods for lifted inference with aggregate factors. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011*, pages 1030–1036, 2011.

[7] Jaesik Choi, Abner Guzman-Rivera, and Eyal Amir. Lifted relational kalman filtering. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 2092–2099, 2011.

[8] Myung Jin Choi, Vincent Y. F. Tan, Animashree Anandkumar, and Alan S. Willsky. Learning latent tree graphical models. *Journal of Machine Learning Research*, 12:1771–1812, 2011.

[9] Wei Chu, Vikas Sindhwani, Zoubin Ghahramani, and S. Sathiya Keerthi. Relational learning with gaussian processes. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, NIPS 2006*, pages 289–296, 2006.

[10] Sanjoy Dasgupta. Learning mixtures of gaussians. In *The 40th Annual Symposium on Foundations of Computer Science, FOCS 1999*, pages 634–644, 1999.

[11] B. de Finetti. Funzione caratteristica di un fenomeno aleatorio. *Mathematice e Naturale*, 1931.

[12] Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. Lifted first-order probabilistic inference. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI 2005,*, pages 1319–1325, 2005.

[13] P. Diaconis and D. Freedman. Finite exchangeable sequences. *Annals of Probability*, 8(1):115–130, 1980.

[14] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 1999*, pages 1300–1309, 1999.

[15] Joseph Y. Halpern. An analysis of first-order logics of probability. *Artif. Intell.*, 46(3):311–350, 1990.

[16] E. Hewitt and L.J. Savage. Symmetric measures on cartesian products. *Trans. Amer. Math. Soc.*, 80:470–501, 1955.

[17] Abhay Kumar Jha, Vibhav Gogate, Alexandra Meliou, and Dan Suciu. Lifted inference seen from the other side : The tractable features. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems, NIPS 2010*, pages 973–981, 2010.

[18] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, UAI 1995*, pages 338–345, 1995.

[19] Michael I. Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[20] G. Lovison. An alternative representation of altham's multiplicative-binomial distribution. *Statistics & Probability Letters*, 36(4):415–420, 1998.

[21] Brian Milch, Bhaskara Marthi, Stuart J. Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. Blog: Probabilistic models with unknown objects. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI 2005*, pages 1352–1359, 2005.

[22] Brian Milch and Stuart J. Russell. First-order probabilistic languages: Into the unknown. In *Inductive Logic Programming, 16th International Conference, ILP 2006*, pages 10–24, 2006.

[23] Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted probabilistic inference with counting formulas. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 1062–1068, 2008.

[24] Raymond Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[25] Avi Pfeffer, Daphne Koller, Brian Milch, and Ken T. Takusagawa. Spook: A system for probabilistic object-oriented knowledge representation. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, UAI 1999*, pages 541–550, 1999.

[26] David Poole. First-order probabilistic inference. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003*, pages 985–991, 2003.

[27] Carl Edward Rasmussen and Zoubin Ghahramani. Infinite mixtures of gaussian process experts. In *Advances in Neural Information Processing Systems 14, Neural Information Processing Systems: Natural and Synthetic*, pages 881–888, 2001.

[28] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[29] Parag Singla and Pedro Domingos. Lifted first-order belief propagation. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 1094–1099, 2008.

[30] Lei Xu and Michael I. Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural Computation*, 8:129–151, 1995.

[31] Zhao Xu, Kristian Kersting, and Volker Tresp. Multi-relational learning with gaussian processes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009*, pages 1309–1314, 2009.

[32] Nevin Lianwen Zhang. Hierarchical latent class models for cluster analysis. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, AAAI/IAAI 2002*, pages 230–237, 2002.

A Bayesian Approach to Constraint Based Causal Inference

Tom Claassen and Tom Heskes
Institute for Computer and Information Science
Radboud University Nijmegen
The Netherlands

Abstract

We target the problem of accuracy and robustness in causal inference from finite data sets. Some state-of-the-art algorithms produce clear output complete with solid theoretical guarantees but are susceptible to propagating erroneous decisions, while others are very adept at handling and representing uncertainty, but need to rely on undesirable assumptions. Our aim is to combine the inherent robustness of the Bayesian approach with the theoretical strength and clarity of constraint-based methods. We use a Bayesian score to obtain probability estimates on the input statements used in a constraint-based procedure. These are subsequently processed in decreasing order of reliability, letting more reliable decisions take precedence in case of conflicts, until a single output model is obtained. Tests show that a basic implementation of the resulting Bayesian Constraint-based Causal Discovery (BCCD) algorithm already outperforms established procedures such as FCI and Conservative PC. It can also indicate which causal decisions in the output have high reliability and which do not.

1 Introduction: Robust Causal Discovery

In many real-world systems the relations and interactions between variables can be modeled in the form of a causal directed acyclic graph (DAG) $\mathcal{G}_C$ over a set of variables $\mathbf{V}$. A directed path from A to B in such a graph indicates a *causal relation* $A \Rightarrow B$ in the system, where cause A *influences* the value of its effect B, but not the other way around. An edge $A \rightarrow B$ in $\mathcal{G}_C$ indicates a *direct* causal link. In data from a system with a causal relation $A \Rightarrow B$ (direct or indirect), the values of A and B have a tendency to vary together, i.e. they become probabilistically dependent.
Two assumptions are usually employed to link the underlying, asymmetric causal relations to observable, symmetric probabilistic dependencies:

The **Causal Markov Condition** states that each variable in a causal DAG $\mathcal{G}_C$ is (probabilistically) independent of its non-descendants, given its parents.
The **Causal Faithfulness Condition** states that there are no independencies between variables that are not entailed by the Causal Markov Condition.

The first makes it possible to go from causal graph to observed probabilistic independencies; the second completes the way back. Together, they imply that the causal DAG $\mathcal{G}_C$ is also *minimal*, in the sense that no proper subgraph can satisfy both assumptions *and* produce the same probability distribution (Zhang and Spirtes, 2008).

Before we continue with causal discovery methods, first a quick recap of some standard concepts and terminology. The joint probability distribution induced by a causal DAG $\mathcal{G}_C$ factors according to a *Bayesian network* (BN): a pair $\mathcal{B} = (\mathcal{G}, \boldsymbol{\Theta})$, where $\mathcal{G} = (\mathbf{V}, \mathbf{A})$ is DAG over random variables $\mathbf{V}$, and the parameters $\boldsymbol{\theta}_V \subset \boldsymbol{\Theta}$ represent the conditional probability of variable $V \in \mathbf{V}$ given its parents $\mathbf{Pa}(V)$ in the graph $\mathcal{G}$.
Probabilistic independencies can be read from the graph $\mathcal{G}$ via the well-known *d-separation* criterion: X is conditionally independent of Y given $\mathbf{Z}$, denoted $X \perp\!\!\!\perp Y \,|\, \mathbf{Z}$, iff there is no unblocked path between X and Y in $\mathcal{G}$ conditional on the nodes in $\mathbf{Z}$; see e.g. (Pearl, 1988; Neapolitan, 2004).

If some of the variables in the causal DAG are hidden then the independence relations between the observed variables may be represented in the form of a *(maximal) ancestral graph* (MAG) (Richardson and Spirtes, 2002). MAGs form an extension of the class of DAGs that is closed under marginalization and selection. In

addition to directed arcs, MAGs can also contain *bi-directed arcs* $X \leftrightarrow Y$ (indicative of marginalization) and undirected edges $X - - Y$ (indicative of selection). The *causal sufficiency* assumption states that there are no hidden common causes of the observed variables in $\mathcal{G}$, which implies that the distribution over the observed variables still conforms to a Bayesian network. In this article we ignore selection bias (= no undirected edges), but do *not* rely on causal sufficiency.

The *equivalence class* $[\mathcal{G}]$ of a graph $\mathcal{G}$ is the set of all graphs that are indistinguishable in terms of (Markov) implied independencies.[1] For a DAG or MAG $\mathcal{G}$, the corresponding equivalence class $[\mathcal{G}]$ can be represented as a *partial ancestral graph* (PAG) $\mathcal{P}$, which keeps the skeleton (adjacencies) and all invariant edge marks, i.e. tails ($-$) and arrowheads ($>$) that appear in all members of the equivalence class, and turns the remaining non-invariant edge marks into circles ($\circ$) (Zhang, 2008). The invariant arrowhead at an edge $A *\!\!\rightarrow B$ in $\mathcal{P}$ signifies that B is *not* a cause of A. An edge $A \rightarrow B$ implies a causal link $A \Rightarrow B$ that is also direct.

With this in mind, the task of a causal discovery algorithm is to find as many invariant features of the equivalence class corresponding to a given data set as possible. From this, all identifiable, present or absent causal relations can be read.

Causal discovery procedures

A large class of **constraint-based** causal discovery algorithms is based directly on the faithfulness assumption: if a conditional independence $X \perp\!\!\!\perp Y \,|\, \mathbf{Z}$ can be found for *any* set of variables $\mathbf{Z}$, then there is no direct causal relation between X and Y in the underlying causal graph $\mathcal{G}_C$, and hence no edge between X and Y in the equivalence class $\mathcal{P}$. In this way, an exhaustive search over all pairs of variables can uncover the entire skeleton of $\mathcal{P}$. In the subsequent stage a number of orientation rules are executed that find the invariant tails and arrowheads.

Members of this group include the IC-algorithm (Pearl and Verma, 1991), PC/FCI (Spirtes et al., 2000), Grow-Shrink (Margaritis and Thrun, 1999), TC (Pellet and Elisseef, 2008), and many others. All involve repeated independence tests in the adjacency search phase, and employ orientation rules as described in Meek (1995). The differences lie mainly in the search strategy employed, size of the conditioning sets, and additional assumptions imposed. Of these, the FCI algorithm in conjunction with the additional orientation rules in (Zhang, 2008) is the only one that is sound and complete in the large-sample limit when hidden common causes and/or selection bias may be present.

Constraint-based procedures tend to output a single, reasonably clear graph, representing the class of all possible causal DAGs. The downside is that for finite data they give little indication of which parts of the network are stable (reliable), and which are not: if unchecked, even one erroneous, borderline independence decision may be propagated through the network, leading to multiple incorrect orientations (Spirtes, 2010).

To tackle the perceived lack of robustness of PC, Ramsey et al. (2006) proposed a **conservative approach** for the orientation phase. The standard rules draw on the implicit assumption that, after the initial adjacency search, a single $X \perp\!\!\!\perp Y \,|\, \mathbf{Z}$ should suffice to orient an unshielded triple $\langle X, Z, Y \rangle$, as Z should be either part of *all* or part of *no* sets that separate X and Y. The Conservative PC (CPC) algorithm tests explicitly whether this assumption holds, and only orients the triple into a noncollider resp. v-structure $X \rightarrow Z \leftarrow Y$ if found true. If not, then it is marked as *unfaithful*. Tests show that CPC significantly outperforms standard PC in terms of overall accuracy, albeit often with less informative output, for only a marginal increase in run-time.

This idea can be extended to FCI: the set of potential separating nodes is now conform FCI's adjacency search, and any of Zhang's orientation rules that relies on a particular unshielded (non-)collider does not fire on an unfaithful triple. See (Glymour et al., 2004; Kalisch et al., 2011) for an implementation of Conservative FCI (CFCI) and many related algorithms.

The **score-based** approach is an alternative paradigm that builds on the implied *minimality* of the causal graph: define a scoring criterion $S(\mathcal{G}, \mathbf{D})$ that measures how well a Bayesian network with structure $\mathcal{G}$ fits the observed data $\mathbf{D}$, while preferring simpler networks, with fewer free parameters, over more complex ones. If the causal relations between the variables in $\mathbf{D}$ form a causal DAG $\mathcal{G}_C$, then in the large sample limit the highest scoring structure $\mathcal{G}$ must be part of the equivalence class of $[\mathcal{G}_C]$.

An example is the (Bayesian) likelihood score: given a Bayesian network $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$, the likelihood of observing a particular data set $\mathbf{D}$ can be computed recursively from the network. Integrating out the parameters $\mathbf{\Theta}$ in the conditional probability tables (CPTs) then results in:

$$p(\mathbf{D}|\mathcal{G}) = \int_{\mathbf{\Theta}} p(\mathbf{D}|\mathcal{G}, \mathbf{\Theta}) f(\mathbf{\Theta}|\mathcal{G}) \, \mathrm{d}\mathbf{\Theta}, \qquad (1)$$

where f is a conditional probability density function over the parameters $\mathbf{\Theta}$ given structure $\mathcal{G}$.

[1] We follow the standard assumption that Markov equivalence implies statistical equivalence (Spirtes, 2010).

A closed form solution to eq.(1) is used in algorithms such as K2 (Cooper and Herskovits, 1992) and the Greedy Equivalence Search (GES) (Chickering, 2002) to find an optimal structure by repeatedly comparing scores for slightly modified alternatives until no more improvement can be found. See also Bouckaert (1995) for an evaluation of different strategies using these and other measures such as the BIC-score and minimum description length.

Score-based procedures can output a set of high-scoring alternatives. This ambiguity makes the result arguably less straightforward to read, but does allow for a measured interpretation of the reliability of inferred causal relations, and is not susceptible to incorrect categorical decisions (Heckerman et al., 1999). The main drawback is the need to rely on the causal sufficiency assumption.

2 The Best of Both Worlds

The strength of a constraint-based algorithm like FCI is its ability to handle data from arbitrary faithful underlying causal DAGs and turn it into sound and clear, unambiguous causal output. The strength of the Bayesian score-based approach lies in the robustness and implicit confidence measure that a likelihood weighted combination of multiple models can bring.

▶ Our idea is to improve on conservative FCI by using a Bayesian approach to estimate the reliability of different constraints, and use this to decide if, when, and how that information should be used.

Instead of classifying pieces of information as reliable or not, we want to rank and process constraints according to a confidence measure. This should allow to avoid propagating unreliable decisions while retaining more confident ones. It also provides a principled means for conflict resolution. The end-result is hopefully a more informative output model than CFCI, while obtaining a higher accuracy than standard FCI can deliver.

To obtain a confidence measure that can be compared across different estimates we want to compute the *probability* that a given independence statement holds from a given data set $\mathbf{D}$. In an ideal Bayesian approach we could compute a likelihood $p(\mathbf{D}|\mathcal{M})$ for each $\mathcal{M} \in \mathbf{M}$ (see section 3 on how to approximate this). If we know that the set $\mathbf{M}$ contains the 'true' structure, then the probability of an independence hypothesis I follows from normalized summation as:

$$p(I|\mathbf{D}) \propto \sum_{\mathcal{M} \in \mathbf{M}(I)} p(\mathbf{D}|\mathcal{M}) p(\mathcal{M}), \qquad (2)$$

(Heckerman et al., 1999), where $\mathbf{M}(I)$ denotes the subset of structures that entail independence statement I, and $p(\mathcal{M})$ represents a prior distribution over the structures (see §3.4).

Two remarks. Firstly, it is well known that the number of possible graphs grows very quickly with the number of nodes $\mathbf{V}$. But eq.(2) equally applies when we limit data and structures to *subsets* of variables $\mathbf{X} \subset \mathbf{V}$. For sparse graphs we can choose to consider only subsets of size $K \ll |\mathbf{V}|$. We opt to go one step further and follow a search strategy similar to PC/FCI, using structures of increasing size. Secondly, it would be very inefficient to compute eq.(2) for each independence statement we want to evaluate. From a single likelihood distribution over structures over $\mathbf{X}$ we can immediately compute the probability of *all* possible independence statements between variables in $\mathbf{X}$, including complex combinations such as those implied by v-structures, just by summing the appropriate contributions for each statement.

Having obtained probability estimates for a list of in/dependence statements $\mathcal{I}$, we can rank these in decreasing order of reliability, and keep the ones based on a decision threshold $p(I|\mathbf{D}) > \theta$, with $\theta = 0.5$ as intuitive default. In case of remaining conflicting statements, the ones with higher confidence take precedence. The resulting algorithm is outlined below:

Algorithm 1 Outline

Start : database $\mathbf{D}$ over variables $\mathbf{V}$
Stage 1 - Adjacency search
1: fully connected graph $\mathcal{P}$, empty list $\mathcal{I}$, $K = 0$
2: **repeat**
3: **for all** $X - Y$ still connected in $\mathcal{P}$ **do**
4: **for all** adjacent sets $\mathbf{Z}$ of K nodes in $\mathcal{P}$ **do**
5: estimate $p(\mathcal{M}|\mathbf{D})$ over $\{X, Y, \mathbf{Z}\}$
6: sum to $p(I|\mathbf{D})$ for independencies I
7: update $\mathcal{I}$ and $\mathcal{P}$ for each $p(I|\mathbf{D}) > \theta$
8: **end for**
9: **end for**
10: $K = K + 1$
11: **until** all relevant found
Stage 2 - Orientation rules
12: rank and filter $\mathcal{I}$ in decreasing order of reliability
13: orient unshielded triples in $\mathcal{P}$
14: run remaining orientation rules
15: return causal model $\mathcal{P}$

In this form, the Bayesian estimates are only used to guide the adjacency search (update skeleton $\mathcal{G}$, l.7), and to filter the list of independencies $\mathcal{I}$ (l.12). Ideally, we would like the probabilities to guide the orientation phase as well. This implies processing the independence statements sequentially, in decreasing order of reliability. For that we can use a recently developed

variant of the FCI algorithm (Claassen and Heskes, 2011), that breaks up the inference into a series of *modular* steps that can be executed in arbitrary order. It works by translating observed independence constraints into **logical statements** about the presence or absence of certain causal relations.

Using square brackets to indicate *minimal* sets of nodes: if a variable Z either *makes* or *breaks* an independence relation between $\{X, Y\}$, then

1. $X \perp\!\!\!\perp Y \,|\, [\mathbf{W} \cup Z] \;\; \vdash \;\; (Z \Rightarrow X) \;\vee\; (Z \Rightarrow Y)$,
2. $X \not\perp\!\!\!\perp Y \,|\, \mathbf{W} \cup [Z] \;\; \vdash \;\; Z \not\Rightarrow (\{X, Y\} \cup \mathbf{W})$.

In words: from a minimal independence we infer the *presence* of at least one from two causal relations, whereas a dependence identifies the *absence* of causal relations. Subsequent causal statements follow from deduction on the properties *transitivity* $(X \Rightarrow Y) + (Y \Rightarrow Z) \vdash (X \Rightarrow Z)$, and *irreflexivity* $(X \not\Rightarrow X)$.

3 Sequential Causal Inference

This section discusses the steps needed to turn the previous idea into a working algorithm in the next section. Main issues are: probability estimates for logical causal statements from substructures, Bayesian likelihood computation, and inference from unfaithful DAGs. Proofs are detailed in (Claassen and Heskes, 2012).

A word on notation: we use $\mathbf{D}$ to denote a data set over variables $\mathbf{V}$ from a distribution that is faithful to some (larger) causal DAG $\mathcal{G}_C$. Ł denotes the set of logical causal statements L over two or three variables in $\mathbf{V}$, of the form given in the r.h.s. of rules 1 and 2, above. We use $\mathbf{M_X}$ to represent the set of MAGs over $\mathbf{X}$, and $\mathbf{M_X}(L)$ to denote the subset that entails logical statement L. We also use $\mathcal{G}$ to explicitly indicate a DAG, $\mathcal{M}$ for a MAG, and $\mathcal{P}$ for a PAG.

3.1 A Modular Approach

In order to process available information in (decreasing) order of reliability we need to obtain probability estimates for logical statements on causal relations from data. Similar to eq.(2), this follows from summing the normalized posterior likelihoods of all MAGs that entail that statement through m-separation:

Lemma 1. The probability of a logical causal statement L given a data set $\mathbf{D}$ is given by

$$p(L|\mathbf{D}) = \frac{\sum_{\mathcal{M} \in \mathbf{M}(L)} p(\mathbf{D}|\mathcal{M}) p(\mathcal{M})}{\sum_{\mathcal{M} \in \mathbf{M}} p(\mathbf{D}|\mathcal{M}) p(\mathcal{M})}, \qquad (3)$$

using the notational conventions introduced above.

As stated, in many cases considering only a small subset of the variables in $\mathbf{V}$ is already sufficient to infer L. But that also implies that there are multiple subsets that imply L, each with different probability estimates. As these relate to different sets of variables, they should not be combined as in standard multiple hypothesis tests, but instead we want to look for the *maximum* value that can be found.

Lemma 2. Let $\mathbf{D}$ be a data set over variables $\mathbf{V}$. Then $\forall \mathbf{X} \subseteq \mathbf{V} : p(L|\mathbf{D}) \geq \sum_{\mathcal{M} \in \mathbf{M_X}(L)} p(\mathcal{M}|\mathbf{D})$.

Proof. Let $p(\mathcal{M}|\mathbf{D})$ be the posterior probability of MAG $\mathcal{M}$ given data $\mathbf{D}$. Let $\mathcal{M}(\mathbf{X})$ denote the MAG $\mathcal{M}$ marginalized to variables $\mathbf{X}$, then:

$$\begin{aligned} p(L|\mathbf{D}) &= \sum_{\mathcal{M} \in \mathbf{M_V}(L)} p(\mathcal{M}|\mathbf{D}) \\ &\geq \sum_{\mathcal{M} \in \mathbf{M_V}(L) : \mathcal{M}(\mathbf{X}) \in \mathbf{M_X}(L)} p(\mathcal{M}|\mathbf{D}) \\ &= \sum_{\mathcal{M} \in \mathbf{M_X}(L)} p(\mathcal{M}|\mathbf{D}) \end{aligned}$$

The inequality follows from the fact that, by definition, no marginal MAG $\mathcal{M}(\mathbf{X})$ entails a statement not entailed by $\mathcal{M}$, whereas the converse can (and does) occur. □

As a result: $p(L|\mathbf{D}) \geq \max_{\mathbf{X} \subset \mathbf{V}} \sum_{\mathcal{M} \in \mathbf{M_X}(L)} p(\mathcal{M}|\mathbf{D})$.

It means that while searching for logical causal statements Ł, it makes sense to keep track of the maximum probabilities obtained so far.

However, computing $p(\mathcal{M}|\mathbf{D})$ for $\mathcal{M} \in \mathbf{M_X}$ still involves computing likelihoods over all structures over $\mathbf{V}$, which is precisely what we want to avoid. A reasonable approximation is provided by $p(\mathcal{M}|\mathbf{D_X})$, i.e. the estimates obtained by only including data in $\mathbf{D}$ from the variables $\mathbf{X}$. It means that the lower bound is no longer guaranteed to hold universally, but should still be adequate in practice.

3.2 Obtaining likelihood estimates

If we know that the 'true' structure over a subset $\mathbf{X} \subseteq \mathbf{V}$ takes the form of a DAG, then computing the required likelihood estimates $p(\mathbf{D_X}|\mathcal{G})$ is relatively straightforward. Cooper and Herskovits (1992) showed that, under some reasonable assumptions, for discrete random variables the integral (1) has a closed-form solution. In the form presented in (Heckerman et al., 1995) this score is known as the *Bayesian Dirichlet* (BD) metric:

$$p(\mathbf{D}|\mathcal{G}) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})}, \qquad (4)$$

with n the number of variables, r_i the multiplicity of variable X_i, q_i the number of possible instantiations of the parents of X_i in $\mathcal{G}$, N_{ijk} the number of cases in data set $\mathbf{D}$ in which variable X_i has the value $r_{i(k)}$ while its parents are instantiated as $q_{i(j)}$, and with $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. The $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$ represent the pseudocounts for a Dirichlet prior over the parameters in the corresponding CPTs.

Different strategies for choosing the prior exist: for example, choosing $N'_{ijk} = 1$ (uniform prior) leads to the original *K2-metric*, see (Cooper and Herskovits, 1992). Setting $N'_{ijk} = N'/(r_i q_i)$ gives the popular *BDeu*-metric, which is *score equivalent* in the sense that structures from the same equivalence class $[\mathcal{G}]$ receive the same likelihood score, cf. (Buntine, 1991). In this article, we opt for the $K2$-metric, as it seems more appropriate in causal settings (Heckerman et al., 1995). But having to consider only one instance of every equivalence class may prove a decisive advantage of the BDe(u)-metric in future extensions.

However, eq.(4) only applies to DAGs. We know that, even when assuming causal sufficiency applies for the variables $\mathbf{V}$, considering arbitrary subsets size $|\mathbf{X}| \geq 4$ in general will require MAG representations to account for common causes that are not in $\mathbf{X}$. Extending the derivation of eq.(4) requires additional assumptions on multiplicity and number of the hidden variables, and turns the nice closed-form solution into an intractable problem that requires approximation, e.g. through sampling (Heckerman et al., 1999). This would make each step in our approach much more expensive. Recently, Evans and Richardson (2010) showed a maximum likelihood approach to fit acyclic directed mixed graphs (a superset of MAGs) directly on binary data. Unfortunately, this method cannot provide the likelihood estimates per model we need for our purposes. Silva and Ghahramani (2009) do present a Bayesian approach, but need to put additional constraints on the distribution in the form of (cumulative) Gaussian models.

In short, even though we would like to use MAGs to compute $p(\mathbf{D_X}|\mathcal{M})$ directly in eq.(3), at the moment we have to rely on DAGs to obtain approximations to the 'true' value. This will result in less accurate reliability estimates for $p(L|\mathbf{D})$, but also means that we may miss certain pieces of information, or, even worse, that the inference may become invalid.

3.3 Unfaithful inference: DAGs vs. MAGs

Fortunately we can show that, even when the true independence structure over a subset $\mathbf{X} \subset \mathbf{V}$ is a MAG, we can still do valid inference via $p(L|\mathbf{D_X})$ from likelihood scores over an exhaustive set of DAGs over $\mathbf{X}$, provided we account for unfaithful DAG representations. This part discusses unfaithful inference, and how the mapping from structures to logical causal statements can be modified. Much of what we show builds on (Bouckaert, 1995).

In the large-sample limit, the Bayesian likelihood score picks the smallest DAG structure(s) that can capture the observed probability distribution exactly.

Definition 1 (Optimal uDAG). A DAG $\mathcal{G}$ is an (unfaithful) **uDAG** approximation to a MAG $\mathcal{M}$ over a set of nodes $\mathbf{X}$, iff for any probability distribution $p(\mathbf{X})$, generated by an underlying causal graph faithful to $\mathcal{M}$, there is a set of parameters $\mathbf{\Theta}$ such that the Bayesian network $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$ encodes the same distribution $p(\mathbf{X})$. The uDAG is **optimal** if there exists no uDAG to $\mathcal{M}$ with fewer free parameters.

A uDAG is just a DAG for which we do not know if it is faithful or not. Reading in/dependence relations from a uDAG goes as follows:

Lemma 3. Let $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$ be a Bayesian network over a set of nodes $\mathbf{X}$, with $\mathcal{G}$ a uDAG for some MAG that is faithful to a distribution $p(\mathbf{X})$. Let $\mathcal{G}_{X \| Y}$ be the graph obtained by eliminating the edge $X - Y$ from $\mathcal{G}$ (if present). Then, if $X \perp\!\!\!\perp_{\mathcal{G}_{X\|Y}} Y \,|\, \mathbf{Z}$ then:

$$(X \perp\!\!\!\perp_{\mathcal{G}} Y \,|\, \mathbf{Z}) \Leftrightarrow (X \perp\!\!\!\perp_{p} Y \,|\, \mathbf{Z}).$$

Proof sketch. The independence rule $(X \perp\!\!\!\perp_{\mathcal{G}} Y \,|\, \mathbf{Z}) \Rightarrow (X \perp\!\!\!\perp_{p} Y \,|\, \mathbf{Z})$ follows (Pearl, 1988). The dependence rule $(X \perp\!\!\!\perp_{\mathcal{G}_{X\|Y}} Y \,|\, \mathbf{Z}) \wedge (X \not\perp\!\!\!\perp_{\mathcal{G}} Y \,|\, \mathbf{Z}) \Rightarrow (X \not\perp\!\!\!\perp_{p} Y \,|\, \mathbf{Z})$ is similar to the 'coupling' theorem (3.11) in (Bouckaert, 1995), but stronger. As we assume a faithful MAG, a dependence $X \not\perp\!\!\!\perp_{p} Y \,|\, \mathbf{Z}$ cannot be destroyed by in/excluding a node U that has no unblocked path in the underlying MAG to X and/or Y given $\mathbf{Z}$. This eliminates one of the preconditions in the coupling theorem. See (Claassen and Heskes, 2012) for details. □

So, in a uDAG all independencies from d-separation are still valid, but the identifiable dependencies are restricted.

Example 1. Treating the uDAG in Figure 1(b) as a faithful DAG would suggest $X \perp\!\!\!\perp_{p} T \,|\, [Z]$, and hence $(Z \Rightarrow X) \vee (Z \Rightarrow T)$. This is wrong: Figure 1(a) shows that Z is ancestor of neither X nor T. Lemma 3 would not make this mistake, as it allows to deduce $X \perp\!\!\!\perp_{p} T \,|\, Z$, but not the erroneous $X \not\perp\!\!\!\perp_{p} T$.

We can generalize Lemma 3 to *indirect* dependencies.

Lemma 4. Let $\mathcal{G}$ be a uDAG for a faithful MAG $\mathcal{M}$. Let X, Y, and $\mathbf{Z}$ be disjoint (sets of) nodes. If $\pi = \langle X, .., Y \rangle$ is the *only* unblocked path from X to Y given $\mathbf{Z}$ in $\mathcal{G}$, then $X \not\perp\!\!\!\perp_{p} Y \,|\, \mathbf{Z}$.

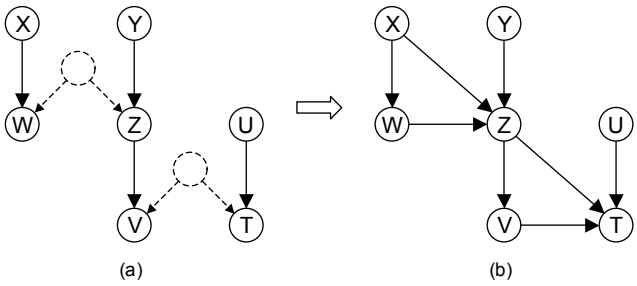


Figure 1: (a) causal DAG with hidden variables, (b) uDAG with unfaithful $X \perp\!\!\!\perp_{\mathcal{G}} T \,|\, [Z]$

Example 2. With Lemma 4 we infer from Figure 1(b) that $X \not\perp\!\!\!\perp_p T \,|\, \{V, W\}$. We also find that $Y \not\perp\!\!\!\perp_p V$, from which, in combination with $Y \perp\!\!\!\perp_p V \,|\, [Z]$, we (rightly) conclude that $(Z \Rightarrow Y) \vee (Z \Rightarrow V)$, see (a).

In general, Lemmas 3 and 4 assert different dependencies for different uDAG members of the same equivalence class. If the uDAG $\mathcal{G}$ is *optimal*, then all in/dependence statements from any uDAG member of the corresponding equivalence class $[\mathcal{G}]$ are valid. In that case we can do the inference based on the PAG representation $\mathcal{P}$ of $[\mathcal{G}]$. This provides additional information, but also simplifies some inference steps. Again, see (Claassen and Heskes, 2012) for details.

For example, identifying an absent causal relation (arrowhead) $X \not\Leftarrow Y$ from an optimal uDAG becomes identical to the inference from a faithful MAG. Let a *potentially directed path* (p.d.p.) be a path in a PAG that could be oriented into a directed path by changing circle marks into appropriate tails/arrowheads, then

Lemma 5. Let $\mathcal{G}$ be an optimal uDAG to a faithful MAG $\mathcal{M}$, then the absence of a causal relation $X \not\Leftarrow Y$ can be identified, iff there is no potentially directed path from X to Y in the PAG $\mathcal{P}$ of $[\mathcal{G}]$.

Proof sketch. The optimal uDAG $\mathcal{G}$ is obtained by (only) adding edges between variables in the MAG $\mathcal{M}$ to eliminate invariant bi-directed edges, until no more are left. At that point the uDAG is a representative of the corresponding equivalence class $\mathcal{P}$ (Theorem 2 in Zhang (2008)). For any faithful MAG all and only the nodes not connected by a p.d.p. in the corresponding PAG have a definite non-ancestor relation in the underlying causal graph. At least one uDAG instance in the equivalence class of an optimal uDAG over a given skeleton leaves the ancestral relations of the original MAG intact.Therefore, any remaining invariant arrowhead in the PAG $\mathcal{P}$ matches a non-ancestor relation in the original MAG. □

For the presence of causal relations (tails) a similar, but more complicated criterion can be found; see Supplement. Ultimately, the impact of having to use uDAGs boils down to a modified mapping of structures to logical causal statements, based on the inference rules above.

Finally, it is worth mentioning that in the large-sample limit, matching uDAGs over increasing sets of nodes we are guaranteed to find all independencies needed to obtain the skeleton, as well as all invariant arrowheads and many invariant tails. However, as the primary goal remains to improve accuracy/robustness when the large-sample limit does *not* apply, we do not pursue this matter further here.

3.4 Consistent prior over structures

The computation of $p(L|\mathbf{D}_{\mathbf{X}})$ requires a prior distribution $p(\mathcal{M})$ over the set of MAGs over $\mathbf{X}$. A straightforward solution is to use a uniform prior, assigning equal probability to each $\mathcal{M} \in \mathbf{M}$. Alternatively, we can use a predefined function that penalizes complexity or deviation w.r.t. some reference structure (Chickering, 2002; Heckerman et al., 1995). If we want to exploit score-equivalence with the BDe(u) metric in eq.(4), we can weight DAG representatives according to the size of their equivalence class.

If we have background information on expected (or desired) properties of the structure, such as max. node degree, average connectivity, or small-world/scale-free networks, we can use this to construct a prior $p(\mathcal{M})$ through *sampling*: generate random graphs over all variables in accordance with the specified characteristics, sample one or more random subsets of variables size K, and compute the marginal structure over that subset. Averaging over structures that are PAG-isomorphs (equivalence classes identical under relabeling) improves both consistency and convergence.

Irrespective of the method, it is essential to ensure the prior is also *consistent* over structures of different size. Perhaps surprisingly, this is *not* obtained by applying the same strategy at different levels: a uniform distribution over DAGs over $\{X, Y, Z\}$ implies $p(\text{“}X \perp\!\!\!\perp Y\text{”}) = 6/25$, whereas a uniform distribution over two-node DAGs implies $p(\text{“}X \perp\!\!\!\perp Y\text{”}) = 1/3$. We obtain a consistent multi-level prior by starting from a preselected level K, and then extend to different sized structures through marginalization.

4 The BCCD algorithm

We can now turn the results from the previous section into a working algorithm. The implementation largely follows the outline in Algorithm 1, except that now

uDAGs (instead of MAGs) are used to obtain a list of logical causal statements (instead of independencies), and that logical inference takes the place of the orientation rules, resulting in the Bayesian Constraint-based Causal Discovery (BCCD) algorithm.

A crucial step in the algorithm is the mapping $\mathcal{G} \times$ Ł from optimal uDAG structures to causal statements in line 9. This mapping is the same for each run, so it can be precomputed from the rules in section 3.3 and the Supplement, and stored for use afterwards (l.1) The uDAGs $\mathcal{G}$ are represented as adjacency matrices. For speed and efficiency purposes, we choose to limit the structures to size $K \leq 5$, which gives a list of 29,281 uDAGs at the highest level. For details about representation and rules, see (Claassen and Heskes, 2012).

Algorithm 2 Bayesian Constr. Causal Discovery

In : database $\mathbf{D}$ over variables $\mathbf{V}$, backgr.info $\mathcal{I}$
Out: causal relations matrix $\mathbf{M}_C$, causal PAG $\mathcal{P}$
Stage 0 - Mapping
1: $\mathcal{G} \times$ Ł $\leftarrow Get\_uDAG\_Mapping(\mathbf{V}, K_{max} = 5)$
2: $p(\mathcal{G}) \leftarrow Get\_Prior(\mathcal{I})$
Stage 1 - Search
3: fully connected $\mathcal{P}$, empty list Ł, $K = 0$, $\theta = 0.5$
4: **while** $K \leq K_{max}$ **do**
5: **for all** $X \in \mathbf{V}, Y \in Adj(X)$ in $\mathcal{P}$ **do**
6: **for all** $\mathbf{Z} \subseteq Adj(X)_{\setminus Y}$, $|\mathbf{Z}| = K$ **do**
7: $\mathbf{W} \leftarrow Check\_Unprocessed(X, Y, \mathbf{Z})$
8: $\forall \mathcal{G} \in \mathcal{G}_{\mathbf{W}}$: compute $p(\mathcal{G}|\mathbf{D}_{\mathbf{W}})$
9: $\forall L : p(L_{\mathbf{W}}|\mathbf{D}_{\mathbf{W}}) \leftarrow \sum_{\mathcal{G} \rightarrow L_{\mathbf{W}}} p(\mathcal{G}|\mathbf{D}_{\mathbf{W}})$
10: $\forall L : p(L) \leftarrow \max(p(L), p(L_{\mathbf{W}}|\mathbf{D}_{\mathbf{W}}))$
11: $\mathcal{P} \leftarrow p(\text{“}W_i \not\times W_j\text{”}|\mathbf{D}_{\mathbf{W}}) > \theta$
12: **end for**
13: **end for**
14: $K = K + 1$
15: **end while**
Stage 2 - Inference
16: $\mathbf{L}_C$ = empty 3D-matrix size $|\mathbf{V}|^3$, $i = 1$
17: Ł $\leftarrow Sort\_Descending$ (Ł, $p(L)$)
18: **while** $p(L_i) > \theta$ **do**
19: $\mathbf{L}_C \leftarrow Run\_Causal\_Logic(\mathbf{L}_C, L_i)$
20: $i \leftarrow i + 1$
21: **end while**
22: $\mathbf{M}_C \leftarrow Get\_Causal\_Matrix(\mathbf{L}_C)$
23: $\mathcal{P} \leftarrow Map\_To\_PAG(\mathcal{P}, \mathbf{M}_C)$

We verified the resulting mapping from uDAG rules to logical statements against a *brute-force approach* that checks the intersection of all logical causal statements implied by all MAGs that can have a particular uDAG as an optimal approximation. We found that at least for uDAG structures up to five nodes our rules are still sound and complete. Interestingly enough, for uDAGs up to four nodes all implied statements are *identical* to those that would be obtained if we just treated uDAGs as faithful DAGs. Only at five+ nodes do we have to take into account that uDAGs are unfaithful.

The adjacency search (l.3-15), loops over subsets from neigbouring nodes for identifiable causal information, while keeping track of adjacencies that can be eliminated (l.11). For structures over five or more nodes we need to consider nodes from FCI's *Possible-D-Sep* set (Spirtes et al., 1999). In practice, it rarely finds any additional (reliable) independencies, and we opt to skip this step for speed and simplicity (l.6), similar to the RFCI approach in (Colombo et al., 2011). As the set $\mathbf{W} = \{X, Y\} \cup \mathbf{Z}$ can be encountered in different ways, line (7) checks if the test on that set has been performed already. A list of probability estimates $p(L|\mathbf{D})$ for each logical causal statement is built up (l.10), until no more information is found.

The inference stage (l.16-21) then processes the list Ł in decreasing order of reliability, until the threshold is reached. Statements in Ł are added one-by-one to the matrix of logical causal statements $\mathbf{L}_C$ (encoding identical to Ł), with additional information inferred from the causal logic rules. Basic conflict resolution is achieved through not overriding existing information (from more reliable statements). The final step (l.22,23) retrieves all explicit causal relations in the form of a causal matrix $\mathbf{M}_C$, and maps this onto the skeleton $\mathcal{P}$ obtained from Stage 1 to return a graphical PAG representation.

5 Experimental Evaluation

We have tested various aspects of the BCCD algorithm in many different circumstances, and against various other methods. The principal aim at this stage is to verify the viability of the Bayesian approach. We compare our results and that of other methods from data against known ground-truth causal models. For that, we generate random causal graphs with certain predefined properties (adapted from Melancon et al. (2000); Chung and Lu (2002)), generate random data from this model, and marginalize out one or more hidden confounders. We looked at the impact of the number of data points, size of the models, sparseness, choices for parameter settings etc. on the performance to get a good feel for expected strengths and weaknesses in real-world situations.

It is well-known that the relative performance of different causal discovery methods can depend strongly on the performance metric and/or specific test problems used in the evaluation. Therefore, we will not claim that our method is inherently better than others based on the experimental results below, but simply note that the fact that in nearly all test cases the BCCD algorithm performed as good or better than

other methods, is a clear indication of the viability and potential of this approach.

Having limited space available, we only include results of tests against the two other state-of-the-art methods that can handle hidden confounders: FCI as the de facto benchmark, and its equivalent adapted from conservative PC. For the evaluation we use two complementary metrics: the *PAG accuracy* looks at the graphical causal model output and counts the number of edge marks that matches the PAG of true equivalence class (excluding self-references). The *causal accuracy* looks at the proportion of all causal decisions, either explicit as BCCD does or implicit from the PAG for FCI, that are correct compared to the generating causal graph.

In a nutshell: we found that in most circumstances conservative FCI outperforms vanilla FCI by about $3-4\%$ in terms of PAG accuracy and slightly more in terms of causal accuracy. In its standard form, with a uniform prior over structures of 5 nodes, the BCCD algorithm consistenly outperforms conservative FCI by a small margin of about $1-2\%$ at default decision thresholds ($\theta = 0.5$ for BCCD, $\alpha = 0.05$ for FCI). Including additional tests / nodes per test and using an extended mapping often increases this difference to about $2-4\%$ at optimal settings for both approaches (cf. Figure 3). This gain does come at a cost: BCCD has an increase in run-time of about a factor two compared to conservative FCI, which in turn is marginally more expensive than standard FCI. Evaluating many large structures can increases this cost even further, unless we switch to evaluating equivalence classes via the BDe metric in §3.2. However, the main benefit of the BCCD approach lies not in a slight improvement in accuracy, but in the added insight it provides into the generated causal model: even in this simple form, the algorithm gives a useful indication of which causal decisions are reliable and which are not, which seems very useful to have in practice.

The figures below illustrate some of these findings.

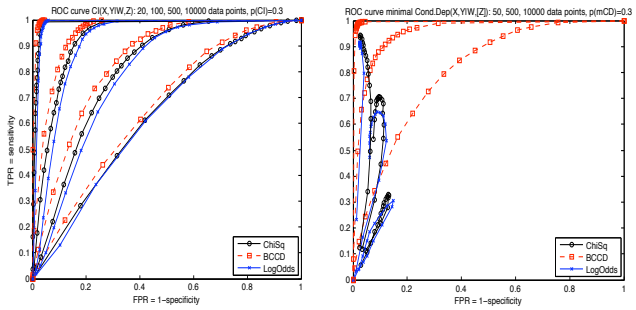


Figure 2: BCCD approach to (complex) independence test in eq.(2); (a) conditional independence $X \perp\!\!\!\perp Y \mid W, Z$, (b) *minimal* conditional dependence $X \not\perp\!\!\!\perp Y \mid W \cup [Z]$

First we implemented the BCCD approach as a (minimal) independence test. Figure 2 shows a typical example in the form of ROC-curves for different sized data sets, compared against a chi-squared test and a Bayesian log-odds test from (Margaritis and Bromberg, 2009), with the prior on independence as the tuning parameter for BCCD. For 'regular' conditional independence there was no significant difference (BCCD slightly ahead), but other methods reject minimal independencies for both high and low decision thresholds, resulting in the looped curves in (b); BCCD has no such problem.

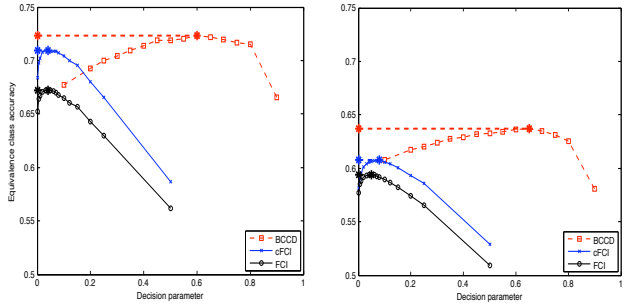


Figure 3: Equivalence class accuracy (% of edge marks in PAG) vs. decision parameter; for BCCD and (conservative) FCI, from 1000 random models; (a) 6 observed nodes, 1-2 hidden, 1000 points, (b) idem, 12 observed nodes

Figure 3 shows typical results for the BCCD algorithm itself: for a data set of 1000 records the *PAG accuracy* for both FCI and conservative FCI peaks around a threshold $\alpha \approx 0.05$ - lower for more records, higher for less - with conservative FCI consistently outperforming standard FCI. The BCCD algorithm peaks at a cut-off value $\theta \in [0.4, 0.8]$ with an accuracy that is slightly higher than the maximum for conservative FCI. The PAG accuracy tends not to vary much over this interval, making the default choice $\theta = 0.5$ fairly safe, even though the number of invariant edge marks does increase significantly (more decisions).

Table 5 shows the *confusion matrices* for edge marks in the PAG model at standard threshold settings for each of the three methods. We can recognize how FCI makes more explicit decisions than the other two (less circle marks), but also includes more mistakes. Conservative FCI starts from the same skeleton: it is more reluctant to orient uncertain v-structures, but manages to increase the overall accuracy as a result (sum of diagonal entries). The BCCD algorithm provides the expected compromise: more decisions than CFCI, but less mistakes than FCI, resulting in a modest improvement on the output PAG.

Figure 4 depicts the *causal accuracy* as a function of the tuning parameter for the three methods. The BCCD dependency is set against $(1-\theta)$ so that going

BCCD	⨯	→	—	−○
⨯	13.2	0.1	0.0	0.1
→	1.0	3.3	0.1	0.7
—	0.3	0.5	0.7	0.5
−○	1.6	1.7	0.8	5.5
	16.1	5.6	1.6	6.8

(a)

cFCI	⨯	→	—	−○
⨯	12.7	0.3	0.0	0.4
→	0.9	3.2	0.0	0.9
—	0.3	0.4	0.4	0.9
−○	1.4	1.8	0.3	6.0
	15.3	5.7	0.7	8.2

(b)

FCI	⨯	→	—	−○
⨯	12.7	0.5	0.0	0.2
→	0.9	3.5	0.0	0.6
—	0.3	0.9	0.4	0.4
−○	1.4	3.7	0.5	3.9
	15.3	8.6	0.9	5.1

(c)

Table 1: Confusion matrices for PAG output: (a) BCCD algorithm, (b) Conservative FCI, (c) Standard FCI ; rows = true value, columns = output edge mark (1000 random models over 6 nodes, 10,000 data points)

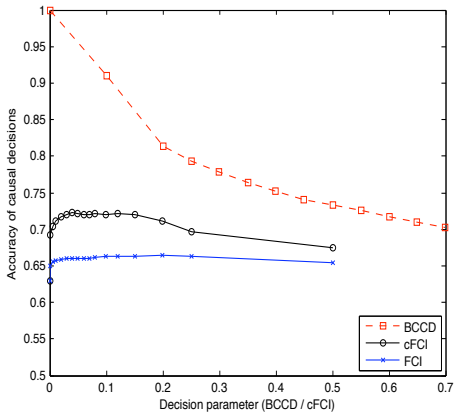


Figure 4: Accuracy of causal decisions as a function of the decision parameter

from $0 \rightarrow 1$ matches processing the list of statements in decreasing order of reliability. As hoped/expected: changing the decision parameter θ allows to access a range of accuracies, from a few very reliable causal relations to more but less certain indications. In contrast, the accuracy of the two FCI algorithms cannot be tuned effectively through the decision parameter α. The reason behind this is apparent from Figure 2(b): changing the decision threshold in an independence test shifts the balance between dependence and independence decisions, but it cannot identify or alter the balance in favor of more reliable decisions. We consider the fact that the BCCD *can* do exactly that as the most promising aspect of the Bayesian approach.

6 Extensions and Future Work

The experimental results confirm that the Bayesian approach is both viable and promising: even in a basic implementation the BCCD algorithm already outperforms other state-of-the-art causal discovery algorithms. It yields slightly better accuracy, both for optimal and standard settings of the decision parameters. Furthermore, BCCD comes with a decision threshold that is easy to interpret and can be used to vary from making just a few but very reliable causal statements to many possibly less certain decisions. Perhaps counterintuitively, changing the confidence level in (conservative) FCI does not lead to similar behavior as it only affects the balance between dependence and independence decisions, which in itself does not increase the reliability of either.

An interesting question is how far off from the theoretical optimum we are: at the moment it is not clear whether we are fighting for the last few percent or if sizeable gains can still be made. There are many opportunities left for improvement, both in speed and accuracy. An easy option is to try to squeeze out as much as possible from the current framework: scoring equivalence classes with the BDe metric should bring a significant performance gain for large structures, without any obvious drawback, as the likelihood contributions of all members are aggregated anyway.

A hopeful but ambitious path is to tackle some of the fundamental problems: as stated, we would like to score MAGs directly instead of having to go via uDAGs. This would improve both the mapping and the probability estimates of the inferred logical causal statements. We can try to include higher order independencies (larger substructures) through sampling: reasonable probability estimate can be obained from a limited number of high scoring alternatives, see, e.g. (Bouckaert, 1995). Finally, we would like to obtain principled probability estimates for new statements derived during the inference process: this would improve conflict resolution, and would ultimately allow to give a meaningful estimate for the probability of all causal relations inferred from a given data set.

Acknowledgements

This research was supported by NWO Vici grant nr.639.023.604.

References

R. Bouckaert. *Bayesian Belief Networks: From Construction to Inference*. PhD thesis, University of Utrecht, 1995.

W. Buntine. Theory refinement on Bayesian networks. In *Proc. of the 7th Conference on Uncertainty in Artificial Intelligence*, pages 52–60, Los Angeles, CA, 1991. Morgan Kaufmann.

D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(3):507–554, 2002.

F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Annals of Combinatorics*, 6(2):125–145, 2002.

T. Claassen and T. Heskes. Supplement to 'A Bayesian approach to constraint based causal inference'. Technical report, 2012. `http://www.cs.ru.nl/~tomc/docs/BCCD_Supp.pdf`.

T. Claassen and T. Heskes. A logical characterization of constraint-based causal discovery. In *Proc. of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

D. Colombo, M. Maathuis, M. Kalisch, and T. Richardson. Learning high-dimensional dags with latent and selection variables (uai2011). Technical report, ArXiv, Zurich, 2011.

G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

R. Evans and T. Richardson. Maximum likelihood fitting of acyclic directed mixed graphs to binary data. In *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence*, 2010.

C. Glymour, R. Scheines, P. Spirtes, and J. Ramsey. The TETRAD project: Causal models and statistical data. `www.phil.cmu.edu/projects/tetrad/current`, 2004.

D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

D. Heckerman, C. Meek, and G. Cooper. A Bayesian approach to causal discovery. In *Computation, Causation, and Discovery*, pages 141–166. 1999.

M. Kalisch, M. Mächler, D. Colombo, M. Maathuis, and P. Bühlmann. Causal inference using graphical models with the R package pcalg. `http://cran.r-project.org/web/packages/pcalg/vignettes/pcalgDoc.pdf`, 2011.

D. Margaritis and F. Bromberg. Efficient Markov network discovery using particle filters. *Computational Intelligence*, 25(4):367–394, 2009.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 12*, pages 505–511, 1999.

C. Meek. Causal inference and causal explanation with background knowledge. In *UAI*, pages 403–410. Morgan Kaufmann, 1995.

G. Melancon, I. Dutour, and M. Bousquet-M'elou. Random generation of DAGs for graph drawing. Technical Report INS-R0005, Centre for Mathematics and Computer Sciences, 2000.

R. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 1st edition, 2004.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman Publishers, San Mateo, CA, 1988.

J. Pearl and T. Verma. A theory of inferred causation. In *Knowledge Representation and Reasoning: Proc. of the Second Int. Conf.*, pages 441–452, 1991.

J. Pellet and A. Elisseef. Using Markov blankets for causal structure learning. *Journal of Machine Learning Research*, 9:1295–1342, 2008.

J. Ramsey, J. Zhang, and P. Spirtes. Adjacency-faithfulness and conservative causal inference. In *Proc. of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 401–408, 2006.

T. Richardson and P. Spirtes. Ancestral graph Markov models. *Ann. Stat.*, 30(4):962–1030, 2002.

R. Silva and Z. Ghahramani. The hidden life of latent variables: Bayesian learning with mixed graph models. *Journal of Machine Learning Research*, 10: 1187–1238, 2009.

P. Spirtes. Introduction to causal inference. *Journal of Machine Learning Research*, 11:1643–1662, 2010.

P. Spirtes, C. Meek, and T. Richardson. An algorithm for causal inference in the presence of latent variables and selection bias. In *Computation, Causation, and Discovery*, pages 211–252. 1999.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, Cambridge, Massachusetts, 2nd edition, 2000.

J. Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873 – 1896, 2008.

J. Zhang and P. Spirtes. Detection of unfaithfulness and robust causal inference. *Minds and Machines*, 2(18):239–271, 2008.

Scaling Up Decentralized MDPs Through Heuristic Search

Jilles S. Dibangoye
INRIA
Loria, Campus Scientifique - BP 239
54506 Vandœuvre-lès-Nancy, France
jilles.dibangoye@inria.fr

Christopher Amato
Computer Science and AI Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
camato@csail.mit.edu

Arnaud Doniec
Université Lille Nord de France
Mines Douai, Département IA
F-59500 Douai, France
arnaud.doniec@mines-douai.fr

Abstract

Decentralized partially observable Markov decision processes (Dec-POMDPs) are rich models for cooperative decision-making under uncertainty, but are often intractable to solve optimally (NEXP-complete). The transition and observation independent Dec-MDP is a general subclass that has been shown to have complexity in NP, but optimal algorithms for this subclass are still inefficient in practice. In this paper, we first provide an updated proof that an optimal policy does not depend on the histories of the agents, but only the local observations. We then present a new algorithm based on heuristic search that is able to expand search nodes by using constraint optimization. We show experimental results comparing our approach with the state-of-the-art Dec-MDP and Dec-POMDP solvers. These results show a reduction in computation time and an increase in scalability by multiple orders of magnitude in a number of benchmarks.

1 Introduction

There has been substantial progress on algorithms for multiagent sequential decision making represented as decentralized partially observable Markov decision processes (Dec-POMDPs) [18, 20, 7, 2]. Algorithms that are able to exploit domain structure when it is present have been particularly successful [24, 1, 25]. Unfortunately, because the general Dec-POMDP problem is NEXP-complete [8], even these methods cannot solve moderately sized problems optimally.

The decentralized Markov decision process (Dec-MDP) with independent transitions and observations represents a general subclass of Dec-POMDPs that has complexity in NP rather than NEXP [5]. A few algorithms for solving this Dec-MDP subclass have been recently proposed [5, 21, 22]. While these approaches can often solve much larger problems than Dec-POMDP methods, they cannot solve truly large problems or those with more than 2 agents.

In this paper, we present a novel algorithm for optimally solving Dec-MDPs with independent transitions and observations that combines heuristic search and constraint optimization. We show that one can cast any Dec-MDP with independent transitions and observations as a continuous deterministic MDP where states are probability distributions over states in the original Dec-MDP, which we call state occupancy distributions. This allows us to adapt continuous MDP techniques [6, 4] to solve decentralized MDPs. Following this insight, we designed an algorithm where the state occupancy exploration is performed similarly to learning real-time A* [13] and the policy selection is in accordance with decentralized POMDP techniques [10, 15]. The result is an approach that is able to leverage problem structure through heuristics, limiting the space of policies that are explored by bounding their value and efficiently generating policies with the use of constraint optimization. This algorithm (termed Markov policy search or MPS), is shown to be a much more efficient algorithm than any other approach that can be used in Dec-MDPs with independent transitions and observations.

The remainder of this paper is organized as follows. First, we provide some motivating examples utilizing properties of Dec-MDPs with independent transitions and observations. Next, we describe the Dec-MDP framework and discuss the related work. We then present theoretical results, showing that the optimal policy for Dec-MDPs with independent transitions and observations does not depend on the agent histories. While this has been proven before, we offer a more general proof that permits additional insights. Next, we describe the decentralized Markov policy search algorithm, which combines constraint optimization and heuristic search to more efficiently produce optimal solutions for Dec-MDPs with independent transitions and observations. Finally, we present an empirical evaluation of this algorithm with respect to the state-of-the-art solvers that apply in decentralized MDPs, showing the ability to

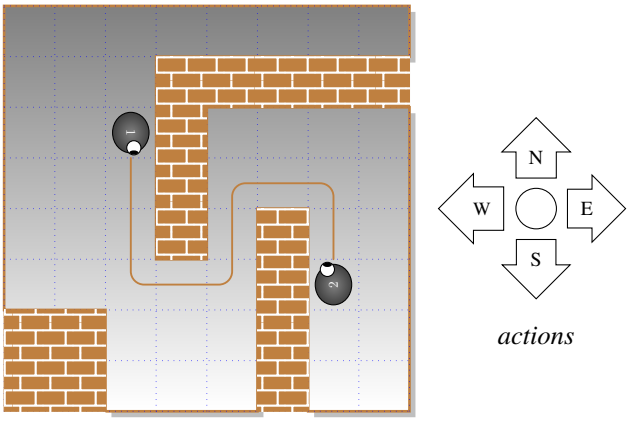


Figure 1: A meeting-grid under uncertainty scenario on a 8×8 grid, inspired from Seuken and Zilberstein [24].

solve problems that are multiple orders of magnitude larger and those that include up to 10 agents.

2 Motivating examples

To illustrate the characteristics of decentralized partially observable Markov decision processes (Dec-POMDPs) that we are interested in, consider a simple two-agent "meeting-in-a-grid under uncertainty" domain in Figure 1. In this scenario, two agents want to meet as soon as possible on a two-dimensional grid. In this world, each agent's possible actions include moving north, south, west, east and staying in the same place. The actions of a given agent do not affect the other agents. After taking an action, each agent can sense some information, which in this case corresponds to its own location. Here, each agent's own partial information is insufficient to determine the global state of the world. This is mainly because agents are not permitted to explicitly communicate their local locations with each other. However, if this (instantaneous and noise-free) communication were allowed the agents' partial information together would reveal the true state of the world, (i.e., the agents' joint location). It is the presence of this *joint full observability* property that differentiates Dec-MDPs from Dec-POMDPs.

More generally, in partially observable models including Dec-POMDPs, the agents' partial information together can map to multiple different states of the world. As a consequence, decisions in such models depend on the entire past histories of actions and observations that the agents ever experienced. In the meeting-in-a-grid under uncertainty problem, since both transitions and observations are not affected by the other agents, each agent's decision depends only on its last piece of partial information, (i.e., the agent's own location) [5]. These characteristics appear in many real-world applications including:

Mars exploration rovers. The meeting-in-a-grid domain was motivated by a real problem of controlling the operation of multiple space exploration rovers, such as the ones used by NASA to explore the surface of Mars [27].

Distributed sensor net surveillance. The sensor net domain [17], where a team of stationary or moveable UAVs, satellites, or other sensors must coordinate to track targets while sensors have independent transitions and observations, is particularly suited to our model.

Distributed smart-grid domains. This application aims at finding the optimal schedules and amounts of generated power for a collection of generating units, given demands, and operational constraints over a time horizon.

3 Background and Related Work

In this section, we review the decentralized MDP model, the assumptions of transition and observation independence, the associated notation, and related work.

Definition 1 (The decentralized MDP) *A n-agent decentralized MDP (S, A, p, r) consists of:*

- *A finite set $S = Z^1 \times Z^2 \times \cdots Z^n$ of states $s = \langle z^1, z^2, \cdots, z^n \rangle$, where Z^i denotes the set of local observations z^i of agent $i = 1, 2, \ldots, n$.*
- *A finite set $A = A^1 \times A^2 \times \cdots A^n$ of joint actions $a = \langle a^1, a^2, \cdots, a^n \rangle$, where A^i is the set of local actions a^i of agent $i = 1, 2, \ldots, n$.*
- *A transition function $p(s, a, s')$, which denotes the probability of transiting from state $s = \langle z^1, z^2, \ldots, z^n \rangle$ to state $s' = \langle z'^1, z'^2, \ldots, z'^n \rangle$ when taking joint action $a = \langle a^1, a^2, \ldots, a^n \rangle$*
- *A reward function $r\colon S \times A \mapsto \mathbb{R}$, where $r(s, a)$ denotes the reward received when executing joint action a in state s*

As noted above, decentralized MDPs are distinguished by the state being *jointly fully observable*. This property ensures that the global state would be known if all agents shared their observations at a given step (i.e., there is no external uncertainty in the problem) and follows trivially from the definition of states as observations for each agent. The Dec-MDP is parameterized by the initial state distribution η_0. When the agents operate over a bounded number of steps (typically referred to as the problem horizon) T, the model is referred to as a finite-horizon decentralized MDP. Solving a decentralized MDP for a given planning horizon T and start state distribution η_0 can be seen as finding n individual *policies* that maximize the expected cumulative reward over the steps of the problem.

3.1 Additional Assumptions

We are interested in decentralized MDPs that exhibit two properties. The first is the transition independence assumption where the local observation of each agent depends only on its previous local observation and the local action taken by that agent.

Definition 2 (The transition independent assumption) *An n-agent decentralized MDP is said to be transition independent if there exists local transition functions $p^1 \colon Z^1 \times A^1 \times Z^1 \mapsto [0,1]$, $p^2 \colon Z^2 \times A^2 \times Z^2 \mapsto [0,1]$, $\ldots$, $p^n \colon Z^n \times A^n \times Z^n \mapsto [0,1]$ such that*

$$p(s,a,s') \quad = \quad \prod_{i=1,\ldots,n} p^i(z^i, a^i, z'^i),$$

where $s = \langle z^1, z^2, \ldots, z^n \rangle$ and $s' = \langle z'^1, z'^2, \ldots, z'^n \rangle$ and $a = \langle a^1, a^2, \ldots, a^n \rangle$.

We also implicitly assume observation independence, which states that the observation function of each agent does not depend on the dynamics of the other agents. That is, $P(z'^1, z'^2, \ldots, z'^n | s, a^1, a^2, \ldots, a^n) = \times_i \; P(z'^i | s, a^i)$. Because we are assuming a Dec-MDP with the state factored into local observations then this becomes the same as transition independence: $\prod_i \; P(z'^i | z^i, a^i)$.

3.2 Preliminary Definitions and Notations

The goal of solving a Dec-MDP is to find a decentralized deterministic joint policy $\pi = \langle \pi^1, \ldots, \pi^n \rangle$. An individual policy π^i is a sequence of decision rules $\pi^i = \langle \sigma^i_0, \ldots, \sigma^i_{T-1} \rangle$. In addition, we call decentralized decision rule σ_τ at time τ an n-tuple of decision rules $(\sigma^1_\tau, \ldots, \sigma^n_\tau)$, for $\tau = 0, 1, \ldots, T-1$. In this paper, we distinguish between history-dependent and Markov decision rules.

Each *history-dependent decision rule* σ^i_τ at time τ maps from τ-step local action-observation histories $h^i_\tau = \langle a^i_0, z^i_1, \ldots, a^i_{\tau-1}, z^i_\tau \rangle$ to local actions: $\sigma^i_\tau(h^i_\tau) = a^i_\tau$, for $\tau = 0, 1, \ldots, T-1$. A sequence of history-dependent decision rules defines a history-dependent policy.

In contrast, each *Markov decision rule* σ^i_τ at time τ maps from local observations z^i_τ to local actions: $\sigma^i_\tau(z^i_\tau) = a^i_\tau$, for $\tau = 0, 1, \ldots, T-1$. A sequence of Markov decision rules defines a Markov policy. Moreover, it is worth noticing that decentralized Markov policies are exponentially smaller than decentralized history-dependent ones.

The *state occupancy* is another important notion in this paper. The τ-th state occupancy of a system under the control of a decentralized Markov policy $\langle \sigma_0, \sigma_1, \cdots, \sigma_{\tau-1} \rangle$, denoted $\sigma_{0:\tau-1}$, and starting at η_0 is given by: $\eta_\tau(s) = P(s | \sigma_{0:\tau-1}, \eta_0)$, for all $\tau \geq 1$. Moreover, the current state occupancy η_τ depends on the past decentralized Markov policy $\sigma_{0:\tau-1}$ only through previous state occupancy $\eta_{\tau-1}$ and decentralized Markov decision rule $\sigma_{\tau-1}$. That is, $\eta_\tau(s') = \sum_s p(s, \sigma_{\tau-1}(s), s') \cdot \eta_{\tau-1}(s)$, for all $\tau \geq 1$. Following [11], this update-rule is denoted $\eta_\tau = \chi_\tau(\eta_{\tau-1}, \sigma_{\tau-1})$ for the sake of simplicity. We also denote $\triangle_\tau$ the state occupancy space at the τ-th horizon, that is the standard $|S|$-dimensional simplex.

Distinction with belief states. The state occupancy may be thought of as a belief state, but there are differences. Formally, a belief state b_τ is given by $b_\tau(s) = P(s | h_\tau, \sigma_{0:\tau-1}, \eta_0)$, for all $\tau \geq 1$ [3]. That is, in belief states, the information agents have about states is typically conditioned on a single joint action-observation history h_τ. From the total probability property, we then have that $\eta_\tau(s) = \sum_{h_\tau} P(s | h_\tau, \sigma_{0:\tau-1}, \eta_0) \cdot P(h_\tau | \sigma_{0:\tau-1}, \eta_0)$. Overall, the τ-th state occupancy summarizes all the information about the world states contained in all belief states at horizon τ. In other words, the doubly exponentially joint action-observation histories are summarized in a single state occupancy that does not make use of local information.

3.3 Related Work

In this section, we focus on approaches for solving Dec-MDPs with independent transitions and observations as well as other relevant solution methods. For a thorough introduction to solution methods in Dec-POMDPs, the reader can refer to [24, 20, 7, 2].

Becker et al. [5] were the first to describe the transition and observation independent Dec-MDP subclass and solve it optimally. Their approach, called the coverage set algorithm, consists of three main steps. First, sets of augmented MDPs are created which incorporate the joint reward into local reward functions for each agent. Then, all best responses for any of the other agent policies are found using these augmented MDPs. Finally, the joint policy that has the highest value from all agents' best responses is returned. While this algorithm is optimal, it keeps track of the complete set of policy candidates for each agent, requiring a large amount of time and memory.

Petrik and Zilberstein [22] reformulated the coverage set algorithm as a bilinear program, thereby allowing optimization approaches to be utilized. The bilinear program can be used as an anytime algorithm, providing online bounds on the solution quality at each iteration. The representation is also better able to take advantage of sparse joint reward distributions by representing independent rewards as linear terms and compressing the joint reward matrix. This results in greatly increased efficiency in many cases, but when the agents' rewards often depend on the other agents the bilinear program can still be inefficient due to lack of reward sparsity.

In general Dec-POMDPs, approximate approaches have at-

tempted to scale to larger problems and horizons by not generating the full set of policies that may be optimal. These approaches, known as memory-bounded algorithms, were introduced by Seuken and Zilberstein [24] and then successively refined [10, 15]. Memory-bounded algorithms sample forward a bounded number of belief states, and back up (i.e., generate next step policies for) one decentralized history-dependent policy for each belief state. To avoid the explicit enumeration of all possible policies, Kumar and Zilberstein [15] perform the backup by solving a corresponding constraint optimization problem (COP) [9], that represents the decentralized backup. Although, memory-bounded techniques are suboptimal, the decentralized backup can be applied in exact settings as we demonstrate in our algorithm.

More specifically, the decentralized backup can build a horizon-τ decentralized policy that is maximal with respect to a belief state and horizon-$(\tau + 1)$ policies available for each agent. The associated COP is given by: a set of variables, one for each local observation of each agent; a set of domains, where the domain for the variables corresponding to an agent is the set of horizon-$(\tau + 1)$ policies available for that agent; a set of soft constraints, one for each joint observation. The soft constraint maps assignments to real values. Intuitively, these values represent the expected reward accrued when agents together perceive a given joint observation and follow a given horizon-$(\tau + 1)$ decentralized policy. Since horizon-τ decentralized policies consist of horizon-$(\tau+1)$ policies, it is easy to see that maximizing the sum of the soft constraints yields a maximal horizon-τ decentralized policy.

Closer to our model is the ND-POMDP framework [19]. It aims at modeling multiagent teamwork where agents have strong locality of interaction, often through binary interactions. That is, the reward model in such domains is decomposed among sets of agents. There has been a substantial body of work that extend general Dec-POMDP techniques (discussed above) to exploit the locality of interaction [19, 14, 16]. Nair *et* al. [19] introduced the only optimal algorithm for this model, namely the General Optimal Algorithm (GOA). When the domain does not contain binary interactions, there is no reason to expect GOA to outperform general Dec-POMDP algorithms, as all methods use similar strategies in selecting policy candidates. However, when the domain contains primarily binary interactions (or more generally when each agent's rewards are not dependent on many other agents), GOA is likely to outperform general Dec-POMDP algorithms.

It is worth noting that ND-POMDPs and transition and observation independent Dec-MDPs make the same assumptions about transition and observation independence, but make different assumptions about the reward model and partial observability. More specifically, ND-POMDPs assume the reward can be decomposed into the sum of local reward models for sets of agents, while the reward model for transition and observation independent Dec-MDPs is more general, allowing global rewards for all agents (i.e., considering all agents to be in one set). Dec-MDPs assume that the state is jointly fully observable (i.e., that the state is fully determined by the combination of local observations of all agents), while ND-POMDPs do not make this limiting assumption. Both models therefore make different assumptions to address complexity and the choice of model depends on which assumptions best match the domain being solved.

4 Theoretical Properties

In this section, we demonstrate the main theoretical results of this paper.

4.1 Optimal Policies

A decentralized MDP solver aims to calculate an optimal decentralized policy π^* that maximizes the expected cumulative reward:

$$\pi^* \quad = \quad \arg\max_\pi E[\sum_{\tau=0}^{T-1} r(s_\tau, a_\tau)|\pi, \eta_0]. \qquad (1)$$

The following theorem proves that decentralized Markov policies yield the optimal performance in decentralized MDPs with independent transitions and observations. Goldman *et* al. [12] established the optimality of Markov policy for an agent under the assumption that the other agents choose Markov policies. Here, we state the optimality of Markov policies for an agent no matter what its teammates' policies are. We also construct the proof in a manner that more directly relates policies to values (rather than information sets). This may be more clear to some readers.

Theorem 1 (Optimality of decentralized Markov policies) *In Dec-MDPs with independent transitions and observations, optimal policies for each agent depend only on the local state and not on agent histories.*

Proof Without loss of generality, we construct a proof by induction for two agents, 1 and 2, from agent 1's perspective. We first show that in the last step of the problem, agent 1's policy does not depend on its local history.

Agent 1's local policy on the last step is: $\sigma^{1^*}_{T-1}(h^1_{T-1}) = \arg\max_{a^1} \sum_{h^2_{T-1}} P(h^2_{T-1}|h^1_{T-1}) \cdot R(s, a^1, \sigma^2_{T-1}(h^2_{T-1}))$, which chooses a local action to maximize value based on the possible local histories of agent 2 and resulting states of the system $s = \langle z^1_{T-1}, z^2_{T-1}\rangle$.

Based on transition and observation independence and the use of decentralized policies, it can be shown that $P(h^2_{T-1}|h^1_{T-1}) = P(h^2_{T-1})$. Due to space limitations, we

do not include full proof of this claim. Intuitively it holds because each agent does not receive any information about the other agents' local histories due to transition independence. Therefore, we can represent agent 1's policy on the last step as $\sigma^{1^*}_{T-1}(h^1_{T-1}) = \arg\max_{a^1} \sum_{h^2_{T-1}} P(h^2_{T-1}) \cdot R(s, a^1, \sigma^2_{T-1}(h^2_{T-1}))$ which no longer depends on the history h^1_{T-1}. Therefore, the policy on the last step for either agent does not depend on history.

This allows us to define the value function on the last step as $\upsilon_{T-1}(s, \sigma^1_{T-1}(z^1_{T-1}), \sigma^2_{T-1}(z^2_{T-1}))$.

Then for the induction step, we can show that if the policy at step $\tau + 1$ does not depend on history, then the policy at step τ also does not depend on its local history. Again, we show this from agent 1's perspective.

Agent 1's policy on step τ can be represented by: $\sigma^{1^*}_{\tau}(h^1_{\tau}) = \arg\max_{a^1} \sum_{h^2_{\tau}} P(h^2_{\tau}|h^1_{\tau}) \cdot \upsilon_{\tau+1}(s, a^1, \sigma^2_{\tau}(h^2_{\tau}))$, where the value function $\upsilon_{\tau+1}$ is assumed to not depend on history. We can again show that $P(h^2_{\tau}|h^1_{\tau}) = P(h^2_{\tau})$ because of transition independence and represent agent 1's policy on step τ as:

$$\sigma^{1^*}_{\tau}(h^1_{\tau}) = \arg\max_{a^1} \sum_{h^2_{\tau}} P(h^2_{\tau}) \cdot \upsilon_{\tau+1}(s, a^1, \sigma^2_{\tau}(h^2_{\tau}))$$

which no longer depends on the local history h^1_{τ}.

Therefore, the policy of either agent does not depend on local history for any step of the problem. ∎

We now establish the sufficient statistic for the selection of decentralized Markov decision rules.

Theorem 2 (Sufficient Statistic) *The state occupancy is a sufficient statistic for decentralized Markov decision rules.*

Proof We build upon the proof of the optimality of decentralized Markov policies in Theorem 1. We note that an optimal decentralized Markov policy starting in η_0 is given by:

$$\pi^* = \arg\max_{\pi} \sum_{\tau} \sum_{h_{\tau}} P(h_{\tau}|\sigma_{0:\tau-1}, \eta_0) \cdot r(s_{\tau}, \sigma_{\tau}[s_{\tau}])$$

The substitution of h_{τ} by $(h_{\tau-1}, a_{\tau-1}, s_{\tau})$ plus the sum over all pairs $(h_{\tau-1}, a_{\tau-1})$ yields

$$\pi^* = \arg\max_{\pi} \sum_{\tau} \sum_{s_{\tau}} P(s_{\tau}|\sigma_{0:\tau-1}, \eta_0) \cdot r(s_{\tau}, \sigma_{\tau}[s_{\tau}]),$$

We denote $\eta^{\pi}_{\tau} = P(s_{\tau}|\sigma_{0:\tau-1}, \eta_0)$ the state occupancy distribution that decentralized Markov policy π produced at horizon τ. And hence,

$$\pi^* = \arg\max_{\pi} \sum_{\tau} \sum_{s_{\tau} \in S} \eta^{\pi}_{\tau}(s_{\tau}) \cdot r(s_{\tau}, \sigma_{\tau}[s_{\tau}])$$

So, state occupancy η^{π}_{τ} summarizes all possible joint action-observation histories h_{τ} decentralized Markov policy π produced at horizon τ for the estimate of joint decision rule σ_{τ}. Thus, the state occupancy is a sufficient statistic for decentralized Markov decision rules since their estimates depend only upon a state occupancy, and no longer on all possible joint observation-histories. ∎

States, belief states, and multi-agent belief states are all sufficient to select directly actions for MDPs, POMDPs, and decentralized POMDPs, respectively. This is mainly because all these statistics summarize the information about the world states from a single agent perspective. The state occupancy, instead, summarizes the information about the world states from the perspective of a team of agents that are constrained to execute their policies independently from each other. In such a setting, joint actions cannot be selected independently, instead, they are selected jointly through decentralized Markov decision rules.

4.2 Optimality Criterion

This section presents the optimality criterion based on the policy value functions.

We first define *the τ-th expected immediate reward function* $r_{\tau}(\cdot, \sigma_{\tau}) \colon \triangle_{\tau} \mapsto \mathbb{R}$ that is given by $r_{\tau}(\eta_{\tau}, \sigma_{\tau}) = E_{s \sim \eta_{\tau}}[r(s, \sigma_{\tau}[s])]$. This quantity denotes the immediate reward of taking decision rule σ_{τ} when the system is in state occupancy η_{τ} at the τ-th time step.

Let $\upsilon_{\pi}(\eta_0)$ represent *the expected total reward* over the decision making horizon if policy π is used and the system is in state occupancy η_0 at the first time step $\tau = 0$. For π in the space of decentralized Markov policies, the expected total reward is given by:

$$\upsilon_{\pi}(\eta_0) \equiv E_{(\eta_1, \cdots, \eta_{T-1})} \left[\sum_{\tau=0}^{T-1} r_{\tau}(\eta_{\tau}, \sigma_{\tau}) \mid \eta_0, \pi \right]$$

We say that a decentralized Markov policy π^* is *optimal* under the total reward criterion whenever $\upsilon_{\pi^*}(\eta_0) \geq \upsilon_{\pi}(\eta_0)$ for all decentralized Markov policies π.

Following the Bellman principle of optimality [23], one can separate the problem of finding the optimal policy π^* into simpler subproblems. Each of these subproblems consists of finding policies $\sigma_{\tau:T-1}$ that are optimal for all $\tau = 0, \cdots, T-1$. To do so, we then define the *τ-th value function* $\upsilon_{\sigma_{\tau:T-1}} \colon \triangle_{\tau} \mapsto \mathbb{R}$ under the control of decentralized Markov policy $\sigma_{\tau:T-1}$ as follows:

$$\upsilon_{\sigma_{\tau:T-1}}(\eta_{\tau}) = r_{\tau}(\eta_{\tau}, \sigma_{\tau}) + \upsilon_{\sigma_{\tau+1:T-1}}(\chi_{\tau+1}(\eta_{\tau}, \sigma_{\tau}))$$

where quantity $\upsilon_{\sigma_{\tau:T-1}}(\eta_{\tau})$ denotes the expected sum of rewards attained by starting in state occupancy η_{τ}, taking one joint action according to σ_{τ}, taking the next joint action according to $\sigma_{\tau+1}$, and so on. We slightly abuse notation and write the τ-th value function under the control of an "unknown" decentralized Markov policy $\sigma_{\tau:T-1}$ using $\upsilon_{\tau} \colon \triangle_{\tau} \mapsto \mathbb{R}$.

We further denote $\mathscr{V}_{\tau}$ to be the space of bounded value functions at the τ-th horizon. For each $\upsilon_{\tau+1} \in \mathscr{V}_{\tau+1}$, and decentralized Markov decision rule σ_{τ}, we define the *linear transformation* $\mathscr{L}_{\sigma_{\tau}} \colon \mathscr{V}_{\tau+1} \mapsto \mathscr{V}_{\tau}$ by

$$[\mathscr{L}_{\sigma_{\tau}} \upsilon_{\tau+1}](\eta_{\tau}) = r_{\tau}(\eta_{\tau}, \sigma_{\tau}) + \upsilon_{\tau+1}(\chi_{\tau+1}(\eta_{\tau}, \sigma_{\tau})).$$

As such, the τ-th value function υ_τ can be built from a $(\tau+1)$-th value function $\upsilon_{\tau+1}$ as follows:

$$\begin{array}{rcl} \upsilon_\tau(\eta_\tau) & = & \max_{\sigma_\tau}\ [\mathscr{L}_{\sigma_\tau}\upsilon_{\tau+1}](\eta_\tau), \\ \upsilon_T(\eta_T) & = & 0. \end{array} \tag{2}$$

In our setting, Equations (2) denote the optimality equations. It is worth noting that the decentralized Markov policy solution $\pi = \langle \sigma_0, \cdots, \sigma_{T-1}\rangle$ of the optimality equations is greedy with respect to value functions $\upsilon_0, \ldots, \upsilon_{T-1}$.

5 Markov Policy Search

In this section, we compute optimal decentralized Markov policy $\langle \sigma_0^*, \cdots, \sigma_{T-1}^*\rangle$ given initial state occupancy η_0 and planning horizon T. Note that while state occupancies are used to calculate heuristics in this algorithm, the final choices at each step do not depend on the state occupancies. That is, the result is a nonstationary policy for each agent mapping local observations to actions at each step.

We cast decentralized MDPs (S, A, p, r) as continuous and deterministic MDPs where: states are state occupancy distributions η_τ; actions are decentralized Markov policies σ_τ; the update-rules $\chi_\tau(\cdot, \sigma_{\tau-1})$ define transitions; and mappings $r_\tau(\cdot, \sigma_\tau)$ denote the reward function. So, techniques that apply in continuous and deterministic MDPs also apply in decentralized MDPs with independent transitions and observations. For the sake of efficiency, we focus only on optimal techniques that exploit the initial information η_0.

The *learning real-time A** (LRTA*) algorithm can be used to solve deterministic MDPs [13]. This approach updates only states that agents actually visit during the planning stage. Therefore, it is suitable for continuous state spaces. Algorithm 1, namely *Markov Policy Search* (MPS), illustrates an adaptation of the LRTA* algorithm for solving decentralized MDPs with independent transitions and observations. The MPS algorithm relies on lower and upper bounds $\underline{\upsilon}_\tau$ and $\bar{\upsilon}_\tau$ on the exact value functions for all planning horizons $\tau = 0, \ldots, T-1$.

We use the following definitions. Q-value functions $\bar{q}_\tau(\eta_\tau, \sigma_\tau)$ denote rewards accrued after taking decision rule σ_τ at state occupancy η_τ and then following the policy defined by upper-bound value functions for the remaining planning horizons. We denote $\Psi_\tau(\eta_\tau) = \{\sigma_\tau\}$ to be the set of all stored decentralized Markov decision rules for state occupancy η_τ. Thus, $\bar{\upsilon}_\tau(\eta_\tau) = \max_{\sigma_\tau \in \Psi_\tau(\eta_\tau)} \bar{q}_\tau(\eta_\tau, \sigma_\tau)$ represents the upper-bound value at state occupancy η_τ. Formally, we have that $\bar{q}_\tau(\eta_\tau, \sigma_\tau) = [\mathscr{L}_{\sigma_\tau}\bar{\upsilon}_{\tau+1}](\eta_\tau)$.

Next, we describe two variants of the MPS algorithm. The exhaustive variant replaces states by state occupancy distributions, and actions by decentralized Markov decision rules in the LRTA* algorithm. The second variant uses a constraint optimization program instead of the memory demanding exhaustive backup operation that both the LRTA* algorithm and the exhaustive variant use.

5.1 The exhaustive variant

The exhaustive variant consists of three major steps: the initialization step (line 1); the backup operation step (line 5); and the update step (lines 6 and 8). It repeats the execution of these steps until convergence ($\bar{\upsilon}_0(\eta_0) - \underline{\upsilon}_0(\eta_0) \leq \varepsilon$). At this point, an ϵ-optimal decentralized Markov policy has been found.

Algorithm 1: The MPS algorithm.

```
begin
1     Initialize bounds υ̲ and ῡ.
2     while ῡ_0(η_0) − υ̲_0(η_0) > ε do
3         MPS-TRIAL(η_0)

MPS-TRIAL(η_τ) begin
4     while ῡ_τ(η_τ) − υ̲_τ(η_τ) > ε do
5         σ_greedy,τ ← arg max_{σ_τ} q̄_τ(η_τ, σ_τ)
6         Update the upper bound value function.
7         MPS-TRIAL(χ_{τ+1}[η_τ, σ_greedy,τ])
8         Update the lower bound value function.
```

Initialization. We initialize lower bound $\underline{\upsilon}_\tau$ with the τ-th value function of any decentralized Markov policy, such as a randomly generated policy $\pi_{\text{rand}} = \langle \sigma_{\text{rand},0}, \ldots, \sigma_{\text{rand},T-1}\rangle$, where $\underline{\upsilon}_\tau = \upsilon_{\sigma_{\text{rand},\tau}, \ldots, \sigma_{\text{rand},T-1}}$. We initialize the upper bound $\bar{\upsilon}_\tau$ with the τ-th value function of the underlying MDP. That is, $\pi_{\text{mdp}} = \langle \sigma_{\text{mdp},0}, \ldots, \sigma_{\text{mdp},T-1}\rangle$, where $\bar{\upsilon}_\tau = \upsilon_{\sigma_{\text{mdp},\tau}, \ldots, \sigma_{\text{mdp},T-1}}$.

The exhaustive backup operation. We choose decentralized Markov decision rule $\sigma_{\text{greedy},\tau}$, which yields the highest value $\bar{\upsilon}_\tau(\eta_\tau)$ through the explicit enumeration of all possible decentralized Markov decision rules σ_τ. We first store all decentralized Markov decision rules σ_τ for each visited state occupancy η_τ together with corresponding values $\bar{q}_\tau(\eta_\tau, \sigma_\tau)$. Hence, the greedy decentralized Markov decision rule $\sigma_{\text{greedy},\tau}$ is $\arg\max_{\sigma_\tau} \bar{q}_\tau(\eta_\tau, \sigma_\tau)$ at state occupancy η_τ.

Update of lower and upper bounds. We update the lower bound value function based on decentralized Markov policies $\pi_{\text{greedy}} = \langle \sigma_{\text{greedy},0}, \ldots, \sigma_{\text{greedy},T-1}\rangle$ selected at each trial. If π_{greedy} yields a value higher than that of the current lower bound, $\underline{\upsilon}_0(\eta_0) < \upsilon_{\pi_{\text{greedy}}}(\eta_0)$, we set $\underline{\upsilon}_\tau = \upsilon_{\sigma_{\text{greedy},\tau}, \ldots, \sigma_{\text{greedy},T-1}}$ for $\tau = 0, \ldots, T-1$, otherwise we leave the lower bound unchanged. We update the upper bound value function based on decentralized Markov decision rules $\sigma_{\text{greedy},\tau}$ and the $(\tau+1)$-th upper-bound value function $\bar{\upsilon}_{\tau+1}$, as follows $\bar{\upsilon}_\tau(\eta_\tau) = [\mathscr{L}_{\sigma_{\text{greedy},\tau}}\bar{\upsilon}_{\tau+1}](\eta_\tau)$.

Theoretical guarantees. The exhaustive variant of MPS yields both advantages and drawbacks. On the one hand, it inherits the theoretical guarantees from the LRTA* al-

gorithm. In particular, it terminates with a decentralized Markov policy within $\varepsilon = \bar{v}_0(\eta_0) - \underline{v}_0(\eta_0)$ of the optimal decentralized Markov policy. Indeed, the upper bound value functions $\bar{v}_\tau$ never underestimate the exact value at any state occupancy η_τ. This is because we update the upper bound value at each state occupancy based upon a greedy decision rule for this state occupancy. On the other hand, the exhaustive variant algorithm requires the exhaustive enumeration of all possible decentralized Markov decision rules at each backup step (Algorithm 1, line 5). In MDP techniques, the exhaustive enumeration is not prohibitive since the action space is often manageable. In decentralized MDP planning, however, the space of all decentralized Markov decision rules increases exponentially with increasing observations and agents. As such, the exhaustive variant can scale only to problems with a moderate number of observations (local states) and two agents.

5.2 The constraint optimization formulation

To overcome the memory limitation of the exhaustive variant, we use constraint optimization instead of the exhaustive backup operation. More precisely, our constraint optimization program returns a greedy decentralized Markov decision rule $\sigma_{\text{greedy},\tau}$ for each state occupancy η_τ visited, but without performing the exhaustive enumeration.

In our constraint optimization formulation, variables are associated with decision rules $\sigma_\tau^i(z^i)$ for all agents $i = 1, \ldots, n$ and all local observations $z^i \in Z^i$. The domain for each variable $\sigma_\tau^i(z^i)$ is action space A^i. For each state $s \in S$, we associated a single soft constraint $c_\tau(s, \cdot)\colon A \mapsto \mathbb{R}$. Each of these assigns a value $c_\tau(s,a) = r(s,a) + \sum_{s'} p(s,a,s') \cdot v_{\sigma_{\text{mdp},\tau+1},\ldots,\sigma_{\text{mdp},T-1}}(s')$ to each joint action $a \in A$. Value $c_\tau(s,a)$ denotes the reward accrued at horizon τ when taking joint action a in state s and then following the underlying MDP joint policy for the remaining planning horizons. For each decentralized Markov decision rule $\sigma_\tau \in \Psi_\tau(\eta_\tau)$, we also associate a single soft constraint $g_\tau(\cdot)$. Each of these assigns value $g_\tau(\sigma_\tau) = \bar{q}_\tau(\eta_\tau, \sigma_\tau) - \bar{q}_{\text{mdp}}(\eta_\tau, \sigma_\tau)$, where $\bar{q}_{\text{mdp}}(\eta_\tau, \sigma_\tau) = [\mathscr{L}_{\sigma_\tau} v_{\sigma_{\text{mdp},\tau+1},\ldots,\sigma_{\text{mdp},T-1}}](\eta_\tau)$. The objective of our constraint optimization model is to find an assignment $\sigma_{\text{greedy},\tau}$ of actions a^i to variables $\sigma_\tau^i(z^i)$ such that the aggregate value is maximized. Stated formally, we wish to find $\sigma_{\text{greedy},\tau} = \arg\max_{\sigma_\tau} g_\tau(\sigma_\tau) + \sum_s \eta_\tau(s) c_\tau(s, \sigma_\tau(s))$. To better understand our constraint optimization program, note that by the definition of mapping $\bar{q}_{\text{mdp}}$ we have that $\sum_s \eta_\tau(s) \cdot c_\tau(s, \sigma_\tau(s)) = \bar{q}_{\text{mdp}}(\eta_\tau, \sigma_\tau)$. Hence, if we use $\bar{q}_{\text{mdp}}(\eta_\tau, \sigma_\tau)$ instead of $\sum_s \eta_\tau(s) \cdot c_\tau(s, \sigma_\tau(s))$, we get $\sigma_{\text{greedy},\tau} = \arg\max_{\sigma_\tau} \bar{q}_\tau(\eta_\tau, \sigma_\tau)$. Thus, our constraint optimization program returns a decentralized Markov decision rule with the highest upper-bound value. Techniques that solve our constraint optimization formulation abound in the literature of constraint programming [9], allowing many different approaches to be utilized.

Theoretical guarantees. The constraint optimization variant yields the same guarantees as the exhaustive variant without the major drawback of exhaustive enumeration of all decentralized Markov decision rules. Instead, it uses a constraint optimization formulation that returns a greedy decentralized Markov decision rule, which will often be much more efficient than exhaustive enumeration. And hence, we retain the property that stopping the algorithm at any time, the solution is within $\varepsilon = \bar{v}_0(\eta_0) - \underline{v}_0(\eta_0)$ of an optimal decentralized Markov policy.

Comparison to COP based algorithms. There is a rich body of work that replaces the exhaustive backup operation by a constraint optimization formulation in decentralized control settings [15, 14, 19]. These constraint optimization programs compute a decentralized history-dependent policy for a given belief state. While MPS also takes advantage of a constraint optimization formulation, it remains fundamentally different. The difference lies in both the COP formulation and the heuristic search. In existing COP based algorithms for decentralized control, authors try to find the best assignment of sub-policies to histories. Instead, in our case the COP formulation aims at mapping local observations to local actions. This provides considerable memory and time savings. Moreover, existing algorithms proceed by backing up policies in a backward direction (i.e., from last step to first) using a set pre-selected belief states. In contrast, the MPS algorithm proceeds forward, expanding the state occupancy distributions and selecting greedily decision rules. Finally, the MPS algorithm returns an optimal solution, whereas other COP based algorithms for Dec-POMDPs return only locally optimal solutions [15, 14]. Approximate solutions are returned by the other algorithms because they plan over (centralized) belief states, which do not constitute a sufficient statistic for Dec-POMDPs (or Dec-MDPs).

6 Empirical Evaluations

We evaluated our algorithm using several benchmarks from the decentralized MDP literature. For each benchmark, we compared our algorithms with state-of-the-art algorithms for solving Dec-MDPs and Dec-POMDPs. Note that we do not compare with ND-POMDP methods. Since our benchmarks allow all agents to interact with all teammates at all times, there is no reason to expect the optimal ND-POMDP method (GOA [19]) to outperform the algorithms presented here. We report on each benchmark the optimal value $v_0(\eta_0)$ together with the running time in seconds for different planning horizons.

The MPS variants were run on a Mac OSX machine with 2.4GHz Dual-Core Intel and 2GB of RAM available. We solved the constraint optimization problems using the aolib library[1]. The bilinear programming approach (listed as

[1]The aolib library is available at the following website:

T	$v_0(\eta_0)$	ICE	IPG	BLP	MPS	
					exh	*COP*
Recycling robot ($\|Z\| = 4$,$\|A\| = 9$)						
50	154.94	1.27	-	8848.7	0.016	0.5
60	185.71	6.00	-	-	0.090	0.555
70	216.47	28.6	-	-	0.111	0.395
80	247.24	-	-	-	0.124	0.545
90	278.01	-	-	-	0.151	0.373
100	308.78	-	-	-	0.156	0.438
1000	**3078.0**	-	-	-	**1.440**	**5.374**
Meeting Grid ($\|Z\| = 81$,$\|A\| = 25$)						
2	0.0	0.00	5	10.0	-	0.030
3	0.13	0.02	17	34.7	-	0.110
4	0.43	0.37	54	192.8	-	0.114
5	0.89	4.38	600	571.2	-	0.131
6	1.49	-	-	1160.6	-	0.159
10	4.68	-	-	3938.5	-	0.309
100	**94.26**	-	-	-	-	**13.12**
1000	**994.2**	-	-	-	-	**33.59**

T	$v_0(\eta_0)$	ICE	IPG	BLP	*COP*
Meeting on a 8x8 Grid ($\|Z\| = 4096$,$\|A\| = 25$)					
5	0.0	12.55	-	-	5.05
6	**0.0**	-	-	-	**6.20**
7	**0.71**	-	-	-	**13.16**
8	**1.67**	-	-	-	**13.76**
9	**2.68**	-	-	-	**16.94**
10	**3.68**	-	-	-	**18.66**
20	**13.68**	-	-	-	**38.37**
30	**23.68**	-	-	-	**52.39**
40	**33.68**	-	-	-	**59.70**
50	**43.68**	-	-	-	**74.28**
100	**93.68**	-	-	-	**214.73**
Navigation (MIT) ($\|Z\| = 7225$,$\|A\| = 16$)					
10	0.0	85.85	-	-	47.322
20	**0.0**	-	-	-	**321.26**
30	**14.28**	-	-	-	**180.70**
40	**34.97**	-	-	-	**400.48**
50	**54.92**	-	-	-	**1061.94**
100	**154.93**	-	-	-	**1236.11**

T	$v_0(\eta_0)$	ICE	IPG	BLP	*COP*
Navigation (ISR) ($\|Z\| = 8100$,$\|A\| = 16$)					
2	0.0	0.71	-	3225.8	2.39
3	0.0	6.50	-	-	3.19
4	**0.0**	-	-	-	**4.31**
5	**0.38**	-	-	-	**13.43**
10	**6.47**	-	-	-	**54.16**
50	**83.02**	-	-	-	**194.66**
100	**182.54**	-	-	-	**1294.28**
Navigation (PENTAGON) ($\|Z\| = 9801$,$\|A\| = 16$)					
2	0.0	0.99	-	4915.1	1.31
3	0.0	6.01	-	-	5.11
4	**0.0**	-	-	-	**8.75**
5	**0.38**	-	-	-	**13.81**
10	**4.82**	-	-	-	**62.89**
20	**19.73**	-	-	-	**129.48**
30	**39.18**	-	-	-	**209.11**
40	**57.75**	-	-	-	**276.20**
50	**76.39**	-	-	-	**1033.70**

Table 1: Experimental results for the *COP* and *exh.* variants of MPS as well as GMAA*-ICE (labeled ICE), IPG, and BLP.

BLP) was run on a 2.8GHz Quad-Core Intel Mac with 2GB of RAM with a time limit of 3 hours. We used the best available version of the bilinear program approach which was the iterative best response version with standard parameters. This is a generic solution method which does not perform as well as the more specialized approaches in [22], but we do not expect results to differ by more than a single order of magnitude. We do not compare to the coverage set algorithm because the bilinear programming methods have been shown to be more efficient for all available test problems.

We provide values for the exhaustive variant, *exh*, on small problems and constraint optimization formulation, *COP*, for all problems. We tested our algorithms on six benchmarks: recycling robot, meeting-in-a-grid 3x3 and 8x8; and navigation problems[2]. These are the largest and hardest benchmarks we could find in the literature. We compare our algorithms with: GMAA*-ICE [25], IPG [1], and BLP. The GMAA*-ICE heuristic search consistently outperforms other generic exact solvers such as (G)MAA* [25]. The IPG algorithm is a competitive alternative to the GMAA* approach and performs well on problems with reduced reachability [1]. Results for GMAA*-ICE were provided by Matthijs Spaan and as such were conducted on a different machine. Similarly, results for IPG were collected on different machine. As a result, the timing results for GMAA*-ICE and IPG are not directly comparable to the other methods, but are likely to only differ by a small constant factor from those that would be obtained on our test machine.

The results can be seen in Table 1. In all benchmarks, the *COP* variant of MPS outperforms the other algorithms. The results show that the *COP* variant produces the optimal policies in much less time for all tested benchmarks. For example, in the meeting in a 3x3 grid problem for $T = 5$: the *COP* variant computed the optimal policies approximately 33, 4358 and 4580 times faster than the GMAA*-ICE, BLP and IPG algorithms, respectively. We also note that the *COP* variant is very useful for the medium and large domains. For example, in all large domains, the *exh.* variant ran out of memory while the *COP* variant computed the optimal solutions for horizons up to 100. Yet, the *exh.* variant can compute the optimal solution of small problems faster than the *COP* variant. For instance, in the recycling robot for horizon $T = 1000$, the *exh.* variant computed the optimal solution in about 5 times faster than the *COP* variant of the MPS algorithm due to overhead in the constraint optimization formulation and a lack of structure that can be utilized.

There are many different reasons for these results. The MPS algorithm outperforms GMAA*-ICE and IPG mainly because they perform a policy search in the space of decentralized history-dependent policies. Instead, the MPS algorithm performs its policy search in the space of decentralized Markov policies, which is exponentially smaller than that of the decentralized history-dependent policies. The MPS outperforms the BLP algorithm mainly because of the dimension of its solution representation. More specifically, the number of bilinear terms in the BLP approach grows polynomially in the horizon of the problem, causing it to not perform well for large problems and large horizons with tightly coupled reward values.

We continue the evaluation of the MPS algorithm on randomly generated instances with multiple agents. The random instances were built upon the recycling robot problem described in Sutton and Barto [26]. Given n such models, each of which is associated with a single agent, we choose a number of interaction events. An interaction event is a pair of joint states and actions (s, a) where the reward $r(s, a)$

http://graphmod.ics.uci.edu/group/aolibWCSP/

[2]All problem definitions are available at the following website: http://users.isr.ist.utl.pt/∼ mtjspaan/decpomdp/

is randomly chosen. This structure ties all agents together since the reward model cannot be decomposed among subgroups of agents. In an effort to provide insight on the degree of interaction among all agents, we distinguish between four classes $\{c_0, c_1, c_2, c_3\}$, each of which depends on the number of interaction events e. For each class c_k, we randomly choose e such that $e \in [\frac{k}{4}e_{\max}; \frac{k}{4}(1 + e_{\max})]$, where $e_{\max}$ denotes the number of joint state and action pairs.

As depicted in Figure 2, the constraint formulation allows us to deal with larger numbers of agents. We calculated optimal value functions for 100 instances of each class, and reported the average computational time. The *COP* variant was able to scale up to 6 agents at horizon 10 in about $5,000$ seconds. We could also produce results for up to 10 agents in about $40,000$ seconds using a more powerful machine. It can also be seen that increasing the number of interaction events on each problem does not substantially increase the amount of time required to solve these problems. This shows that even for dense reward matrices, our approach will continue to perform well. Despite this high running time, MPS is the first generic algorithm that scales to teams of more than two agents without taking advantage of the locality of interaction. For example the BLP algorithm as it currently stands can only solve two-agent problems. Moreover, ND-POMDP techniques exploit the small number of local interactions among agents to scale to multiple agents, but in this problem all agents interact.

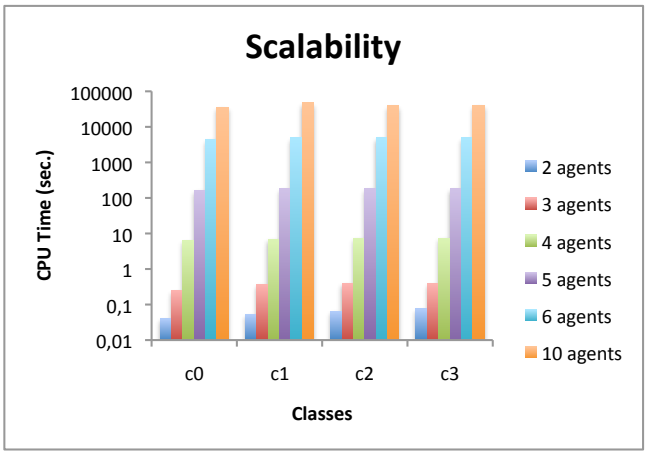


Figure 2: The MPS performance for increasing number of agents at planning horizon $N = 10$ for randomized instances of the recycling robot scenario.

7 Conclusion and Future Work

This paper explores new theory and algorithms for solving independent transition and observation Dec-MDPs. We provide a new proof that optimal policies do not depend on agent histories in this subclass, generalizing previous theoretical results. We also describe a novel algorithm that combines heuristic search and constraint optimization to more efficiently produce optimal solutions for this class of problems. This new algorithm, termed learning Markov policy or MPS, was shown to scale up to large problems and planning horizons, reducing computation time by multiple orders of magnitude over previous approaches. We were also able to demonstrate scalability with respect to the number of agents in domains with up to 10 agents. These results show that our approach could be applied to many large and realistic domains.

In the future, we plan to explore extending the MPS algorithm to other classes of problems and larger teams of agents. For instance, we may be able to produce an optimal solution to more general classes of Dec-MDPs or provide approximate results for Dec-POMDPs by extending the idea of an occupancy distribution to those problems. Furthermore, the scalability of our approach to larger numbers of agents is encouraging and we will pursue methods to increase this even further. In particular, we think our approach could help increase the number of agents that interact in conjunction with other structure in the model such as locality of interaction (as in ND-POMDPs) or sparse joint reward matrices (as in bilinear programming approaches).

8 Acknowledgements

We would like to thank Frans Oliehoek and the anonymous reviewers for their helpful comments about initial versions of the paper as well as Matthijs Spaan and Marek Petrik for providing algorithmic results and code respectively. Research supported in part by AFOSR MURI project #FA9550-091-0538.

References

[1] C. Amato, J. S. Dibangoye, and S. Zilberstein. Incremental policy generation for finite-horizon DEC-POMDPs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 2–9, Thessaloniki, Greece, 2009.

[2] R. Aras and A. Dutech. An investigation into mathematical programming for finite horizon decentralized POMDPs. *Journal of Artificial Intelligence Research*, 37:329–396, 2010.

[3] K. J. Astrom. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

[4] A. G. Barto, S. J. Bradtke, S. P. Singh, T. T. R. Yee, V. Gullapalli, and B. Pinette. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[5] R. Becker, S. Zilberstein, V. R. Lesser, and C. V. Goldman. Solving transition independent decentralized markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[6] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[7] D. S. Bernstein, C. Amato, E. A. Hansen, and S. Zilberstein. Policy iteration for decentralized control of Markov decision processes. *Journal of Artificial Intelligence Research*, 34:89–132, 2009.

[8] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4), 2002.

[9] R. Dechter. Constraint optimization. In *Constraint Processing*, pages 363 – 397. Morgan Kaufmann, San Francisco, 2003.

[10] J. S. Dibangoye, A.-I. Mouaddib, and B. Chaib-draa. Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, Budapest, Hungary, 2009.

[11] R. G. Gallager. *Discrete Stochastic Processes*. Kluwer Academic Publishers, Boston, 1996.

[12] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174, 2004.

[13] R. E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

[14] A. Kumar and S. Zilberstein. Constraint-based dynamic programming for decentralized POMDPs with structured interactions. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 561–568, 2009.

[15] A. Kumar and S. Zilberstein. Point-based backup for decentralized POMDPs: Complexity and new algorithms. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1315–1322, Toronto, Canada, 2010.

[16] A. Kumar, S. Zilberstein, and M. Toussaint. Scalable multiagent planning using probabilistic inference. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2140–2146, 2011.

[17] V. Lesser, C. Ortiz, and M. Tambe, editors. *Distributed Sensor Networks: A Multiagent Perspective*, volume 9. Kluwer Academic Publishers, May 2003.

[18] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 705–711, 2003.

[19] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 133–139, 2005.

[20] F. A. Oliehoek, M. T. J. Spaan, and N. A. Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

[21] M. Petrik and S. Zilberstein. Anytime coordination using separable bilinear programs. In *Proceedings of the AAAI Conference on Artificial intelligence*, pages 750–755, 2007.

[22] M. Petrik and S. Zilberstein. A bilinear programming approach for multiagent planning. *Journal of Artificial Intelligence Research*, 35:235–274, 2009.

[23] M. L. Putterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, New York, NY, 1994.

[24] S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Journal of Autonomous Agents and Multi-Agent Systems*, 17(2):190–250, 2008.

[25] M. T. J. Spaan, F. A. Oliehoek, and C. Amato. Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2027–2032, 2011.

[26] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[27] S. Zilberstein, R. Washington, D. S. Bernstein, and A.-I. Mouaddib. Decision-theoretic control of planetary rovers. In *Advances in Plan-Based Control of Robotic Agents.*, pages 270–289, London, UK, 2002. Springer-Verlag.

Graph-Coupled HMMs for Modeling the Spread of Infection

Wen Dong, Alex "Sandy" Pentland
M. I. T. Media Laboratory
M. I. T.
{wdong, sandy}@media.mit.edu

Katherine A. Heller
Department of Brain and Cognitive Sciences
M. I. T.
{kheller@gmail.com}

Abstract

We develop Graph-Coupled Hidden Markov Models (GCHMMs) for modeling the spread of infectious disease locally within a social network. Unlike most previous research in epidemiology, which typically models the spread of infection at the level of entire populations, we successfully leverage mobile phone data collected from 84 people over an extended period of time to model the spread of infection on an individual level. Our model, the GCHMM, is an extension of widely-used Coupled Hidden Markov Models (CHMMs), which allow dependencies between state transitions across multiple Hidden Markov Models (HMMs), to situations in which those dependencies are captured through the structure of a graph, or to social networks that may change over time. The benefit of making infection predictions on an individual level is enormous, as it allows people to receive more personalized and relevant health advice.

1 INTRODUCTION

Growing amounts of information available from social network data afford us an unprecedented opportunity to answer questions about the individual agents who are represented in these social networks, typically as nodes, via the use of statistical models. The goal of this paper is to provide a framework for modeling the dynamical interactions between individual agents in a social network, and to specifically apply it to modeling the spread of infection.

We present the Graph-Coupled Hidden Markov Model (GCHMM), a discrete-time model for analyzing the interactions between agents in a dynamic social network.

New data and computational power have driven recent developments in models of epidemic dynamics, in which individuals in a population can express different epidemic states, and change state according to certain events. These dynamics models range from assuming homogeneous individuals and relations [Kermack and McKendrick, 1927] to incorporating increasingly more information on individuals and their relationships [Eubank et al., 2004, Salathé et al., 2010, Stehlé et al., 2011, Hufnagel et al., 2004]. For example, Eubank [Eubank et al., 2004] predicted outbreaks ahead of time by placing sensors in key locations, and contained epidemics through a strategy of targeted vaccination using land-use and population-mobility from census data. Salath [Salathé et al., 2010] looked at matching absentee records at a high school with the epidemic size predicted by the susceptible-exposed-infectious-recovered (SEIR) model.

In this paper, we focus on modeling the spread of infection at an individual level, instead of for an entire population. We aim to predict the state of each individual's health, and the paths of illness transmission through individuals in a social network. This information helps us to understand how disease is transmitted locally within a network, and to give individual-level behavioral advice about how to best maintain good health.

The social network data that we employ for this research is mobile phone data collected from 84 people over an extended period of time. The data track each person's proximity to others within the network, and daily symptom reports signal whether individual members of the network might be ill.

Our graph coupled hidden Markov model (GCHMM) incorporates dynamic social network structure into a coupled hidden Markov model [Brand et al., 1997]. A GCHMM allows us to predict at an individual level how the spread of illness occurs and can be avoided. Our results point to a new paradigm of infection control based on sensor networks and individual-level modeling.

Access to dynamic social networks is an essential part of modeling the spread of disease, and is useful for many other real-world social applications as well. The study of dynamic social networks has attracted considerable research in the machine-learning community [Goldenberg et al., 2010]. Since diffusion within dynamic social networks is important in many applications, we believe the GCHMM will also be useful for studying the dynamics of fads, rumors, emotions, opinions, culture, jargon, and so on [Castellano et al., 2009]. Even though we focus on epidemiology in this paper, the same model could be applied to determining, for example, to what extent an individual will change his opinion to match the value of a random neighbor in the formation of community consensus [Holley and Liggett, 1975], to what extent an individual will change one of his traits to match the value of a random neighbor using the Axelrod model of culture formation [Axelrod, 41], how a real-world vocabulary is formed at the society level through imitation and alignment at the individual level [Steels, 1995], and for any of these what further implications might be at both the individual level and the network level.

This paper therefore makes several novel contributions to the field of human behavior modeling and machine learning: 1) We introduce a new class of models, GCHMMs, which combine coupled HMMs with dynamic social networks. We inject dynamic social network structure into the CHMM allowing us to use this class of models for a potentially very wide range of multi-agent network or behavior modeling tasks (e.g. rumor, innovation, or organizational flow in social networks). 2) We specify a particular model in that class, which is a novel model for epidemics. This model allows us to make individual-level epidemics predictions not enabled by previous epidemics methods. 3) We provide methods for performing inference, generally in the case of 1) and specifically in the case of 2), and discuss how they relate to each other and previous work. These methods provide tools for researchers interested in a broad range of multi-agent and social network applications. 4) We validate our epidemics model on an interesting new dataset which tracks the spread of illness on a local, individual level.

The rest of the paper is organized as follows. In section 2 we review the coupled hidden Markov model. In section 3 we introduce the GCHMM for multi-agent modeling in dynamic social networks. In section 4 we show how the GCHMM can be applied to modeling the spread of infection in networks, and in 5 we derive the Gibbs sampler for parameter learning and latent state estimation of the GCHMM for epidemics. In section 6 we apply the GCHMM to our epidemic data from an undergraduate university residence hall, which includes daily symptom reports and hourly proximity tracking.

2 COUPLED HIDDEN MARKOV MODELS

A coupled hidden Markov model (CHMM) describes the dynamics of a discrete-time Markov process that links together a number of distinct standard hidden Markov models (HMMs). In a standard HMM, the value of the latent state at time t (X_t) is dependent on only the value of the latent state at time $t-1$ (X_{t-1}). In contrast, the latent state of HMM i at time t in the CHMM ($X_{i,t}$) is dependent on the latent states of all HMMs in the CHMM at time $t-1$ ($X_{\cdot,t-1}$).

The CHMM generative model is defined as follows:

$$X_{i,t} \sim \text{Categorical}(\phi_{i,X_{\cdot,t-1}}) \quad (1)$$

$$Y_{i,t} \sim F(\theta_{X_{i,t}}) \quad (2)$$

where $X_{i,t}$ is the hidden state of HMM i at time t, $Y_{i,t}$ is the emission of HMM i at time t, $X_{\cdot,t-1}$ is a vector of the state values of all HMMs at time $t-1$, and $\phi_{i,X_{\cdot,t-1}}$ is a vector the dimensionality of which is equal to the number of states in the HMM and the entries of which sum to 1. The entries in $\phi_{i,X_{\cdot,t-1}}$ represent the probability that the state variable in HMM i will transition from its state at time $t-1$ to each possible state at time t, given the states of all other HMMs at time $t-1$. $\theta_{X_{i,t}}$ is the emission parameter for observations that derive from state $X_{i,t}$. The graphical model for the CHMM can be seen in figure 1.

Historically, inference in CHMMs has been achieved via maximum likelihood estimation, usually using an Expectation-Maximization (EM) algorithm. Specialized CHMMs are often used in practice, however, because a CHMM with M_i states for HMM i has $\prod_i M_i$ states in total, and the state transition kernel ϕ_i is a $\prod_i M_i \times \prod_i M_i$ matrix, both of which are a considerable size. Many specializations either omit the inter-chain probability dependence, such as in the factorial hidden Markov model [Ghahramani and Jordan, 1997, Brand et al., 1997], or introduce fixed sparse inter-chain probability dependence, such as in hidden Markov decision trees [Jordan et al., 1996]. The dynamic influence model [Pan et al., 2012] allows one chain to probabilistically depend on all other chains through only a few sufficient statistics without increasing modeling complexity.

However, in the following sections we introduce the GCHMM, which differs from the models described above in considering HMMs that are coupled based

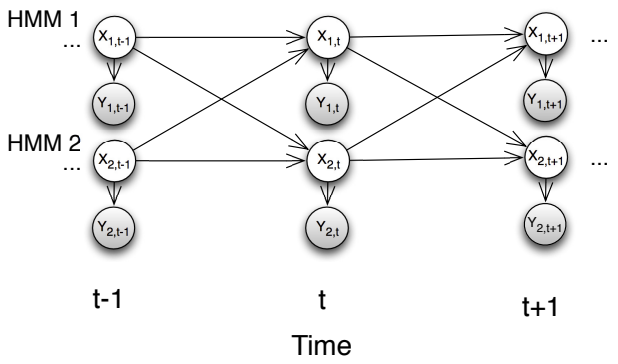


Figure 1: CHMM graphical model

on social network structure, and for which efficient inference can be performed based on the inter-chain "relations" of sufficient statistics.

The CHMM generative model can easily be formulated in a Bayesian manner, which helps to avoid overfitting problems in real-world applications [Beal, 2003]:

$$\theta_{X_i} \sim \text{Conj}(\gamma) \quad (3)$$
$$\phi_{i,X_\cdot} \sim \text{Dirichlet}(\alpha) \quad (4)$$

where Conj refers to the distribution conjugate to F, above, and γ and α are shared hyperparameters. Here, θ_{X_i} and $\phi_{i,X_\cdot}$ are drawn once for all times t. X_i is the set of all state values that HMM i can take on, while $X_\cdot$ is the set of all state values that all HMMs can take on in the CHMM.

This is the Bayesian formulation of the CHMM that we extrapolate in the next section in order to deal with situations wherein the relationship between HMMs is mediated by a dynamic social network.

3 GRAPH-COUPLED HIDDEN MARKOV MODELS

Here, we introduce the graph-coupled hidden Markov model (GCHMM) to model how infection and influence spread via the interactions of agents in a dynamic social network.

Let $G_t = (N, E_t)$ be a dynamic network, where each $n \in N$ is a node in G_t representing an individual agent, and $E_t = \{(n_i, n_j)\}$ is a set of edges in G_t representing interactions between agents n_i and n_j at time t. Graph G_t changes over time, where there is a discrete number of observations of the structure of the graph at times $t \in \{1 \dots T\}$. Changes in G_t over time represent changes in interactions between agents in the network over time.

We can use the CHMM in conjunction with dynamic network G_t if we let $X_{n,t}$ be the state of agent (node) n at time t, and $Y_{n,t}$ be noisy observations about agent n at time t. G_t restricts the connections between latent state variables in the CHMM, allowing us to model multi-agent phenomena of interest while potentially allowing for efficient inference.

The generative model of the GCHMM in its most general form is as follows:

$$X_{n,t} \sim \text{Categorical}(\phi_{n,X_{e:\{n,\cdot\}\in G_t,t-1}}) \quad (5)$$
$$Y_{n,t} \sim F(\theta_{X_{n,t}}) \quad (6)$$
$$\theta_{X_n} \sim \text{Conj}(\gamma) \quad (7)$$
$$\phi_{n,X_{e:\{n,\cdot\}\in G_t}} \sim H(X_{e:\{n,\cdot\}\in G_t}, \mu) \quad (8)$$

Unlike the CHMM, whose transition matrix ϕ_n for HMM n is dependent on all other HMMs, ϕ_n in the GCHMM is dependent on only the HMMs that have edges in the graph connected to node n. Thus, $\phi_{i,X_\cdot}$ becomes $\phi_{n,X_{e:\{n,\cdot\}\in G_t}}$. We assume that n itself is also included in this set of HMMs on which n is dependent. There is also a difference in the prior distribution of ϕ_n. In the CHMM, the prior is the same for all rows of the transition matrix, whereas in the GCHMM the prior on ϕ_n, H, depends on the values of the states of the HMMs on which n is dependent at the previous time step.

The graphical model for the GCHMM is given in figure 2. Here, we show a GCHMM with 3 HMMs. Network structure G_t is depicted in the bubbles above each time step, also showing the dependency structure corresponding to G_t in the GCHMM graphical model. The structure for G_{t-1} is not displayed, since it would be off the left side of the figure.

This is a discrete-time multi-agent model, and thus it approximates its continuous-time multi-agent counterpart: a Markov jump process, also called a compound Poisson process. This approximation works well only when the time step size in the discrete-time model is smaller than the average rate of interaction events. In our setting, as in many settings in which these multi agent models may be used, this is not an issue.

In the following, we describe an application of the GCHMM to fit susceptible-infectious-susceptible epidemic dynamics. Here, we assume specific forms for distributions F and H. Much like in the CHMM, efficient inference may not always be possible in the GCHMM, but we show that efficient inference can easily be done in our specific application. We expect that efficient inference in the GCHMM will be possible for many applications, since the incorporation of social networks in the GCHMM leads to sparsity in the con-

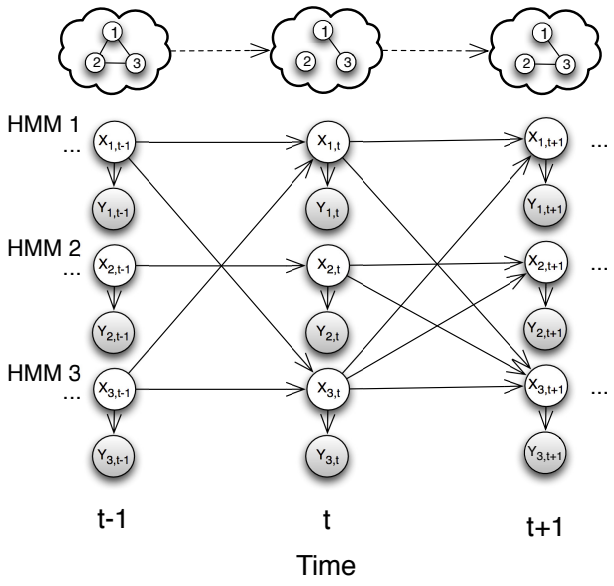


Figure 2: GCHMM graphical model

nections between latent variables, since these networks are typically sparse, and the incorporation of structure resulting from network-specific prior knowledge in H can also often be leveraged. For example, as in the epidemics model, H can become a simple function of a few parameters and sufficient statistics of connected states.

4 GCHMMS FOR MODELING INFECTION DYNAMICS

In this section, we show that the GCHMM can be used as a discrete-time multi-agent epidemic model, where the spread of infection is modeled on an individual level as opposed to the standard population-level models commonly used in epidemiology. In section 6, we show that the GCHMM can be applied to real-world data based on individual proximity and symptom reporting.

In particular, the GCHMM that we apply to epidemics can be seen as an individual-level version of the classic susceptible-infectious-susceptible (SIS) epidemiology model. The SIS model describes infection dynamics in which the infection does not confer long-lasting immunity, and so an individual becomes susceptible again once recovered (e.g., the common cold).

Using the GCHMM to model SIS dynamics identifies paths of infection based on individual-level interactions that are captured in our data.The classical differential equation and stochastic models of SIS dynamics work at the population level, and their variables are density and size of susceptible and infectious populations, respectively. The differential equation model $\dot{S} = -\beta \cdot SI + \gamma \cdot I$ and $\dot{I} = \beta \cdot SI - \gamma \cdot I$ for SIS specifies that the rate of change of infectious-population density is bilinear for both infectious-population density and susceptible-population density. In this system, any two individuals from the infectious population are treated as the same, as are any two individuals from the susceptible population, and therefore any two individuals from different populations have an equal chance of causing infection. The stochastic model $S + I \rightarrow 2I, rate = \beta' \cdot |S| \cdot |I|$ and $I \rightarrow S, rate = \gamma' \cdot |I|$ specifies that infection happens at a bilinear rate for both the infectious-population density and the susceptible-population density. This model enables us to reason about the randomness in the SIS system when the population size is small and randomness cannot be ignored.

The above models focus on the statistics of the spread of infection over a homogeneous population. However, we are instead interested in predicting the spread of infection on an individual level, given relevant information about each specific individual. Our goal is to explain symptom observations in a community with susceptible-infectious-susceptible dynamics at any given point in time. How likely is a person to be infectious at time t, given that his friends are reporting symptoms, reporting no symptoms, or not answering surveys, and given the infectious person's own survey responses and his recent proximity to his friends? Which nodes and links are critical in spreading infection in the community? How can we control infection in this community?

We use the GCHMM to address these questions, following the generative model given in section 3 and the details (state space, H, F, and so on) specified here.

$G_t = (N, E_t)$ is a dynamic network, where each node $n \in N$ represents a person in the network, and $E_t = \{(n_i, n_j)\}$ is a set of edges in G_t representing the fact that person n_i and person n_j have interacted or come into contact at time t. There are two possible latent states for each person n at each time t, $X_{n,t} \in \{0, 1\}$, where 0 represents the susceptible state and 1 the infectious state. There are six possible symptoms that can be displayed by a person at any given time, $Y_{n,t}$, which are runny nose, coughing, fever, feeling stressed, sadness, and diarrhea, and each symptom $Y_{n,t,i} \in \{0, 1\}$ represents the presence or absence of symptom i.

Our generative model is therefore as follows:

$$X_{n,t} \sim \text{Bernoulli}(\phi_{n, X_{e:\{n,\cdot\} \in G_t, t-1}}) \quad (9)$$

$$Y_{n,t,i} \sim \text{Bernoulli}_i(\theta_{X_{n,t}}) \quad (10)$$

$$\theta_{X_n} \sim \text{Beta}(h) \quad (11)$$

This is identical to the generative model from section 3 (the Bernoulli is the same as Categorical for 2 states), with F specified as a multivariate Bernoulli distribution. Here, h are given hyperparameters. The generative process for $\phi_{n,X_{e:\{n,\cdot\}\in G_t}}$ is a little more subtle, as it is defined using the interaction structure in the network, detailed below..

In keeping with the SIS model, we assume that there are certain transmission rates for the infection of one person by another, and likewise a recovery rate for an infected individual:

$$\mu = \begin{cases} \alpha \sim \text{Beta}(a, b) \\ \beta \sim \text{Beta}(a', b') \\ \gamma \sim \text{Beta}(a'', b'') \end{cases} \quad (12)$$

γ is the probability that a previously-infectious individual recovers and so again becomes susceptible ($p(X_{n,t+1} = 0|X_{n,t} = 1)$), β represents the probability that an infectious person infects a previously-susceptible person ($p(X_{n_i,t+1} = 1|X_{n_i,t} = 0, X_{n_j,t} = 1, \{n_i, n_j\} \in E_{t+1})$), and α represents the probability that an infectious person from outside the network infects a previously-susceptible person within the network. Each of these infection probabilities is assumed to be independent. It is also assumed that a person cannot be infected by more than one infectious persons at the same time. Here, $\{a, b, a', b', a'', b''\}$ are given hyperparameters.

Therefore, we can now compute transition matrix $\phi_{n,X_{e:\{n,\cdot\}\in G_t}}$ (and thus specify H from section 3) in terms of α, β, and γ:

$$\begin{aligned} & p(X_{n,t+1} = 0|X_{n,t} = 1) = \gamma & (13) \\ & P(X_{n,t+1} = 1|X_{n,t} = 0, X_{e:\{n,\cdot\}\in G_t}) & (14) \\ = \; & 1 - P(X_{n,t+1} = 0|X_{n,t} = 0, X_{e:\{n,\cdot\}\in G_t}) \\ = \; & 1 - (1-\alpha)(1-\beta)^{\sum_{e:\{n,\cdot\}\in G_t} X_{n',t}} \end{aligned}$$

Thus our H from equation 8 is now a simple function of parameters μ and sufficient statistics of connected states, $g(X)$, given by the summation term in the above equation. Intuitively, the probability of a susceptible person becoming infected is 1 minus the probability that no one, including someone from outside the network, or any of the people within the network with whom the susceptible person is known to have interacted, infected that individual. When the probability of infection is very small, it is approximately the sum of the probabilities from different sources ($P(X_{n,t+1} = 1|X_{n,t} = 0, X_{e:\{n,\cdot\}\in G_t}) \approx \alpha + \beta \cdot \sum_1^{|X_{e:\{n,\cdot\}\in G_t|}} X_{e:\{n,\cdot\}\in G_t,t}$), since the probability that more than one source contributed to infection is also small.

If this assumption is correct, then the probability of seeing an entire state sequence/matrix X is therefore as follows:

$$\begin{aligned} & P(X, \alpha, \beta, \gamma) \\ = \; & P(\alpha)P(\beta)P(\gamma)\prod_n P(X_{n,1}) \\ & \prod_{t,n} P(X_{n,t+1}|\{X_{n'\in N,t}\}, \alpha, \beta, \gamma) & (15) \\ = \; & P(\alpha)P(\beta)P(\gamma)\prod_n P(X_{n,1}) \\ & \prod_{t,n} \gamma^{1_{X_{n,t}=1}\cdot 1_{X_{n,t+1}=0}} \cdot (1-\gamma)^{1_{X_{n,t}=1}\cdot 1_{X_{n,t+1}=1}} \cdot \\ & (\alpha + \beta \cdot \sum_1^{|X_{e:\{n,\cdot\}\in G_t}|} X_{e:\{n,\cdot\}\in G_t,t})^{1_{X_{n,t}=0}\cdot 1_{X_{n,t+1}=1}} \cdot \\ & (1 - \alpha - \beta \cdot \sum_1^{|X_{e:\{n,\cdot\}\in G_t}|} X_{e:\{n,\cdot\}\in G_t,t})^{1_{X_{n,t}=0}\cdot 1_{X_{n,t+1}=0}} \end{aligned}$$

It is assumed that all people start off in the susceptible state, and that $1_\bullet$ in the above equation is the indicator function.

In general, we can often identify M types of events in agent-based models, sometimes by taking small enough time intervals in time-discretization to make first-order approximation sufficient, and count the different ways that events of type $m \in \{1, \ldots, M\}$ could change an agent's state from x' at time t to state $x \neq x'$ at time $t+1$, $g_m(X_{\bullet,t+1} = x, \{X_{n,t} : n, t\})$. The state transition matrix can then be defined as

$$\begin{aligned} & P(X_{n,t+1} = x|X_{n,t} \neq x, X_{e:\{n,\bullet\}\in G_t}) \approx & (16) \\ & \sum_{m=1}^{M} \mu_m \cdot g_m(X_{n,t+1} = x, X_{n,t}, X_{e:\{n,\bullet\}\in G_t}), \\ & \text{where } \mu_m \text{ is the success rate of event type } m. \end{aligned}$$

This formulation of the GCHMM enables us to fit a wide range of agent-based models to "big data" sets of networked people, by specifying the right ways of counting events [Castellano et al., 2009]. For example, in the Sznajd model of opinion dynamics [Sznajd-Weron, 2005], any pair of agents with the same opinion will have a small chance μ of convincing one of their neighbors to change to their opinion. The number of ways that an individual can be convinced by a pair of neighbors is then $g(X) = \binom{\sum_{(n',n)\in G_t} 1_{X_{n'}\neq X_n}}{2}$, and the chance that an agent is convinced is $\mu \cdot g(X)$. Sznajd's model captures the intuition that we might not pay

attention to a single person looking up, but we would look up if a group of people look up.

5 INFERENCE

In this section, we present an efficient inference algorithm for our GCHMM in order to describe and predict the spread of infection. We start by describing inference in the most general, worst case for GCHMMs, and progress from there to much more efficient inference for the epidemics model. The worst case inference algorithm for GCHMMs will be the same as for CHMMs (the case where the graph is complete). A Gibbs sampling algorithm for a particular CHMM with two chains is given in [Rezek et al., 2000], which we extend here to an unconstrained number of chains. The two sampling steps relevant to the extension to multiple chains become:

$$\begin{aligned} X_{n,t+1} &\sim \text{Categorical}([\frac{p(X_{n,t+1}=j, Y|\gamma,\phi)}{\sum_k p(X_{n,t+1}=k, Y|\gamma,\phi)}]) \\ \phi_{n,i} &\sim \text{Dirichlet}([\alpha_j + \\ &\qquad \sum_{\tau=1}^{T} 1(X_{n,\tau+1}=j \wedge X_{\bullet,\tau}=i)]) \end{aligned}$$

Here j and k are states of n, and i is the transition matrix row number corresponding to a combination of states for all nodes. We can see that this sampling procedure is not very efficient, particularly for a reasonably large number of chains.

Fortunately, in practice most social networks are sparse - far from complete. The number of parameters needed to be inferred will decrease dramatically for sparse networks (from $O(N^N)$ to $O(N^{n_{max}})$ where $n_{\max}$ is the maximum number of connections for node n). The parameters of the transition matrix can now be sampled conditioned on the network structure:

$$\begin{aligned} \phi_{n,i} \sim \text{Dirichlet}([\alpha_j + \\ \sum_{\tau=1}^{T} 1(X_{n,\tau+1}=j \wedge X_{e:\{n,\cdot\}\in G_{\tau+1},\tau}=i)]) \end{aligned} \tag{17}$$

Here i now corresponds to a combination of the states of nodes connected to n only. While significantly better than the full CHMM, this may still be intractable for large N or T. However, the interaction structure that we are interested in is also not typically governed by unrestricted transition matrices, ϕ. There is **structure** to interactions or behavior that we can leverage to drastically increase efficiency. One common example is that all agents may be subject to the same ϕ. Another is that interactions themselves have a structure which can be captured by an $H(\mu, g(X))$, where H is now a function parameterized by a reasonably small set of parameters μ and sufficient statistics of the current state space $g(X)$. This allows us to sample $p(\mu|g(X))$, and then compute ϕ, unlike in the algorithms above. This form of H applies to many multi-agent network models including the emergence of collaboration, the spread of rumors and the formation of culture. It similarly applies to our epidemics model, as described in section 4, where where $g(X)$ can be efficiently computed at each node, at each time step. We now describe the resulting inference method for the epidemics model, and then briefly discuss applying the same inference scheme to some more general, $g(X)$.

The epidemics inference algorithm learns the parameters of infection dynamics, including the rate of infection and rate of recovery. It then estimates the likelihood that an individual becomes infectious from the contact with other students based on the reported symptoms of others, and even when the individual's own symptom report is not available. Finally, the algorithm enables us to make useful predictions about the spread of infections within the community in general.

We employ a Gibbs sampler to iteratively sample the infectious/susceptible latent state sequences, to sample infection and recovery events conditioned on these state sequences, and to sample model parameters.

The Gibbs sampler for the GCHMM for epidemics is given in detail below.

The Gibbs sampler takes the following as its input:

$G = (N, E)$: a dynamic network where nodes $n \in N$ represent a person in the network, and $E_t = \{(n_i, n_j)\}$ is a set of edges in G_t representing that person n_i and person n_j have interacted or come into contact at time t.

Y: an observation matrix of symptoms indexed by time and node.

The Gibbs sampler gives the following output:

$\{X, \alpha, \beta, \gamma, \theta\}_s$: samples s. This includes several parameters: α, the base rate of infection; β, the rate of infection by each infectious neighbor in G_t; and γ, the rate of recovery. It includes the emission matrix θ, which expresses the probability of seeing each of the six symptoms given the infectious/susceptible latent state. It also includes the state matrix X of the epidemics GCHMM, which shows sequences of states 0 (susceptible) and 1 (infectious) for each individual over time.

We randomly initialize the model parameters and set the state matrix so that every individual in the network is in the susceptible state. The Gibbs sampler

then iterates through sampling the latent states:

$$X_{n,t+1}|\{X,Y\}\backslash X_{n,t+1};\alpha,\beta,\gamma \sim \text{Bernoulli}\left(\frac{P(X_{n,t+1}=1)}{\sum_{x=0,1}P(X_{n,t+1}=x)}\right) \quad (18)$$
$$P(X_{n,t+1}=1) = P(X|\alpha,\beta,\gamma)P(Y|X)$$

where $P(X|\alpha,\beta,\gamma)$, is the same as in equation 15 minus the priors on α, β, and γ. $P(Y|X)$ can be computed in a straightforward manner from the product of entries in the emissions matrix, θ, where $P(Y|X) = \prod_{n,t,i} P(Y_{i,n,t}|X_{n,t})$.

After sampling the latent states X, we sample infection events. Due to the interaction structure, and to ease sampling, we introduce the auxiliary variable $R_{n,t}$ to track the source of an infection (inside or outside the network) if an infection event occurs for person n at time $t+1$ (i.e., $X_{n,t}=0$ and $X_{n,t+1}=1$):

$$R_{n,t} \sim \text{Categorical}\left(\frac{\alpha,\beta,\ldots,\beta}{\alpha+\beta\sum_{n'}1_{(n',n)\in E_t\cap X_{n',t}=1}}\right) \quad (19)$$

Here $R_{n,t}$ takes the value 1 if the infection event originated outside the network, and $R_{n,t}>1$ if transmission came from someone within the network. Given the state sequences X and infection events R, we can sample the remaining parameters from their posteriors:

$$\alpha \sim \text{Beta}(a+\sum_{n,t}1_{\{R_{n,t}=1\}}, b+\sum_{n,t:X_{n,t}=0}1-\sum_{n,t}1_{\{R_{n,t}=1\}}), \quad (20)$$

$$\beta \sim \text{Beta}(a'+\sum_{n,t}1_{\{R_{n,t}>1\}}, b'+\sum_{n,t:X_{n,t}=0;n'}1_{(n',n)\in E_t\cap X_{n',t}=1}-\sum_{n,t}1_{\{R_{n,t}>1\}}), \quad (21)$$

$$\gamma \sim \text{Beta}(a''+\sum_{n,t:X_{n,t}=1}1_{\{X_{n,t+1}=0\}}, b''+\sum_{n,t:X_{n,t}=1}1-\sum_{n,t:X_{n,t}=1}1_{\{X_{n,t+1}=0\}}). \quad (22)$$

In the more general M state case, we can sample $X_{n,t+1}$ from its posterior categorical distribution similar to equation 18, sample events $R_{n,t}\in\{1,\ldots,M\}$ (reflecting which type of event caused the state change of $X_{n,t}$, c.f., equation 16) similar to equation 19, and sample the success rates of different types of events similar to equation 20.

$$X_{n,t+1}|\{X,Y\}\backslash X_{n,t+1};\mu \sim \text{Categorial}\left(\frac{P(X_{n,t+1})}{\sum_x P(X_{n,t+1}=x)}\right), \quad (23)$$

$$R_{n,t} \sim \text{Categorial}\left(\frac{\mu_1 g_1,\ldots,\mu_M g_M}{\sum_m \mu_m g_m}\right), \quad (24)$$

$$\mu_m \sim \text{Beta}(a_m+\sum_{n,t}1_{R_{n,t}=m}, b_m+\sum_{n,t}g_m-1_{R_{n,t}=m}) \quad (25)$$

6 EXPERIMENTAL RESULTS

In this section we describe the performance of the epidemics GCHMM in predicting missing data in multiple synthetic time series, comparing to a Support Vector Machine (SVM) and standard SIS model. We also fit our epidemics model to the hourly proximity records and self-reported symptoms in a real world Social Evolution data set.

6.1 CONTAGION IN THE SOCIAL EVOLUTION EXPERIMENT

To demonstrate the potential of GCHMMs and our epidemics model, we use it on the data collected in the Social Evolution experiment [Dong et al., 2011], part of which tracked common cold symptoms in a student residence hall from January 2009 to April 2009. This study monitored more than 80% of the residents of the hall through their mobile phones from October 2008 to May 2009, taking periodic surveys which included health-related issues and interactions, and tracked their locations. In particular students answered flu surveys, about the following symptoms: (1) runny nose, nasal congestion, and sneezing; (2) nausea, vomiting, and diarrhea; (3) frequent stress; (4) sadness and depression; and (5) fever. Altogether, 65 residents out of 84 answered the flu surveys.

Because of the symptom reports and proximity information, the Social Evolution data is a good test bed for fitting infection models to real-world data, and for inferring how friends infect one another: For almost all symptoms, a student with a symptom had 3-10 times higher odds of having friends with the same symptom, confirming that symptoms are probabilistically dependent on their friendship network. The durations of symptoms averaged two days, and fit an exponential distribution well, agreeing with the discussed epidemic models. The base probability of a subject reporting a symptom is approximately 0.01, and each individual had a 0.006˜0.035 increased chance of reporting

a symptom for each additional friend with the same symptom, in line with the assumption that the rate of contagion is proportional to the likelihood of contact with an infected individual.

6.2 CALIBRATING PERFORMANCE

We took several steps to calibrate the performances of our epidemics GCHMM and a support vector classifier on synthetic data. First, we synthesized 200 time series – each 128 days long – from the Bluetooth proximity pattern in the Social Evolution data and different parameterizations. Then, we randomly removed the infectious/susceptible data from 10% of the population, added noise to each time series, and then inferred the held-out data with each method, for each parameterization.

The different parameterizations were (1) $\alpha = 0.01$, $\beta = 0.02$, and $\gamma = 0.3$, with observation error 0.01; (2) $\alpha = 0.01$, $\beta = 0.02$, and $\gamma = 0.3$, with observation error 0.001; and (3) $\alpha = 0.005$, $\beta = 0.045$, and $\gamma = 0.3$, with observation error 0.01. Comparing performances between (1) and (2) enables us to see the effect of observation error on algorithm performance. Comparing performances between (1) and (3) enables us to see the effect of the network on algorithm performance. Comparing performances between methods enables us to see the difference between our model-based learning and the SVM or SIS model.

We ran Gibbs samplers for 10,000 iterations, including 1000 for burn-in. We trained the SVM on a 1000-day time series synthesized using the correct parameterization, and used the number of infectious contacts yesterday, today, and tomorrow as features. We assigned different weights to the "infected" class and the "susceptible" class to balance the true and false prediction rates.

All methods can easily identify 20% of infectious cases in the missing data with little error; however, the our model-based method consistently performs better than SIS and SVM. Less noise in symptoms observations and in the contact networks of individuals significantly improves the performance of inferring missing data, as shown in Figure 3. An ROC curve shows how different algorithms compare in terms of inferring infectious cases in the held out 10% of the data.

The SVM performs poorly – especially at identifying isolated infectious cases – because it assumes that cases are i.i.d and because properly incorporating the temporal structure of epidemic dynamics into the features is not easy. The SVM also assumes that we have enough training data. This assumption often cannot be satisfied for the kinds of problems we are interested in. Lastly, we also compare to the traditional

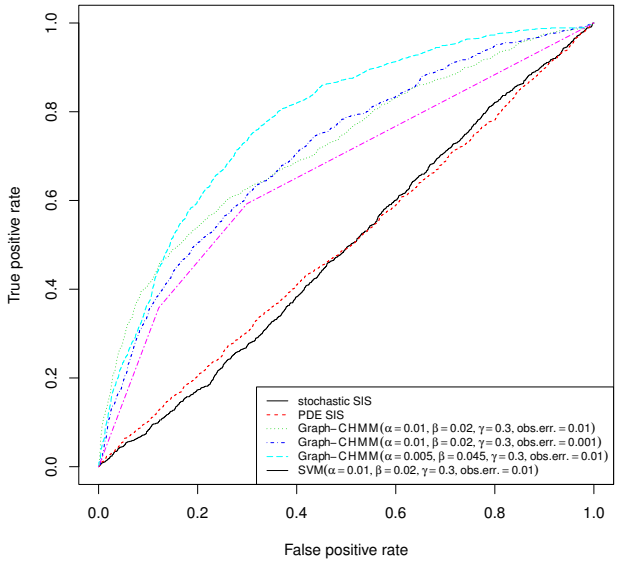


Figure 3: Less observation error (obs.err.=0.001) and better knowledge about the network ($\beta = 0.045$) result in a better trade-off between true positive rate (TPR) and false positive rate (FPR). Lack of knowledge about the network and assuming a compete graph structure result in poor trade-off between TPR and FPR. The support vector classifier has a worse trade-off between TPR and FPR than the epidemics GCHMM does.

SIS methods (both pde and stochastic). They do not predict well because they treat everyone in the population the same, regardless of individual interaction network information.

Observation noise makes inferring the individual states difficult, since it increases uncertainty in the parameters of the system. Knowledge of the dynamic contact network also affects the quality of parameter estimation and state inference. The more we know about who contacted whom, the more targeted we can be in locating the infectious cases.

6.3 INFERRING COLD FROM SYMPTOMS AND PROXIMITY

In this section, we report the results of our epidemics model on the Social Evoluation data. In order to infer infections we extracted an hour-by-hour proximity snapshot over the 107 days that we monitored symptoms, and interpolated hourly symptom reports from the daily symptom reports. We assumed that the symptoms are probabilistically independent given the common cold (infectious) state. We ran the Gibbs sampler for 10,000 iterations, with 1000 burn-in iterations.

We do not have the clinical certainty of common cold

diagnoses . However, the statistics that we discuss below give solid evidence that the epidemics model captures the structure of a infection process accompanying the symptom report.

Figure 4 shows the (marginal) likelihood of the daily common-cold states of individuals. Rows in this heat map are indexed by subjects, arranged so that friends go together, and then placed next to a dendrogram that organizes friends hierarchically into groups. Different colors on the leaves of the dendrogram represent different living sectors in the student residence hall. Columns in this heat map are indexed by date in 2009. Brightness of a heat-map entry indicates the likelihood of infectiousness - the brighter a cell, the more likely it is that the corresponding subject is infectious on the corresponding day. Sizes of black dots represent the number of reported symptoms. When a black dot doesn't exist on a table entry, the corresponding person didn't answer the survey that day.

This heat map shows clusters of common-cold infections. When larger clusters of interpersonal proximities occurred , symptom clusters lasted longer and involved more people. The heat map also tells us that subjects often submitted flu-symptom surveys daily when they were healthy, but would forget to submit surveys when in the infectious state. The Gibbs sampler nonetheless uses the data to determines that the individual was infectious for these days. Similarly, a subject sometimes reported isolated symptoms. The Gibbs sampler is able to conclude the he was most likely healthy, because the duration of the symptom reports didn't agree with the typical duration of a common cold, and because his symptom report was isolated in his contact network.

Inferred Infectious states normally last four days to two weeks. A student typically caught a cold 2-3 times during this time span. The bi-weekly searches of the keyword "flu" from January 2009 to April 2009 in Boston – as reported by Google Trends – correlated with these findings.

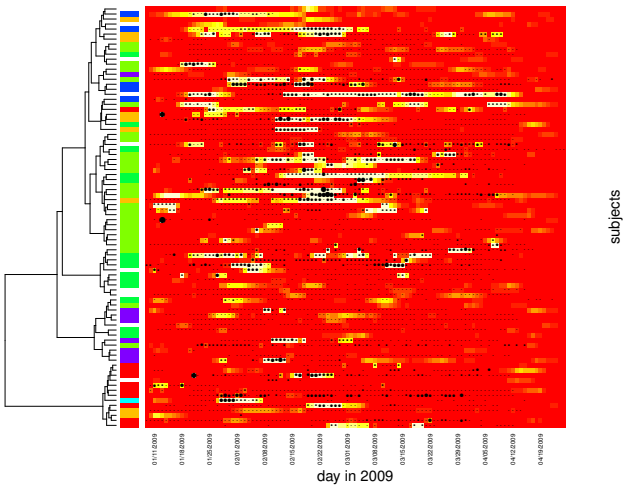


Figure 4: Our epidemics GCHMM can infer common cold state, and captures infection data from symptom self-reports and the proximity network. The size of black dots represents the number of symptoms reported, ranging from zero symptoms to all, and no black dot means no self-report.

7 CONCLUSIONS

We have presented the GCHMM for modeling discrete-time multi-agent dynamics when the agents are connected through a social network. We showed that the GCHMM can be used as an individual-level SIS model to successfully predict the spread of infection throughout a social network. In the future, it would be interesting to use the GCHMM to learn graph dynamics, or to predict missing links. It would also be interesting to try to use the GCHMM in applications with a more complex transition matrix structure.

The study of infection in small populations has important implications both for refining epidemic models and for advising individuals about their health. The spread of infection in this context is poorly understood because of the difficulty in closely tracking infection throughout a complete community. This paper showcases the spread of an infection centered on a student dormitory, with data based on daily symptom surveys over a period of four months and on proximity tracking through mobile phones. It also demonstrates that fitting a discrete-time multi-agent model of infection to real-world symptom self-reports and proximity observations gives us useful insights into infection paths and control.

Acknowledgements

Research was sponsored by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, and by AFOSR under Award Number FA9550-10-1-0122. Views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation. Katherine Heller was supported on an NSF postdoctoral fellowship.

References

Robert Axelrod. The dissemination of culture — a model with local convergence and global polarization. *Journal of Conflict Resolution*, 2(203-226), 41.

Matthew J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden markov models for complex action recognition. In *CVPR*, pages 994–999. IEEE Computer Society, 1997.

Claudio Castellano, Santo Fortunato, and Vittorio Loreto. Statistical physics of social dynamics. *Reviews of Modern Physics*, 81:591–646, 2009.

Wen Dong, Bruno Lepri, and Alex Pentland. Modeling the co-evolution of behaviors and social relationships using mobile phone data. In Qionghai Dai, Ramesh Jain, Xiangyang Ji, and Matthias Kranz, editors, *MUM*, pages 134–143. ACM, 2011. ISBN 978-1-4503-1096-3.

S. Eubank, H. Guclu, V. Kumar, M. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429: 180–4, 2004.

Zoubin Ghahramani and Michael I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273, 1997.

Anna Goldenberg, Alice X. Zheng, Stephen E. Fienberg, and Edoardo M. Airoldi. A survey of statistical network models. *Found. Trends Mach. Learn.*, 2:129–233, February 2010. ISSN 1935-8237.

Richard A. Holley and Thomas M. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter model. *Annals of Probability*, 3(4):643–663, 1975.

L. Hufnagel, D. Brockmann, and T. Geisel. Forecast and control of epidemics in a globalized world. In *Proc Natl Acad Sci USA*, volume 101, pages 15124–9, 2004.

Michael I. Jordan, Zoubin Ghahramani, and Lawrence K. Saul. Hidden markov decision trees. In Michael Mozer, Michael I. Jordan, and Thomas Petsche, editors, *NIPS*, pages 501–507. MIT Press, 1996.

W. Kermack and A McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, August 1927.

Wei Pan, Wen Dong, Manuel Cebrian, Taemie Kim, James H. Fowler, and Alex Pentland. Modeling dynamical influence in human interaction: Using data to make better inferences about influence within social systems. *Signal Processing Magazine, IEEE*, 29(2):77 –86, march 2012. ISSN 1053-5888. doi: 10.1109/MSP.2011.942737.

I. Rezek, P. Sykacek, and S. J. Roberts. Learning interaction dynamics with coupled hidden Markov models. *Science, Measurement and Technology, IEE Proceedings-*, 147(6):345–350, 2000. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=899989`.

Marcel Salathé, Maria Kazandjiev, Jung Woo Lee, Philip Levis, Marcus W. Feldman, and James H. Jones. A high-resolution human contact network for infectious disease transmission. In *Proceedings of the National Academy of Science of the United States of America*, volume 107, pages 22020–22025, 2010.

Luc Steels. A self-organizing spatial vocabulary. *Artificial Life*, 2(3):319–332, 1995. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/steels96aSelf.html`.

Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Vittoria Colizza, Lorenzo Isella, Corinne Régis, Jean-François Pinton, Nagham Khanafer, Wouter Van den Broeck, and Philippe Vanhems. Simulation of an seir infectious disease model on the dynamic contact network of conference attendees. *BMC Medicine*, 9(1):87, 2011.

K. Sznajd-Weron. Sznajd model and its applications. *Acta Physica Polonica B*, 36(8):2537–2547, 2005.

DBN-Based Combinatorial Resampling for Articulated Object Tracking

Séverine Dubuisson **Christophe Gonzales** **Xuan Son NGuyen**
Laboratoire d'Informatique de Paris 6 — Université Pierre et Marie Curie
4, place Jussieu, F-75005 Paris
email: firstname.lastname@upmc.fr

Abstract

Particle Filter is an effective solution to track objects in video sequences in complex situations. Its key idea is to estimate the density over the possible states of the object using a weighted sample whose elements are called *particles*. One of its crucial step is a resampling step in which particles are resampled to avoid some degeneracy problem. In this paper, we introduce a new resampling method called *Combinatorial Resampling* that exploits some features of articulated objects to resample over an implicitly created sample of an exponential size better representing the density to estimate. We prove that it is sound and, through experimentations both on challenging synthetic and real video sequences, we show that it outperforms all classical resampling methods both in terms of the quality of its results and in terms of response times.

1 INTRODUCTION

Tracking articulated structures with accuracy and within a reasonable time is challenging due to the high complexity of the problem to solve. Actually, the state space of such a problem is inevitably high-dimensional and the estimation of the state of an object thus requires that of many parameters. When the dynamics of the objects is linear or linearizable and when the uncertainties about their position are Gaussian or mixtures of Gaussians, tracking can be performed analytically by Kalman-like Filters [Chen, 2003]. Unfortunately, in practice, such properties seldom hold and people often resort to sampling to approximate solutions of the tracking problem. The Particle Filter (PF) methodology [Gordon et al., 1993] is popular among these approaches and, in this paper, we focus on PF.

PF consists of estimating the density over the states of the tracked object using weighted samples whose elements are possible realizations of the object state and are called *particles*. PF and its variants, e.g., Partition Sampling (PS) [MacCormick and Blake, 1999], all use a resampling step to avoid a problem of degeneracy of the particles, i.e., the case when all but one of the particle's weights are close to zero [Douc et al., 2005]. Without this step, this problem would necessarily occur [Doucet et al., 2001].

A few resampling algorithms are classically used, e.g., Multinomial Resampling [Gordon et al., 1993], Residual Resampling [Liu and Chen, 1998], Stratified and Systematic Resampling [Kitagawa, 1996]. However, these methods have not been designed specifically to deal with articulated objects and, as such, they do not exploit their features. In this paper, we introduce *Combinatorial Resampling*, an algorithm that exploits them to produce better samples by resampling over an implicitly created sample of an exponential size. More precisely, in articulated object tracking, a particle may be thought of as a tuple of the realizations of each "part" of the object and it is often the case that swapping the realizations of a given part among several particles has no impact on the estimated distribution. For instance, in a human body tracking, it may be the case that swapping the positions of the left arm estimated by two particles does not alter the estimated distribution. Given a particle set, Combinatorial Resampling produces implicitly a new set of particles resulting from all such swappings and resamples from it. As such, this new set is of exponential size and acts as a much better description of the state space.

The paper is organized as follows. The next section recalls the basics of articulated object tracking and, in particular, Partitioned Sampling. It also recalls the fundamentals of dynamic Bayesian networks, as our resampling method relies on them. Section 3 presents a short overview of the aforementioned classical resampling approaches and the next one details our new resampling approach and its correctness. Section 5 shows some experimental results both on challenging synthetic and real video sequences. Those highlight the efficiency of our method both in terms of the quality of its results and in terms of response times. Finally, we give some concluding remarks and perspectives.

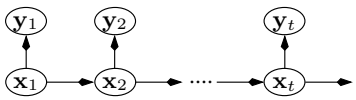


Figure 1: A Markov chain for object tracking.

2 ARTICULATED OBJECT TRACKING

In this paper, articulated object tracking consists of estimating a state sequence $\{\mathbf{x}_t\}_{t=1,\dots,T}$, whose evolution is given by equation $\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{n}_t^{\mathbf{x}})$, from observations $\{\mathbf{y}_t\}_{t=1,\dots,T}$ related to the states by $\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_t, \mathbf{n}_t^{\mathbf{y}})$. Usually, $\mathbf{f}_t$ and $\mathbf{h}_t$ are nonlinear functions, and $\mathbf{n}_t^{\mathbf{x}}$ and $\mathbf{n}_t^{\mathbf{y}}$ are i.i.d. noise sequences. From a probabilistic viewpoint, this problem can be represented by the Markov chain of Fig. 1 and it amounts to estimate, for any t, $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ where $\mathbf{x}_{1:t}$ denotes the tuple $(\mathbf{x}_1, \dots, \mathbf{x}_t)$. This can be computed iteratively using Eq. (1) and (2), which are referred to as a *prediction step* and a *correction step* respectively.

$$p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1}) \quad (1)$$

$$p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t-1}) \quad (2)$$

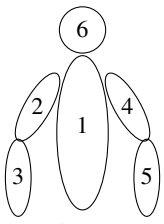
with $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ the transition corresponding to $\mathbf{f}_t$ and $p(\mathbf{y}_t|\mathbf{x}_t)$ the likelihood corresponding to $\mathbf{h}_t$.

The PF framework [Gordon et al., 1993] approximates the above densities using weighted samples $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$, $i = 1, \dots, N$, where each $\mathbf{x}_t^{(i)}$ is a possible realization of state $\mathbf{x}_t$ called a *particle*. In its prediction step (Eq. (1)), PF propagates the particle set $\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}$ using a proposal function $q(\mathbf{x}_t|\mathbf{x}_{1:t-1}^{(i)},\mathbf{y}_t)$ which may differ from $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$ (but, for simplicity, we will assume they do not); in its correction step (2), PF weights the particles using a likelihood function, so that $w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{x}_t^{(i)}) \frac{p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{1:t-1}^{(i)},\mathbf{y}_t)}$, with $\sum_{i=1}^N w_t^{(i)} = 1$. The particles can then be resampled: those with the highest weights are duplicated while the others are eliminated. The estimation of the posterior density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is then given by $\sum_{i=1}^N w_t^{(i)} \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t)$, where $\delta_{\mathbf{x}_t^{(i)}}$ are Dirac masses centered on particles $\mathbf{x}_t^{(i)}$.

As shown in [MacCormick and Isard, 2000], the number of particles necessary for a good estimation of the above densities grows exponentially with the dimension of the state space, hence making PF's basic scheme unusable in real-time for articulated object tracking. To cope with this problem, different variants of PF have been proposed, ranging from local search-based methods like the *Annealed Particle Filter* [Deutscher and Reid, 2005, Gall, 2005] and hierarchical-refining methods [Chang and Lin, 2010] to decomposition techniques like *Partitioned Sampling* (PS) [MacCormick and Blake, 1999] and its siblings [Rose et al., 2008, Besada-Portas et al., 2009]. Here, we focus on decomposition-based particle filters like PS.

PS's key idea is that the state and observation spaces $\mathcal{X}$ and $\mathcal{Y}$ can often be naturally decomposed as $\mathcal{X} = \mathcal{X}^1 \times \dots \times \mathcal{X}^P$ and $\mathcal{Y} = \mathcal{Y}^1 \times \dots \times \mathcal{Y}^P$ where each $\mathcal{X}^j$ represents some "part" of the object. For instance, on Fig. 2, a human body is decomposed as 6 parts (head, torso, etc.) numbered from 1 to 6. The state of the jth part at time t is denoted $\mathbf{x}_t^j$. Then, by exploiting conditional independences among different subspaces $(\mathcal{X}^j, \mathcal{Y}^j)$, PS estimates $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ using only sequential applications of PF over $(\mathcal{X}^j, \mathcal{Y}^j)$. For instance, on Fig. 2, given the position of the torso, the left and right arm positions may be independent so, after applying PF on the torso, PS can apply it sequentially to the left and right arms and still compute a correct estimation of $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$. As the $(\mathcal{X}^j, \mathcal{Y}^j)$ subspaces are "smaller" than $(\mathcal{X}, \mathcal{Y})$, the distributions to estimate at each iteration of PF have fewer parameters than those defined on $(\mathcal{X}, \mathcal{Y})$, which significantly reduces the number of particles needed for their estimation and, thus, speeds up the computations.

The exploitation of the conditional independences among the $(\mathcal{X}^j, \mathcal{Y}^j)$ leads to generalizing the Markov chain of Fig. 1 by the Dynamic Bayesian Network (DBN) of Fig. 2



A human body: part 1 corresponds to the torso, parts 2 and 3 to the left arm, parts 4 and 5 to the right arm and part 6 to the head. On the right side of the figure, the corresponding DBN: to the jth part corresponds a pair of state and observation variables $\mathbf{x}_t^j, \mathbf{y}_t^j$. The arcs show the dependences between variables, including between different time slices.

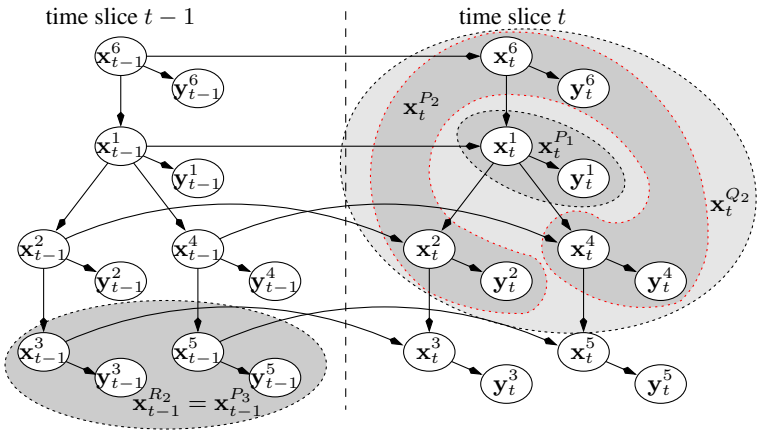


Figure 2: A Dynamic Bayesian Network.

[Murphy, 2002], in which the global state $\mathbf{x}_t$ of the object is more finely described as the set of states $\mathbf{x}_t^j$ of each part of the object. The semantics of DBNs is similar to that of Markov chains: the arcs correspond to probabilistic dependences and the joint distribution over all the nodes in the network is equal to the product of the distributions of each node conditionally to its parents in the graph.

The resampling scheme we introduce in this paper, i.e., Combinatorial Resampling, is designed to be part of PS-like algorithms and relies on DBNs. Therefore, we shall now formalize PS in terms of operations over DBNs. For this purpose, for any set $J = \{j_1, \ldots, j_k\}$, let $\mathbf{x}_t^J$ denote the tuple $(\mathbf{x}_t^{j_1}, \ldots, \mathbf{x}_t^{j_k})$, i.e., the tuple of the states of the object parts in J. For instance, on Fig. 2, if $J = \{2, 3\}$, then $\mathbf{x}_t^J$ represents the state of the whole left arm. Similarly, let $\mathbf{x}_t^{(i),J}$ denote the tuple of the parts in J of the ith particle. For instance, for $J = \{2, 3\}$, $\mathbf{x}_t^{(i),J}$ corresponds the state of left arm as represented by the ith particle. In the rest of the paper, we will assume that the object is composed of precisely P parts (in Fig. 2, $P = 6$). Now, we shall describe a slight generalization of PS where PF is iteratively applied on sets of object parts instead of just singletons like PS does. When PF is applied on a set, it is applied independently on all its elements. We need to distinguish at each step of such tracking algorithm the parts that were already processed by PF from those that were not yet. Thus, for any step j,

- let P_j denote the set of object parts being processed at the jth step (in the case of PS, $P_j = \{j\}$);
- let $Q_j = \sum_{h=1}^{j} P_h$ denote the set of all the object parts processed up to (including) the jth step;
- let $R_j = \sum_{h=j+1}^{P} P_h$ denote the set of the object parts yet to process after the jth step is completed.

Fig. 2 illustrates these notations: here, $P_1 = \{1\}$, i.e., PF is first applied only on the torso; $P_2 = \{2, 4, 6\}$, i.e., at its 2nd step, the tracking algorithm applies PF in parallel on parts 2, 4 and 6. Therefore, at the 2nd step, parts $Q_2 = \{1, 2, 4, 6\}$ have been processed and there remains to process parts $R_2 = \{3, 5\}$. Thus, if PF has propagated all the parts in Q_2 from time $t-1$ to t, in the particles, the parts in R_2 still refer to time $t-1$ (see Fig. 2). Let K denote the number of steps of the tracking algorithm, i.e., the number of sets P_j (for PS, $K = P$). To simplify the proofs in the rest of the paper, we shall fix $Q_0 = R_K = \emptyset$. Now, PS can be described in Algorithm 1. In [MacCormick and Isard, 2000], it is showed that this algorithm is mathematically sound when its resampling method is a *weighted resampling* using a g function corresponding to $p(\mathbf{y}_t|\mathbf{x}_t)$ (see the next section).

Input: A particle set $\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}$ at time $t-1$, an image $\mathcal{I}$
Output: A particle set $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ at time t
$Q \leftarrow \emptyset$; $R \leftarrow \{1, \ldots, P\}$
for $j = 1$ **to** K **do**
  foreach k **in** P_j **do**
    $Q' \leftarrow Q \cup \{k\}$; $R' \leftarrow R \backslash \{k\}$
    $\{(\mathbf{x}_t^{(i),Q'}, \mathbf{x}_{t-1}^{(i),R'})\} \leftarrow$ propagate $(\{\mathbf{x}_t^{(i),Q}, \mathbf{x}_{t-1}^{(i),R}\})$
    $\{(w_t^{(i),Q'}, w_{t-1}^{(i),R'})\} \leftarrow$
      correct $(\{(\mathbf{x}_t^{(i),Q'}, \mathbf{x}_{t-1}^{(i),R'}), (w_t^{(i),Q}, w_{t-1}^{(i),R})\}, \mathcal{I})$
    $Q \leftarrow Q'$; $R \leftarrow R'$
  $\{(\mathbf{x}_t^{(i),Q}, \mathbf{x}_{t-1}^{(i),R}), (w_t^{(i),Q}, w_{t-1}^{(i),R})\} \leftarrow$
    resample $(\{(\mathbf{x}_t^{(i),Q}, \mathbf{x}_{t-1}^{(i),R}), (w_t^{(i),Q}, w_{t-1}^{(i),R})\})$
return $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$

Algorithm 1: Partitioned Sampling PS.

3 RESAMPLING METHODS

Several resampling schemes are classically used, that we shall review briefly now. Comparisons of their pros and cons can be found in [Douc et al., 2005].

Multinomial resampling consists of selecting N numbers k_i, $i = 1, \ldots, N$, w.r.t. a uniform distribution $U((0, 1])$ on $(0, 1]$. Then, sample $\mathcal{S} = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ is substituted by a new sample $\mathcal{S}' = \{\mathbf{x}_t^{(D(k_i))}, \frac{1}{N}\}$ where $D(k_i)$ is the unique integer j such that $\sum_{h=1}^{j-1} w_t^{(h)} < k_i \leq \sum_{h=1}^{j} w_t^{(h)}$. If $(n_1, \ldots, n_N)$ denote the number of times each of the particles in $\mathcal{S}$ are duplicated, then $(n_1, \ldots, n_N)$ is distributed w.r.t. the multinomial distribution Mult$(N; w_t^{(1)}, \ldots, w_t^{(N)})$. *Stratified resampling* differs from multinomial resampling by selecting randomly the k_i's w.r.t. the uniform distribution $U((\frac{i-1}{N}, \frac{i}{N}])$. In *systematic resampling*, some number k is drawn w.r.t. $U((0, \frac{1}{N}])$ and, then, the k_i's are defined as $k_i = \frac{i-1}{N} + k$.

Residual resampling [Liu and Chen, 1998] is a method very efficient for decreasing the variance of the particle set induced by the resampling step. It is performed in two steps. First, for every $i \in \{1, \ldots, N\}$, $n'_i = \lfloor N w_t^{(i)} \rfloor$ duplicates of particle $\mathbf{x}_t^{(i)}$ of $\mathcal{S}$ are inserted into $\mathcal{S}'$. The $N - \sum_{i=1}^{n} n'_i$ particles still needed to complete the N-sample $\mathcal{S}'$ are drawn randomly using the multinomial distribution Mult$(N - \sum_{i=1}^{n} n'_i; N w_t^{(1)} - n'_1, \ldots, N w_t^{(N)} - n'_N)$, for instance using the multinomial resampling algorithm. The weights assigned to all the particles in $\mathcal{S}'$ are $1/N$.

Finally, *weighted resampling* is defined as follows: let $g : \mathcal{X} \mapsto \mathbb{R}$ be any strictly positive continuous function, where $\mathcal{X}$ denotes the state space. Weighted resampling proceeds as follows: let ρ_t be defined as $\rho_t(i) = g(\mathbf{x}_t^{(i)}) / \sum_{j=1}^{N} g(\mathbf{x}_t^{(j)})$ for $i = 1, \ldots, N$. Select independently indices $k_1, \ldots, k_N$ according to probability ρ_t. Finally, construct the new set of particles $\mathcal{S}' = \{\mathbf{x}_t^{(k_i)}, w_t^{(k_i)} / \rho_t(k_i)\}_{i=1}^{N}$. [MacCormick, 2000] shows that $\mathcal{S}'$ represents the same probability distribution as $\mathcal{S}$ while

focusing the particles on the peaks of g. Note however that, unlike the other resampling methods described above, weighted resampling does not assign equal weights $(1/N)$ to all the particles. In the rest of the paper, we will need this "equal weight" feature, so whenever weighted resampling will be used, it will be implicitly followed by one of the other above resampling methods.

In the next section, we will propose a new resampling method that exploits the structure within articulated objects to improve the efficiency of particle filtering.

4 DBN-BASED COMBINATORIAL RESAMPLING

Our resampling scheme is suitable for particle filters as described in Algo. 1. More precisely, we will show in Subsection 4.1 that, in articulated object tracking, the set $\{1,\ldots,P\}$ of parts of the objects to track can be partitioned into some sets $\{P_1,\ldots,P_K\}$ such that those parts in each P_j are all independent conditionally to $\cup_{h<j}P_h$. For instance, in Fig. 2, $P=6$ and $K=3$, $P_1=\{1\}$ corresponds to the torso, $P_2=\{2,4,6\}$ to the head and both arms, and $P_3=\{3,5\}$ to the forearms. In addition, given the position of the torso (P_1), those of the head and the arms (P_2) are independent. In Subsection 4.2, these independences will be exploited to justify that permutations of some particles' parts do not alter the estimation of $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$. Then, our resampling scheme, which will be described in Subsection 4.3, will exploit these permutations to construct implicitly some new exponential-size sample from which it will resample new high-quality samples.

4.1 IDENTIFYING SETS $P_1,\ldots,P_K$

To be sound, i.e., to not alter the estimation of $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$, Combinatorial Resampling exploits conditional independences among the different parts of the object. The partition into sets $P_1,\ldots,P_K$ precisely accounts for these independences and thus naturally results from a d-separation analysis, the independence property at the core of DBNs:

Definition 1 (*d*-separation [Pearl, 1988]) *Two nodes $\mathbf{x}_t^i$ and $\mathbf{x}_s^j$ of a DBN are dependent conditionally to a set of nodes $\mathbf{Z}$ if and only if there exists a chain, i.e., an undirected path, $\{\mathbf{c}_1=\mathbf{x}_t^i,\ldots,\mathbf{c}_n=\mathbf{x}_s^j\}$ linking $\mathbf{x}_t^i$ and $\mathbf{x}_s^j$ in the DBN such that the following two conditions hold:*

1. *for every node $\mathbf{c}_k$ such that the arcs are $\mathbf{c}_{k-1}\rightarrow\mathbf{c}_k\leftarrow\mathbf{c}_{k+1}$, either $\mathbf{c}_k$ or one of its descendants is in $\mathbf{Z}$;*
2. *none of the other nodes $\mathbf{c}_k$ belongs to $\mathbf{Z}$.*

Such a chain is called active *(else it is* blocked*). If there exists an active chain linking two nodes, these nodes are dependent and are called d-connected, otherwise they are independent conditionally to $\mathbf{Z}$ and are called d-separated.*

In Fig. 2, conditionally to the position of the torso up to time t, both arms are thus independent.

In the rest of the paper, we will assume that, *within each time slice*, the DBN structure is a directed tree, i.e., there do not exist nodes $\mathbf{x}_t^i,\mathbf{x}_t^j,\mathbf{x}_t^k$ such that $\mathbf{x}_t^i\rightarrow\mathbf{x}_t^j\leftarrow\mathbf{x}_t^k$. We will also assume that arcs across time slices link similar nodes, i.e., there exist no arc $\mathbf{x}_{t-1}^i\rightarrow\mathbf{x}_t^j$ with $j\neq i$. Finally, we will assume that nodes $\mathbf{y}_t^j$ have only one parent $\mathbf{x}_t^j$ and no children. For articulated object tracking, these requirements are rather mild and Fig. 2 satisfies all of them.

Now, we can construct sets $P_1,\ldots,P_K$: for any node, say X_t, in time slice t of the DBN, let $\mathbf{Pa}(X_t)$ and $\mathbf{Pa}_t(X_t)$ denote the set of parents of X_t in the DBN in all time slices and in time slice t only respectively. For instance, in Fig. 2, $\mathbf{Pa}(\mathbf{x}_t^2)=\{\mathbf{x}_t^1,\mathbf{x}_{t-1}^2\}$ and $\mathbf{Pa}_t(\mathbf{x}_t^2)=\{\mathbf{x}_t^1\}$. Let $\{P_1,\ldots,P_K\}$ be a partition of $\{1,\ldots,P\}$ defined by:

- $P_1=\{k\in\{1,\ldots,P\}:\mathbf{Pa}_t(\mathbf{x}_t^k)=\emptyset\}$;
- for any $j>1$, $P_j=\{k\in\{1,\ldots,P\}\backslash\bigcup_{h=1}^{j-1}P_h:\mathbf{Pa}_t(\mathbf{x}_t^k)\subseteq\bigcup_{h=1}^{j-1}\bigcup_{r\in P_h}\{\mathbf{x}_t^r\}\}$.

On Fig. 2, this results in $P_1=\{1\}$, $P_2=\{2,4,6\}$ and $P_3=\{3,5\}$. It turns out that the way we constructed the P_j's, all the $\mathbf{x}_t^k\in P_j$ can be processed independently by PF because they are independent conditionally to $\mathbf{Pa}(\mathbf{x}_t^k)$, and this is precisely this independence property which is needed to enable a sound object part swapping within Combinatorial Resampling:

Proposition 1 *The particle set resulting from Algorithm 1, with P_j defined as in the preceding paragraph, $Q_j=\sum_{h=1}^{j}P_h$ and $R_j=\sum_{h=j+1}^{K}P_h$, represents $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.*

Proof: By induction on j. Assume that, before processing parts P_j, particles estimate $p(\mathbf{x}_t^{Q_{j-1}},\mathbf{x}_{t-1}^{R_{j-1}}|\mathbf{y}_{1:t-1},\mathbf{y}_t^{Q_{j-1}})$. This is clearly the case for P_1 since P_1 are the first parts processed. Remember that P_j,Q_j,R_j are the set of parts processed at the jth step, up to the jth step and still to process respectively. We will now examine sequentially the distributions estimated by the particle set after applying in parallel PF's prediction step over the parts in P_j, then after applying PF's correction step and, finally, after resampling.

1. Let us show that after the parallel propagations of the parts in P_j (prediction step), the particle set represents density $p(\mathbf{x}_t^{Q_j},\mathbf{x}_{t-1}^{R_j}|\mathbf{y}_{1:t-1},\mathbf{y}_t^{Q_{j-1}})$. For instance, on Fig. 2, this means that, after propagating the parts in P_2, the particle set estimates $p(\mathbf{x}_t^{1,2,4,6},\mathbf{x}_{t-1}^{3,5}|\mathbf{y}_{1:t-1},\mathbf{y}_t^1)$, i.e., only the positions of the forearms still refer to time $t-1$ and the only observation taken into account at time t is the position of the torso (not yet those of the head and arms). All these

parallel operations correspond to computing:

$$\int p(\mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \prod_{k \in P_j} p(\mathbf{x}_t^k | \mathbf{Pa}(\mathbf{x}_t^k)) \, d\mathbf{x}_{t-1}^{P_j}.$$

By d-separation, a node is independent of all of its non descendants conditionally to its parents. Hence, for every $k \in P_j$, $\mathbf{x}_t^k$ is independent of $\mathbf{x}_t^{P_j \setminus \{k\}} \cup \mathbf{x}_t^{Q_{j-1}} \cup \mathbf{x}_{t-1}^{R_{j-1}} \cup \mathbf{y}_{1:t-1} \cup \mathbf{y}_t^{Q_{j-1}}$ conditionally to $\mathbf{Pa}(\mathbf{x}_t^k)$ (see Fig. 3.a where $\mathbf{x}_t^k$ is the doubly-circled node, $\mathbf{Pa}(\mathbf{x}_t^k)$ are the striped nodes and the black and shaded nodes correspond to the independent observation and state nodes respectively). Consequently, the above integral is equivalent to:

$$\begin{aligned} &\int p(\mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \\ &\quad p(\mathbf{x}_t^{P_j} | \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \, d\mathbf{x}_{t-1}^{P_j} \\ &= \int p(\mathbf{x}_t^{P_j}, \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \, d\mathbf{x}_{t-1}^{P_j}. \end{aligned}$$

As $Q_j = Q_{j-1} \cup P_j$ and $R_{j-1} = P_j \cup R_j$, the above equation is equivalent to $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})$.

2. Let us show that after the parallel corrections of the P_j parts, the particle set estimates $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. These operations correspond to computing, up to a constant, distribution $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \times \prod_{k \in P_j} p(\mathbf{y}_t^k | \mathbf{x}_t^k)$. By d-separation, nodes $\mathbf{y}_t^k$ are independent of the rest of the DBN conditionally to $\mathbf{x}_t^k$, so $p(\mathbf{y}_t^{P_j} | \mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) = \prod_{k \in P_j} p(\mathbf{y}_t^k | \mathbf{x}_t^k)$. After the corrections over P_j, the particle set thus estimates $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_t^{P_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})$, which, when normalized, is equal to $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}, \mathbf{y}_t^{P_j}) = p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. As resamplings do not alter densities, at the end of the algorithm, the particle set estimates $p(\mathbf{x}_t^{Q_K}, \mathbf{x}_{t-1}^{R_K} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_K}) = p(\mathbf{x}_t | \mathbf{y}_{1:t})$. □

4.2 SUBSTATE PERMUTATIONS

The advantage of using the P_j's as defined above instead of singletons as the classical PS does is that this enables to improve by permutations the particle set without altering the joint posterior density. Those permutations are the core of Combinatorial Resampling as the latter creates implicitly new particle sets resulting from all the possible permutations. The next proposition determines the permutations that guarantee the distributions are not altered. Intuitively, it asserts that whenever two particles are such that they have the same states on some nodes $\mathbf{Pa}_s(\mathbf{x}_s^k)$, then swapping their states on $\mathbf{x}_s^k$ and its descendants cannot alter the density estimated by the particle set. For instance, on Fig. 3.b, if two particles have the same value for the striped nodes $\mathbf{Pa}_s(\mathbf{x}_s^k)$, their values on the shaded node $\mathbf{x}_s^k$ and the black one ($\mathbf{x}_s^k$'s descendant) can be safely swapped.

Proposition 2 *Let $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}$ be the particle set at the jth step of Algo. 1. Let $k \in P_j$ and let $\mathbf{Desc}_t(\mathbf{x}_t^k)$ be the set of descendants of $\mathbf{x}_t^k$ in time slice t. Let $\sigma : \{1, \ldots, N\} \mapsto \{1, \ldots, N\}$ be any permutation such that $\mathbf{x}_s^{(i),h} = \mathbf{x}_s^{(\sigma(i)),h}$ for all the nodes $\mathbf{x}_s^h \in \cup_{s=1}^t \mathbf{Pa}_s(\mathbf{x}_s^k)$. Then, the particle set resulting from the application of σ on the parts of $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}$ corresponding to $\{\mathbf{x}_t^k\} \cup \mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)$ still estimates $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$.*

Proof: If $j = 1$, the proposition trivially holds since σ is applied to all the nodes of the connected component of $\mathbf{x}_t^k$. Assume now that $j \neq 1$. We shall now partition the object parts as described on Fig. 3.c to highlight which parts shall be permuted, which ones shall be identical to enable permutations and which parts are unconcerned: let $\mathbf{x}_{t-1}^D = \mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)$, $\mathbf{x}_t^{k'} = \mathbf{Pa}_t(\mathbf{x}_t^k)$, $\mathbf{x}_t^V = \mathbf{x}_t^{Q_j} \setminus (\{\mathbf{x}_t^k, \mathbf{x}_t^{k'}\})$ and $\mathbf{x}_{t-1}^W = \mathbf{x}_{t-1}^{R_j} \setminus \mathbf{x}_{t-1}^D$. Thus, the permuted parts are $\mathbf{x}_t^k \cup \mathbf{x}_{t-1}^D$ (see Fig. 3.c), the identical part is $\mathbf{x}_t^{k'}$, and the unconcerned parts are $\mathbf{x}_t^V \cup \mathbf{x}_{t-1}^W$.

$$\begin{aligned} p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j}) &\propto p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j}) \\ &= p(\mathbf{x}_t^{\{k,k'\} \cup V}, \mathbf{x}_{t-1}^{D \cup W}, \mathbf{y}_{1:t}^{\{k,k'\} \cup V}, \mathbf{y}_{1:t-1}^{D \cup W}) \\ &= \int p(\mathbf{x}_t^{\{k\} \cup V}, \mathbf{x}_{1:t}^{k'}, \mathbf{x}_{t-1}^{D \cup W}, \mathbf{y}_{1:t}^{\{k,k'\} \cup V}, \mathbf{y}_{1:t-1}^{D \cup W}) d\mathbf{x}_{1:t-1}^{k'} \end{aligned}$$

Conditionally to $\{\mathbf{x}_{1:t}^{k'}\}$, $S = \{\mathbf{x}_t^k\} \cup \mathbf{x}_{t-1}^D \cup \mathbf{y}_{1:t}^k \cup \mathbf{y}_{1:t-1}^D$ is independent of the rest of the DBN because, by Definition 1, no active chain can pass through an arc outgoing from a node in a conditioning set and, removing from the DBN the arcs outgoing from $\{\mathbf{x}_{1:t}^{k'}\}$, S is not connected anymore to the rest of the DBN. For the same reason,

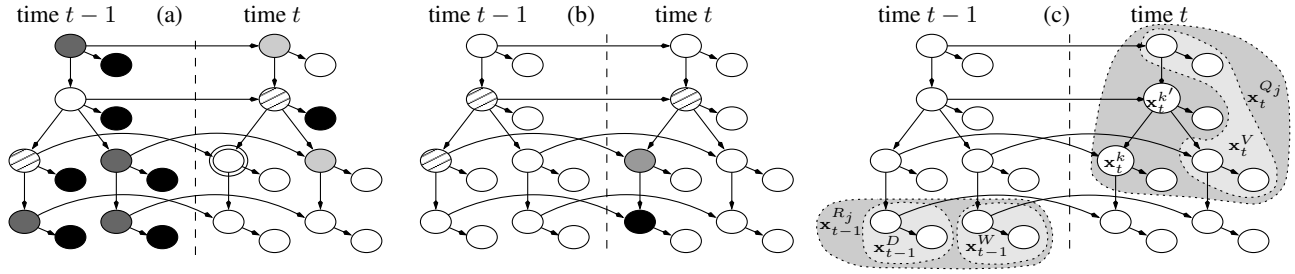


Figure 3: d-separation analysis.

$\mathbf{x}_t^V \cup \mathbf{x}_{t-1}^W \cup \mathbf{y}_{1:t}^V \cup \mathbf{y}_{1:t-1}^W$ is independent of the rest of the DBN conditionally to $\{\mathbf{x}_{1:t}^{k'}\}$. Therefore, the above integral is equal to:

$$\int p(\mathbf{x}_{1:t}^{k'}, \mathbf{y}_{1:t}^{k'})\, p(\mathbf{x}_t^k, \mathbf{x}_{t-1}^D, \mathbf{y}_{1:t}^k, \mathbf{y}_{1:t-1}^D | \mathbf{x}_{1:t}^{k'}) \; p(\mathbf{x}_t^V, \mathbf{x}_{t-1}^W, \mathbf{y}_{1:t}^V, \mathbf{y}_{1:t-1}^W | \mathbf{x}_{1:t}^{k'})\, d\mathbf{x}_{1:t-1}^{k'}. \quad (3)$$

Permuting particles over parts $\{\mathbf{x}_t^k\} \cup \mathbf{x}_{t-1}^D$ for fixed values of $\mathbf{x}_{1:t}^{k'}$ cannot change the estimation of density $p(\mathbf{x}_t^k, \mathbf{x}_{t-1}^D, \mathbf{y}_{1:t}^k, \mathbf{y}_{1:t-1}^D | \mathbf{x}_{1:t}^{k'})$ because estimations by samples are insensitive to the order of the elements in the samples. Moreover, it can neither affect the estimation of density $p(\mathbf{x}_t^V, \mathbf{x}_{t-1}^W, \mathbf{y}_{1:t}^V, \mathbf{y}_{1:t-1}^W | \mathbf{x}_{1:t}^{k'})$ since $\mathbf{x}_t^V \cup \mathbf{x}_{t-1}^W \cup \mathbf{y}_{1:t}^V \cup \mathbf{y}_{1:t-1}^W$ is independent of $\{\mathbf{x}_t^k\} \cup \mathbf{x}_{t-1}^D$ conditionally to $\{\mathbf{x}_{1:t}^{k'}\}$. Consequently, applying permutation σ on parts $\{\mathbf{x}_t^k\} \cup \mathbf{x}_{t-1}^D$ does not change the estimation of Eq. (3) and, therefore, of $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. □

As shown in the next subsection, these permutations can be exploited at the resampling level to improve samples.

4.3 OUR RESAMPLING APPROACH

All the permutations satisfying Proposition 2 can be applied to the particle set without altering the estimation of the posterior density. For instance, let $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ be two particles whose torso positions are identical, then swapping their left arm and forearm positions $(\mathbf{x}_t^2, \mathbf{x}_{t-1}^3)$ cannot alter the density estimation. Similarly, the latter is unaffected by duplications of all the particles within a particle set. This leads to Combinatorial Resampling:

Definition 2 (Combinatorial Resampling) *Let S be the particle set at the jth step of Algo. 1. For any $k \in P_j$, let Σ_k be the set of permutations satisfying Proposition 2. Let $\Sigma = \prod_{k \in P_j} \Sigma_k$. Let $S' = \cup_{\sigma \in \Sigma}\{$particle set resulting from the application of σ to $S\}$. Combinatorial resampling consists of applying any resampling algorithm over the combinatorial set S' instead of S.*

On the example of Fig. 2, let $\mathbf{x}_t^{(1)} = \langle 1, 2, 3, 4, 5, 6\rangle$, $\mathbf{x}_t^{(2)} = \langle 1, 2', 3', 4', 5', 6'\rangle$ and $\mathbf{x}_t^{(3)} = \langle 1'', 2'', 3'', 4'', 5'', 6''\rangle$ be three particles, where each number, $1, 1'', 2, 2', 2''$, etc., corresponds to the state of a part in Fig. 2. Assume that $S = \{\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \mathbf{x}_t^{(3)}\}$ at the 2nd step of Algo. 1, i.e., the object parts just processed are $P_2 = \{2, 4, 6\}$. Parts $\{2, 3\}$, $\{4, 5\}$ and $\{6\}$ can be permuted in $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ because their torso, i.e. 1, are identical, hence S' is the union of the result of all such permutations over S and is thus equal to:

$\langle 1,2\ ,3\ ,4\ ,5\ ,6\ \rangle\ \langle 1,2',3',4',5',6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle$
$\langle 1,2\ ,3\ ,4\ ,5\ ,6'\rangle\ \langle 1,2',3',4',5',6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle$
$\langle 1,2\ ,3\ ,4',5',6\ \rangle\ \langle 1,2',3',4\ ,5\ ,6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle$
$\langle 1,2\ ,3\ ,4',5',6'\rangle\ \langle 1,2',3',4\ ,5\ ,6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle$
$\langle 1,2',3',4\ ,5\ ,6\ \rangle\ \langle 1,2\ ,3\ ,4',5',6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle$
$\langle 1,2',3',4\ ,5\ ,6'\rangle\ \langle 1,2\ ,3\ ,4',5',6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle$
$\langle 1,2',3',4',5',6\ \rangle\ \langle 1,2\ ,3\ ,4\ ,5\ ,6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle$
$\langle 1,2',3',4',5',6'\rangle\ \langle 1,2\ ,3\ ,4\ ,5\ ,6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle$

Constructing S' in extension is impossible in practice because $|\Sigma|$ grows exponentially with N, the number of particles. Fortunately, we can sample over S' without actually constructing it. We shall explain the idea on the particle set S illustrated on Fig. 4, which corresponds to the object of Fig. 2 in which we omitted the head part for clarity reasons. Assume that parts $P_j = \{3, 5\}$, i.e., the forearms, have just been processed and we wish to sample over combinatorial sample S' induced by S. To construct a new particle, the idea is to first select a value for the parts in Q_{j-1}, i.e., those processed at previous steps by PF and in which no permutation will occur. Here, $Q_{j-1} = \{1, 2, 4\}$. We thus first determine the different values of $\mathbf{x}_t^{Q_{j-1}}$ in S and partition S into sets $S_1, \ldots, S_R$ such that all the particles in each set S_h have the same value for $\mathbf{x}_t^{Q_{j-1}}$ (see Fig. 4). In this figure, S_1 thus contains the first two particles since their values on $\mathbf{x}_t^{Q_{j-1}}$ are both $\langle 1, 0, 3\rangle$. To each such set S_h is assigned a weight W_h defined below so that picking the value of $\mathbf{x}_t^{Q_{j-1}}$ in S_h w.r.t. weight W_h results in a particle set estimating the same distribution as that of S. Once the value of $\mathbf{x}_t^{Q_{j-1}}$ has been chosen, there just remains to pick independently values for each $\mathbf{x}_t^k$ and their descendants, $k \in P_j$, that are compatible with that chosen for $\mathbf{x}_t^{Q_{j-1}}$. Thus, for any $h \in \{1, \ldots, R\}$, let S_h^k denote the set of particles in S whose kth part value is compatible with the value of $\mathbf{x}_t^{Q_{j-1}}$ in S_h. By Proposition 2, S_h^k is the set of particles in S that have the same value of $\mathbf{Pa}_t(\mathbf{x}_t^k)$ as those in S_h. For instance, in Fig. 4, S_1^3 is the set of the first 3 particles because all of them have value 1 on part 2.

Now, to determine the aforementioned weights W_h, there just needs to count how many times the combinatorial set has duplicated S_h. So, let $N_1, \ldots, N_R$ and $N_1^k, \ldots, N_R^k$ denote the sizes of $S_1, \ldots, S_R$ and $S_1^k, \ldots, S_R^k$ respectively. Finally, let $N^k = \max\{N_1^k, \ldots, N_R^k\}$ and, for any $h \in \{1, \ldots, R\}$, let W_h^k denote the sum of the weights assigned to the kth part of the particles in S_h^k, i.e., $W_h^k = \sum_{\mathbf{x}_t^{(i)} \in S_h^k} w^{(i),k}$. Then, as proved below, for any h,

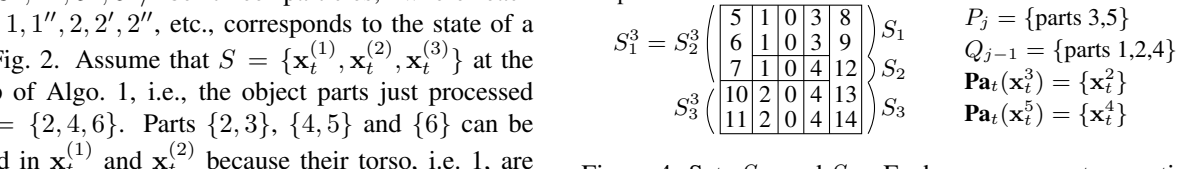


Figure 4: Sets $S_{\mathbf{i}_h}$ and $S_{\mathbf{i}_h^k}$. Each row represents a particle $\mathbf{x}_t^{(i)}$ and each number a value $\mathbf{x}_t^{(i),j}$ of part j of the particle.

Input: A particle set $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j}), w_t^{(i)}\}_{i=1}^N$

Output: A new particle set $\{(\mathbf{x}_t^{\prime\prime(i),Q_j}, \mathbf{x}_{t-1}^{\prime\prime(i),R_j}), w_t^{\prime\prime(i)}\}_{i=1}^N$

1 **for** $i = 1$ **to** N **do**
2 $\quad h \leftarrow$ sample $\{1, \ldots, R\}$ w.r.t. weights $W_1, \ldots, W_R$
3 $\quad \mathbf{x}_t^{\prime\prime(i),Q_{j-1}} \leftarrow \mathbf{x}_t^{(z),Q_{j-1}}$ where $\mathbf{x}_t^{(z)}$ is any element in S_h
4 $\quad w_t^{\prime\prime(i)} \leftarrow 1$
5 $\quad$ **foreach** k **in** P_j **do**
6 $\quad\quad \mathbf{x}_t^{(r)} \leftarrow$ sample from S_h^k w.r.t. weights $\{w_t^{(r),k}\}_{\mathbf{x}_t^r \in S_h^k}$
7 $\quad\quad \mathbf{x}_t^{\prime\prime(i),k} \leftarrow \mathbf{x}_t^{(r),k}$; $\quad w_t^{\prime\prime(i)} \leftarrow w_t^{\prime\prime(i)} \times w_t^{(r),k}$
8 $\quad\quad \mathbf{x}_{t-1}^{\prime\prime(i),\mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)} \leftarrow \mathbf{x}_{t-1}^{(r),\mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)}$
9 **return** $\{\mathbf{x}_t^{\prime\prime(i)}, w^{\prime\prime(i)}\}_{i=1}^N$

Algorithm 2: Efficient resampling over S'.

$$W_h = N_h \times \prod_{k \in P_j} \frac{N^k!}{A_{N_h^k}^{N_h}} \times A_{N_h^k - 1}^{N_h - 1} \times W_h^k, \qquad (4)$$

where $A_n^k = n!/(n-k)!$ stands for the number of k-permutations out of n elements. Resampling over S' can thus be performed efficiently as in Algo. 2. To scale-up to large particle sets, $\log(W_h)$ should be computed instead of W_h and the weights used in line 2 of Algo. 2 should be $\exp(\log W_h - \log W)$, where $W = \max\{W_1, \ldots, W_R\}$.

Proposition 3 *Algorithm 2 produces a particle set estimating the same density as that given in input.*

Proof: Let $S = \{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}_{i=1}^N$ and S' its combinatorial set (see Def. 2). In lines 2–3, Algo. 2 selects which central part Q_{j-1} particle $\mathbf{x}_t''$ should have. By definition, this amounts to selecting one set S_h w.r.t. the sum of the weights of the particles in S' having the same central part as those in S_h. Let us show that this is achieved using the weights described in Eq. (4).

In Definition 2, Σ_k is the set of all the possible permutations of the kth part of the particles in S. Clearly, within each set S_h^k, all the $N_h^k!$ permutations of the kth part of the particles of this set are admissible. They form the cycles within the permutations of Σ_k and, as such, a given permutation σ over S_h^k shall appear many times within Σ_k. There is no need to count precisely how many times σ is repeated, what is important is that the repeated sets of particles estimate the same density as S. To do so, remark that $N^k = \max\{N_1^k, \ldots, N_R^k\}$ is the size of the biggest set $S_1^k, \ldots, S_R^k$. Applying all the permutations over this set multiplies its size by $N^k!$, so the size of all the other sets should be multiplied by the same amount. Duplicating $N^k!/N_h^k!$ permutation σ guarantees that all the Q_{j-1}-central parts of the particles in S are duplicated the same number of times. Now, the particles in S_h also belong to S_h^k. As $|S_h| = N_h$, there are $A_{N_h^k}^{N_h}$ different possibilities to assign some k-part of S_h^k to the particles of S_h. The number of times these permutations are repeated within those over S_h^k is thus $N_h^k!/A_{N_h^k}^{N_h}$. Hence, duplicating $(N^k!/N_h^k!) \times (N_h^k!/A_{N_h^k}^{N_h}) = N^k!/A_{N_h^k}^{N_h}$ times the permutations over S_h ensures that the particle set estimates the same density as S. The same applies to all the other parts in P_j, hence the product in Eq. (4).

Now, let us compute the sum of the weights of the particles resulting from all the permutations over S_h. Each such permutation generates a new set of N_h particles. By symmetry, if $\mathbf{W}$ is the sum of the weights of the first particle in each set, call it $\mathbf{x}_t^{(i)}$, then the overall sum we look for is $N_h \times \mathbf{W}$. As permutations over the parts in P_j are independent, $\mathbf{W}$ is equal to the product over parts $k \in P_j$ of the sum $\mathbf{W}^k$ of the weights induced by all the permutations over the kth part, i.e., the permutations over S_h^k. By symmetry, any weight in S_h^k can be assigned to $\mathbf{x}_t^{(i)}$, hence $\mathbf{W}^k$ is equal to the sum of all these weights, W_h^k, times the number $\mathbf{O}$ of occurrences of each weight induced by all the permutations over S_h^k. For instance, if there are 3 weights 1,2,3, then there are $\mathbf{O} = 2$ permutations where the first particle as a weight of 1: $\langle 1, 2, 3\rangle$ and $\langle 1, 3, 2\rangle$. Once particle $\mathbf{x}_t^{(i)}$ has been assigned a weight, there remains $N_h - 1$ weights to assign to the other particles from a set of $N_h^k - 1$ weights, hence there are $\mathbf{O} = A_{N_h^k - 1}^{N_h - 1}$ possibilities. Overall, $\mathbf{W}^k$ is thus equal to $W_h^k \times A_{N_h^k - 1}^{N_h - 1}$ and we get Eq. (4).

So, lines 2–3 select correctly the Q_{j-1} part. Once this is done, by d-separation, all the parts in P_j are independent and should be sampled w.r.t. $p(\mathbf{x}_t^k | \mathbf{Pa}_t(\mathbf{x}_t^k))$, which is done in lines 5–8 since $p(\mathbf{x}_t^k | \mathbf{Pa}_t(\mathbf{x}_t^k)) \propto w_t^{(i),k}$. □

We shall now provide some experiments highlighting the efficiency of our resampling scheme.

5 EXPERIMENTATIONS

We performed experiments on synthetic data in order to create sequences varying the criteria whose impact on our algorithm's efficiency are the most important, i.e. the number of parts processed in parallel and the length of the object's arms. As such, this resulted in a fine picture of the behaviors of our algorithm. These results are given in Subsection 5.1. Of course, our algorithm is also effective on real sequences. This is illustrated in Subsection 5.2.

For both cases, articulated objects are modeled by a set of P polygonal parts (or regions): a central one P_1 (containing only one polygon) to which are linked $|P_j|, j > 1$, arms of length $K - 1$ (see Fig. 5 for some examples). The polygons are manually positioned in the first frame. State vectors contain the parameters describing all the parts and are defined by $\mathbf{x}_t = \{\mathbf{x}_t^1, \ldots, \mathbf{x}_t^P\}$, with $\mathbf{x}_t^k = \{x_t^k, y_t^k, \theta_t^k\}$, where (x_t^k, y_t^k) is the center of part k, and θ_t^k is its orientation, $k = 1, \ldots, P$. We thus have $|\mathcal{X}| = 3P$. A particle $\mathbf{x}_t^{(i)} = \{\mathbf{x}_t^{(i),1}, \ldots, \mathbf{x}_t^{(i),P}\}$, $i = 1, \ldots, N$, is a possible spatial configuration, i.e., a realization, of the ar-

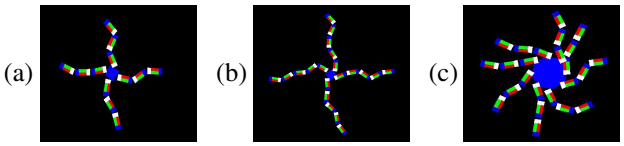
ticulated object. Particles are propagated using a random walk whose variances σ_x, σ_y and σ_θ have been empirically fixed. Particle weights are computed by measuring the similarity between the distribution of pixels in the region of the estimated part of the object and that of the corresponding reference region using the Bhattacharyya distance [Bhattacharyya, 1943]. The particle weights are then computed by $w_{t+1}^{(i)} = w_t^{(i)} p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}^{(i)}) \propto w_t^{(i)} e^{-\lambda \mathbf{d}^2}$, with, in our tests, $\lambda = 50$ and $\mathbf{d}$ the Bhattacharyya distance between the target (prior) and the reference (previously estimated) 8-bin histograms. The articulated object's distribution is estimated starting from its central part P_1.

5.1 TESTS ON SYNTHETIC VIDEO SEQUENCES

We have generated our own video sequences composed of 300 frames of 800×640 pixels. Each video displays an articulated object randomly moving and deforming over time, subject to either weak or strong motions. Some examples are given in Fig. 5. With various numbers of parts, the articulated objects are designed to test the ability of resampling to deal with high-dimensional state spaces.

We compare six different resampling approaches. The first five (multinomial, systematic, stratified, residual and weighted resampling) are integrated into PS. PS propagates and corrects particles polygon after polygon to derive a global estimation of the object. For combinatorial resampling, the object's arms are considered independent conditionally to the central part and, thus, the P_j parts, $j > 1$, correspond to the jth joints of all the arms. For weighted resampling, function g is set empirically to $g(w) = e^{20w}$ to favor the selection of high-weighted particles over low-weighted ones. Results are compared w.r.t. two criteria: computation times and estimation errors, defined as the sum of the Euclidean distances between each corner of the estimated parts and its sibling in the ground truth. For all these tests, we fixed $\sigma_x = \sigma_y = 1$ pixel and $\sigma_\theta = 0.025$ rad. All the results presented are a mean over 30 runs performed on a MacBook Pro with a 2.66 GHz Intel Core i7.

We first compared the estimation errors. Fig 6.(a-c) show a convergence study of the resampling methods depending on the number N of particles for the 3 objects of Fig. 5. For



(a) $K = 4, |P_j| = 4$, $P = 13, |\mathcal{X}| = 39$ (b) $K = 6, |P_j| = 4$, $P = 21, |\mathcal{X}| = 63$ (c) $K = 4, |P_j| = 8$, $P = 25, |\mathcal{X}| = 125$

Figure 5: Excerpts of frames from our synthetic video sequences, and the features of the corresponding articulated objects (number of arms $|P_j|$, $j > 1$, length of arms $K - 1$, total number of parts P, and dimension of state space $\mathcal{X}$).

all these tests, combinatorial resampling (CR) outperforms all the other methods: i) it converges faster (about only $N = 100$ particles are necessary to do so) when the other methods often require 300 particles to converge; ii) CR's error at convergence is much lower than that of the other methods. For instance, in Fig.6(a), CR reaches the convergence error of the other methods (about 230 pixels) with only $N = 20$ particles and, with 100 particles, its error decreases to 112 pixels. When the length of the arms (given by K) increases (Fig 6(b)), CR stays robust, whereas multinomial, systematic, stratified and residual resampling tend to fail (estimation errors twice higher). Weighted resampling seems more stable, but gives estimation errors 25% higher than those of CR. Finally, when the number of parts treated in parallel increases (Fig 6(c)), CR stays stable: with only $N = 20$ particles, its estimation error is 2.5 to 3 times lower than the one of other resampling approaches.

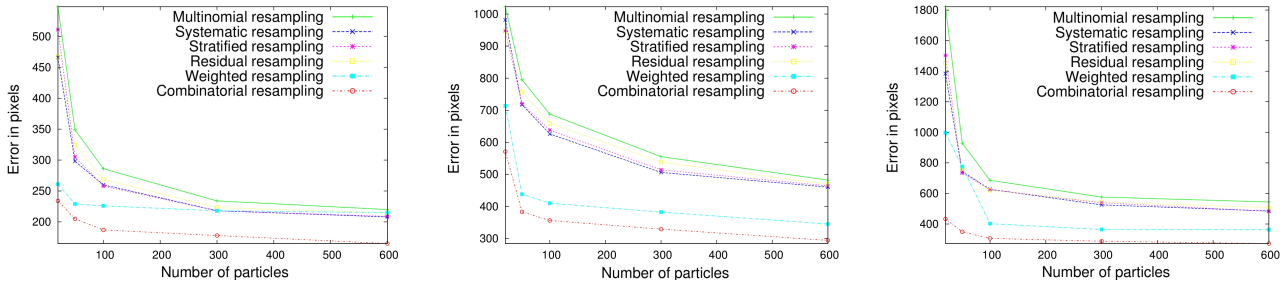
Finally, resampling times (in seconds) over the whole sequences, are reported in Table 1 for the estimation of the densities of the objects of Fig. 5(a-c) with $N = \{100, 600\}$ particles. The first four resampling approaches have similar behaviors. Due to its additional step ensuring that weights are equal to $1/N$, which is required by Algo. 1, weighted resampling is longer. The best approach is CR when the number of particles is high (600) and when the size of the P_j's is high. For instance, when tracking the object of Fig. 5(c) (8 parts processed simultaneously), the resampling times are considerably lower with CR than with the other methods. This is due to the fact that, by processing several object parts simultaneously, the number of resamplings performed is significantly reduced. Hence, even if performing CR once is longer than performing another method, overall, CR is globally faster. Note also that CR's response times increase more slowly with N than the other methods. Finally, when K increases (Fig. 5(b)), our approach also provides significantly smaller resampling times when N becomes high.

5.2 TESTS ON REAL VIDEO SEQUENCES

We tested our approach on sequences from the UCF50 dataset (http://server.cs.ucf.edu/∼vision/data/UCF50.rar), to demonstrate the efficiency of our combinatorial resampling to make the particle set better focus on the modes

Table 1: Resampling times (in seconds) for the estimation of the density of different objects, with $N = \{100, 600\}$.

	Fig. 5.a		Fig. 5.b		Fig. 5.c	
	100	600	100	600	100	600
Multinomial	0.5	17.1	1.3	46.9	1.8	79.6
Systematic	0.5	19.8	1.2	53.6	1.7	80.5
Stratified	0.5	16.9	1.3	44.8	1.7	74.9
Residual	0.5	20.3	1.3	55.7	1.8	83.4
Weighted	1.0	33.0	2.5	90.1	3.5	157.8
Combinatorial	0.7	10.6	1.5	26.3	1.5	22.3



(a) (b) (c)

Figure 6: Comparison of convergence for different resampling approaches: errors of estimation of the density of objects depending on N: (a) with $|P_i| = 4$, $K = 4$ (object of Fig. 5.(a)), (b) with $|P_i| = 4$, $K = 6$ (object of Fig. 5.(b)) and (c) with $|P_i| = 8$, $K = 4$ (object of Fig. 5.(c)).

Figure 7: Tracking results on `JumpRope` sequence with $N = 500$ particles (frames 10, 50, 121 and 234). First line, using residual resampling, last line, using our combinatorial resampling.

of the densities to estimate. This feature holds even when there are wide movements over time and when images have a low resolution. Qualitative results are given by superimposing on the frames of the sequences a red articulated object corresponding to the estimation derived from the weighted sum of the particles. For this test, we fixed $\sigma_x = \sigma_y = 2$ pixels and $\sigma_\theta = 0.08$ rad.

Figure 7 shows tracking results on the `JumpRope` sequence (containing 290 frames of 320×240 pixels) with $N = 500$ particles. In this sequence, a person is quickly moving from left to right while jumping, and crossing/uncrossing his arms and legs. For this test, we defined an articulated object with $P = 12$ parts, hence $|\mathcal{X}| = 36$, and we compared the estimations resulting from PS with a residual resampling (top line) with those resulting from our proposed resampling approach (bottom line). As can be observed, our approach produces better results: its estimations are more stable along the sequence. For example, on the images of the 2nd and 3rd columns, we can see the estimation of the articulated object fails with residual resampling but is correct with our combinatorial resampling. For this sequence, on average over 20 runs, our method needed 16 seconds while residual resampling needed 22. In addition, our algorithm proved to be more robust and provided more accurate results. As for synthetic sequences, our tests show that the higher the number of particles, the more our algorithm outperforms residual resampling in terms of response time. It is also always more accurate.

6 CONCLUSION

In this paper, we have introduced a new resampling method called *Combinatorial Resampling*. From a given sample S, this algorithm constructs implicitly a new sample S' exponentially larger than S. By construction, S' is more representative than S of the density over the whole state space and resampling from S' rather than S produces much better results, as confirmed by our experiments. We proved the mathematical correctness of the method and showed that it is effective for real time tracking. For future researches, there remains to exhibit theoretical convergence results for PS combined with this new resampling scheme.

References

[Besada-Portas et al., 2009] Besada-Portas, E., Plis, S., Cruz, J., and Lane, T. (2009). Parallel subspace sampling for particle filtering in dynamic Bayesian networks. In *Proc. of ECML PKDD*, pages 131–146.

[Bhattacharyya, 1943] Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–110.

[Chang and Lin, 2010] Chang, I.-C. and Lin, S.-Y. (2010). 3D human motion tracking based on a progressive particle filter. *Pattern Recognition*, 43(10):3621–3635.

[Chen, 2003] Chen, Z. (2003). Bayesian filtering: from Kalman filters to particle filters, and beyond.

[Deutscher and Reid, 2005] Deutscher, J. and Reid, I. (2005). Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61:185–205.

[Douc et al., 2005] Douc, R., Cappé, O., and Moulines, E. (2005). Comparison of resampling schemes for particle filtering. In *International Symposium on Image and Signal Processing and Analysis*, pages 64–69.

[Doucet et al., 2001] Doucet, A., de Freitas, N., , and Gordon, N., editors (2001). *Sequential Monte Carlo methods in practice*. Springer Verlag, New York.

[Gall, 2005] Gall, J. (2005). Generalised annealed particle filter - mathematical framework, algorithms and applications. Master's thesis, University of Mannheim.

[Gordon et al., 1993] Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. of Radar and Signal Processing*, 140(2):107–113.

[Kitagawa, 1996] Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25.

[Liu and Chen, 1998] Liu, J. and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044.

[MacCormick, 2000] MacCormick, J. (2000). *Probabilistic modelling and stochastic algorithms for visual localisation and tracking*. PhD thesis, Oxford University.

[MacCormick and Blake, 1999] MacCormick, J. and Blake, A. (1999). A probabilistic exclusion principle for tracking multiple objects. In *Proc. of ICCV*, pages 572–587.

[MacCormick and Isard, 2000] MacCormick, J. and Isard, M. (2000). Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Proc. of ECCV*, pages 3–19.

[Murphy, 2002] Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman.

[Rose et al., 2008] Rose, C., Saboune, J., and Charpillet, F. (2008). Reducing particle filtering complexity for 3D motion capture using dynamic Bayesian networks. In *Proc. of AAAI*, pages 1396–1401.

Sample-efficient Nonstationary Policy Evaluation for Contextual Bandits

Miroslav Dudík[*]
Microsoft Research
New York, NY

Dumitru Erhan
Yahoo! Labs
Sunnyvale, CA

John Langford[*]
Microsoft Research
New York, NY

Lihong Li[*]
Microsoft Research
Redmond, WA

Abstract

We present and prove properties of a new offline policy evaluator for an exploration learning setting which is superior to previous evaluators. In particular, it simultaneously and correctly incorporates techniques from importance weighting, doubly robust evaluation, and nonstationary policy evaluation approaches. In addition, our approach allows generating longer histories by careful control of a bias-variance tradeoff, and further decreases variance by incorporating information about randomness of the target policy. Empirical evidence from synthetic and real-world exploration learning problems shows the new evaluator successfully unifies previous approaches and uses information an order of magnitude more efficiently.

1 Introduction

We are interested in the "contextual bandit" setting, where on each round:

1. A vector of features (or "context") $x \in \mathcal{X}$ is revealed.
2. An action (or arm) a is chosen from a given set $\mathcal{A}$.
3. A reward $r \in [0,1]$ for the action a is revealed, but the rewards of other actions are not. In general, the reward may depend stochastically on x and a.

This setting is extremely natural, because we commonly make decisions based on some contextual information and get feedback about that decision, but not about other decisions. Prominent examples are the ad display problems at Internet advertising engines [15, 7], content recommendation on Web portals [18], as well as adaptive medical treatments. Despite a similar need for exploration, this setting is notably simpler than full reinforcement learning [25], because there is no temporal credit assignment problem—each reward depends on the current context and action only, not on previous ones.

The goal in this setting is to develop a good *policy* for choosing actions. In this paper, we are mainly concerned with *nonstationary* policies which map the current context and a history of past rounds to an action (or a distribution of actions). Note that as a special case we cover also *stationary* policies, whose actions depend on the currently observed context alone. While stationary policies are often sufficient in supervised learning, in situations with partial feedback, the most successful policies need to remember the past, i.e., they are nonstationary.

The gold standard of performance here is deploying a policy and seeing how well it actually performs. This standard can be very expensive in industrial settings and often impossible in academic settings. These observations motivate us to construct methods for scoring policies *offline* using recorded information from previously deployed policies. As a result, we can dramatically reduce the cost of deploying a new policy while at the same time rapidly speeding up the development phase through the use of benchmarks, similar to supervised learning (e.g., the UCI repository [1]) and *off-policy* reinforcement learning [21]. We would like our method to be as general and rigorous as possible, so that it could be applied to a broad range of policies.

An offline policy evaluator is usually only useful when the future behaves like the past, so we make an IID assumption: the contexts are drawn IID from an unknown distribution $D(x)$, and the conditional distribution of rewards $D(r|x,a)$ does not change over time (but is unknown). In our intended applications like medical treatments or Internet advertising, this is a reasonable assumption.

Below, we identify a few desiderata for an offline evaluator:

- *Low estimation error.* This is the first and foremost desideratum. The error typically comes from two sources—bias (due to covariate shift and/or insufficient expressivity) and variance (insufficient number of samples). Successful methods allow optimization

[*]This work was done while MD, JL, and LL were at Yahoo! Research.

of the tradeoff between these two components.

- *Incorporation of a prior reward estimator.* The evaluator should be able to take advantage of a reasonable reward estimator whenever it is available.
- *Incorporation of "scavenged exploration."* The evaluator should be able to take advantage of *ad hoc* past deployment data, with the quality of such data determining the bias introduced.

Existing Methods. Several prior evaluators have been proposed. We are going to improve on all of them.

- The direct method (DM) first builds a reward estimator $\hat{r}(x, a)$ from logged data that predicts the average reward of choosing action a in context x, and then evaluates a policy against the estimator. This straightforward approach is flexible enough to evaluate any policy, but its evaluation quality relies critically on the accuracy of the reward estimator. In practice, learning a highly accurate reward estimator is extremely challenging, rendering high bias in the evaluation results.
- Inverse Propensity Scoring (IPS) [11] and importance weighting require the log of past deployment that includes for each action the probability p with which it was chosen. The expected reward of policy π is estimated by $\frac{rI(\pi(x)=a)}{p}$, where $I(\cdot)$ is the indicator function. This formula comes up in many contexts and is built into several algorithms for learning in this setting such as EXP4 [3]. The IPS evaluator can take advantage of scavenged exploration through replacing p by an estimator $\hat{p}$ [16, 24], but it does not allow evaluation of nonstationary policies and does not take advantage of a reward estimator.
- Doubly Robust (DR) Policy Evaluation [6, 23, 22, 20, 13, 7, 8] incorporates a (possibly biased) reward estimator in the IPS approach according to:

$$(r - \hat{r}(x, a))I(\pi(x) = a)/p + \hat{r}(x, \pi(x)) \ , \quad (1.1)$$

  where $\hat{r}(x, a)$ is the estimator of the expected reward for context x and action a. The DR evaluator remains unbiased (for arbitrary reward estimator), and usually improves on IPS [8]. However, similar to IPS, it does not allow evaluation of nonstationary policies.
- Nonstationary Policy Evaluation [18, 19] uses rejection sampling (RS) to construct an unbiased history of interactions between the policy and the world. While this approach is unbiased, it may discard a large fraction of data through stringent rejection sampling, especially when the actions in the log are chosen from a highly non-uniform distribution. This can result in unacceptably large variance.

Contributions. In this paper, we propose a new policy evaluator that takes advantage of all good properties from the above approaches, while avoiding their drawbacks. As fundamental building blocks we use DR estimation, which is extremely efficacious in stationary settings, and rejection sampling, which tackles nonstationarity. We introduce two additional strategies for variance control:

- In DR, we harness the knowledge of the randomness in the evaluated policy ("revealed randomness"). Randomization is the preferred tool for handling the exploration/exploitation tradeoff and if not properly incorporated into DR, it would yield an increase in the variance. We avoid this increase without impacting bias.
- We substatially improve sample use (i.e., acceptance rate) in rejection sampling by modestly increasing the bias. Our approach allows an easy control of the bias/variance tradeoff.

As a result, we obtain an evaluator of nonstationary policies, which is extremely sample-efficient while taking advantage of reward estimators and scavenged exploration through incorporation into DR. Our incorporation of revealed randomness yields a favorable bonus: when the past data is generated by the same (or very similar) policy as the one evaluated, we accept all (or almost all) samples—a property called "idempotent self-evaluation."

After introducing our approach in Sec. 2, we analyze its bias and variance in Sec. 3, and finally present an extensive empirical evaluation in Sec. 4.

Algorithm 1 DR-ns(π, $\{(x_k, a_k, r_k, p_k)\}$, q, $c_{\max}$)

1. $h_0 \leftarrow \emptyset$, $t \leftarrow 1$, $c_1 \leftarrow c_{\max}$
   $R \leftarrow 0$, $C \leftarrow 0$, $Q \leftarrow \emptyset$
2. For $k = 1, 2, \ldots$ consider event (x_k, a_k, r_k, p_k)
   (a) $R_k \leftarrow \sum_{a'} \pi(a'|x_k, h_{t-1})\hat{r}(x_k, a') + \frac{\pi(a_k|x_k, h_{t-1})}{p_k} \cdot (r_k - \hat{r}(x_k, a_k))$
   (b) $R \leftarrow R + c_t R_k$
   (c) $C \leftarrow C + c_t$
   (d) $Q \leftarrow Q \cup \left\{ \frac{p_k}{\pi(a_k|x_k, h_{t-1})} \right\}$
   (e) Let $u_k \sim \text{Uniform}[0, 1]$
   (f) If $u_k \leq \frac{c_t \pi(a_k|x_k, h_{t-1})}{p_k}$
      i. $h_t \leftarrow h_{t-1} + (x_k, a_k, r_k)$
      ii. $t \leftarrow t + 1$
      iii. $c_t \leftarrow \min\{c_{\max}, \ q\text{-th quantile of } Q\}$
3. Return R/C

2 A New Policy Evaluator

Algorithm 1 describes our new policy evaluator DR-ns (for "doubly robust nonstationary"). Over the run of the algorithm, we process the past deployment data (exploration samples) and run rejection sampling (Steps 2e–2f) to create a simulated history h_t of the interaction between the

target policy and the environment. The algorithm returns the expected reward estimate R/C.

The algorithm takes as input a target policy to evaluate, exploration samples, and two scalars q and $c_{\max}$ which control the tradeoff between the length of h_t and bias. On each exploration sample, we use a modified DR to estimate $\mathbf{E}_\pi[r_t|x_t = x_k, h_{t-1}]$ (Step 2a). Compared with Eq. (1.1), we take advantage of revealed randomness.

The rate of acceptance in rejection sampling is controlled by the variable c_t that depends on two parameters: $c_{\max}$, controlling the maximum allowed acceptance rate, and q, which allows adaptive (policy-specific) adjustment of the acceptance rate. The meaning of q is motivated by unbiased estimation as follows: to obtain no bias, the value of c_t should never exceed the ratio $p_k/\pi(a_k|x_k, h_{t-1})$ (i.e., the right-hand side in Step 2f should never exceed one). During the run of the algorithm we keep track of the observed ratios (in Q), and q determines the quantile of the empirical distribution in Q, which we use as an upper bound for c_t. Setting $q = 0$, we obtain the unbiased case (in the limit). By using larger values of q, we increase the bias, but get longer sequences through increased acceptance rate. Similar effect is obtained by varying the value of $c_{\max}$, but the control is cruder, since it ignores the evaluated policy.

In reward estimation, we weigh the current DR estimate by the current acceptance rate, c_t (Step 2b). In unbiased case, for each simulated step t, we expect to accumulate multiple samples with the total weight of 1 in expectation. To obtain better scaling (similar to importance weighted methods), we also accumulate the sum of weights c_t in the variable C, and use them to renormalize the final estimate as R/C.

As a quick observation, if we let $c_{\max} = 1$ and evaluate a (possibly randomized nonstationary) policy on data that this policy generated, every event is accepted into the history regardless of q. Note that such aggressive all-accept strategy is unbiased for this specific "self-evaluation" setting and makes the maximal use of data. Outside self-evaluation, q has an effect on acceptance rate (stronger if exploration and target policy are more different). In our experiments, we set $c_{\max} = 1$ and rely on q to control the acceptance rate.

3 Analysis

We start with basic definitions and then proceed to analysis.

3.1 Definitions and Notation

Let $D(x)$ and $D(r|x, a)$ denote the unknown (conditional) distributions over contexts and over rewards. To simplify notation (and avoid delving into measure theory), we assume that rewards and contexts are taken from some countable sets, but our theory extends to arbitrary measurable context spaces $\mathcal{X}$ and arbitrary measurable rewards in $[0, 1]$. We assume that actions are chosen from a finite set $\mathcal{A}$ (this is a critical assumption). Our algorithm also uses an estimator of expected conditional reward $\hat{r}(x, a)$, but we do not require that this estimator be accurate. For example, one can define $\hat{r}(x, a)$ as a constant function for some value in $[0, 1]$; often the constant may be chosen 0.5 as the minimax optimum [4]. However, if $\hat{r}(x, a) \approx \mathbf{E}_D[r|x, a]$, then our value estimator will have a lower variance but unchanged bias [8]. In our analysis, we assume that $\hat{r}$ is fixed and determined before we see the data (e.g., by initially splitting the input dataset).

We assume that the input data is generated by some past (possibly nonstationary) policy, which we refer to as the "exploration policy." Contexts, actions, and rewards observed by the exploration policy are indexed by timesteps $k = 1, 2, \ldots$. The input data consists of tuples (x_k, a_k, r_k, p_k), where contexts and rewards are sampled according to D, and p_k is the logged probability with which the action a_k was chosen. In particular, we will *not* need to evaluate probabilities of choosing actions $a' \neq a_k$, nor require the full knowledge of the past policy, substantially reducing logging efforts.

Our algorithm augments tuples (x_k, a_k, r_k, p_k) by independent samples u_k from the uniform distribution over $[0, 1]$. A history up to the k-th step is denoted

$$z_k = (x_1, a_1, r_1, p_1, u_1, \ldots, x_k, a_k, r_k, p_k, u_k) \ ,$$

and an infinite history $(x_k, a_k, r_k, p_k, u_k)_{k=1}^{\infty}$ is denoted z. In our analysis, we view histories z as samples from a distribution μ. Our assumptions about data generation then translate into the assumption about factoring of μ as

$$\begin{aligned}\mu(x_k, a_k, r_k, p_k, u_k|z_{k-1}) \\ = D(x_k)\mu(a_k|x_k, z_{k-1})D(r_k|x_k, a_k) \\ I\left(p_k = \mu(a_k|x_k, z_{k-1})\right) U(u_k)\end{aligned}$$

where U is the uniform distribution over $[0, 1]$. Note that apart from the unknown distribution D, the only degree of freedom above is $\mu(a_k|x_k, z_{k-1})$, i.e., the unknown exploration policy.

When z_{k-1} is clear from the context, we use a shorthand μ_k for the distribution over the k-th tuple

$$\begin{aligned}&\mu_k(x, a, r, p, u) \\ &= \mu\left(x_k = x,\, a_k = a,\, r_k = r,\, p_k = p,\, u_k = u \;\middle|\; z_{k-1}\right) \ .\end{aligned}$$

We also write $\mathbf{P}_k^\mu$ and $\mathbf{E}_k^\mu$ for $\mathbf{P}_\mu[\cdot|z_{k-1}]$ and $\mathbf{E}_\mu[\cdot|z_{k-1}]$.

For the target policy π, we index contexts, actions, and rewards by t. Finite histories of this policy are denoted as

$$h_t = (x_1, a_1, r_1, \ldots, x_t, a_t, r_t)$$

and the infinite history is denoted h. Nonstationary policies depend on a history as well as the current context, and

hence can be viewed as describing conditional probability distributions $\pi(a_t|x_t, h_{t-1})$ for $t = 1, 2, \ldots$. In our analysis, we extend the nonstationary target policy π into a probability distribution over h defined by the factoring

$$\pi(x_t, a_t, r_t|h_{t-1}) = D(x_t)\pi(a_t|x_t, h_{t-1})D(r_t|x_t, a_t) \ .$$

Similarly to μ, we define shorthands $\pi_t(x, a, r)$, $\mathbf{P}_t^\pi$, $\mathbf{E}_t^\pi$.

We assume a continuous running of our algorithm on an infinite history z. For $t \geq 1$, let $\kappa(t)$ be the index of the t-th sample accepted in Step 2f; thus, κ converts an index in the target history into an index in the exploration history. We set $\kappa(0) = 0$ and define $\kappa(t) = \infty$ if fewer than t samples are accepted. Note that κ is a deterministic function of the history z. For simplicity, we assume that for every t, $\mathbf{P}_\mu[\kappa(t) = \infty] = 0$. This means that the algorithm generates a distribution over histories h, we denote this distribution $\hat{\pi}$.

Let $B(t) = \{\kappa(t-1)+1, \kappa(t-1)+2, \ldots, \kappa(t)\}$ for $t \geq 1$ denote the set of sample indices between the $(t-1)$-st acceptance and the t-th acceptance. This set of samples is called the t-th block. The inverse operator identifying the block of the k-th sample is $\tau(k) = t$ such that $k \in B(t)$. The contribution of the t-th block to the value estimator is denoted $R_{B(t)} = \sum_{k \in B(t)} R_k$. In our analysis, we assume a completion of T blocks, and consider both normalized and unnormalized estimators:

$$R = \sum_{t=1}^{T} c_t R_{B(t)} \ , \quad \tilde{R}^{\text{avg}} = \frac{\sum_{t=1}^{T} c_t R_{B(t)}}{\sum_{t=1}^{T} c_t |B(t)|} \ .$$

3.2 Bias Analysis

Our goal is to develop an accurate estimator. Ideally, we would like to bound the error as a function of an increasing number of exploration samples. For nonstationary policy, it can be easily shown that a single simulation trace of a policy can yield reward that is bounded away by 0.5 from the expected reward regardless of the length of simulation (see, e.g., Example 3 in [19]). Hence, even for unbiased methods, we cannot accurately estimate the expected reward from a single trace.

A simple (but wasteful) approach is to divide the exploration samples into several parts, run the algorithm separately on each part, obtaining estimates $R^{(1)}, \ldots, R^{(m)}$, and return the average $\sum_{i=1}^{m} R^{(i)}/m$. (We only consider the unnormalized estimator in this section. We assume that the division into parts is done sequentially, so that each estimate is based on the same number of blocks T.) Using standard concentration inequalities, we can then show that the average is within $O(1/\sqrt{m})$ of the expectation $\mathbf{E}_\mu[R]$. The remaining piece is then bounding the bias term $\mathbf{E}_\mu[R] - \mathbf{E}_\pi\left[\sum_{t=1}^{T} r_t\right]$.

Recall that $R = \sum_{t=1}^{T} c_t R_{B(t)}$. The source of bias are events when c_t is not small enough to guarantee that $c_t\pi(a_k|x_k, h_{t-1})/p_k$ is a probability. In this case, the probability that the k-th exploration sample is accepted is

$$p_k \min\left\{1, \tfrac{c_t\pi(a_k|x_k,h_{t-1})}{p_k}\right\} = \min\left\{p_k, c_t\pi(a_k|x_k, h_{t-1})\right\},$$

which violates the unbiasedness requirement that the probability of acceptance be proportional to $\pi(a_k|x_k, h_{t-1})$.

Let $\mathcal{E}_k$ denote this "bad" event (conditioned on z_{k-1} and the induced target history h_{t-1}):

$$\mathcal{E}_k = \{(x, a) : c_t\pi_t(a|x) > \mu_k(a|x)\} \ .$$

Associated with this event is the "bias mass" ε_k:

$$\varepsilon_k = \mathop{\mathbf{P}}_{(x,a)\sim\pi_t}[\mathcal{E}_k] - \mathop{\mathbf{P}}_{(x,a)\sim\mu_k}[\mathcal{E}_k]/c_t \ .$$

Notice that from the definition of $\mathcal{E}_k$, this mass is nonnegative. Since the first term is a probability, this mass is at most 1. We assume that this mass is bounded away from 1, i.e., that there exists ε such that for all k and z_{k-1} we have the bound $0 \leq \varepsilon_k \leq \varepsilon < 1$.

The following theorem analyzes how much bias is introduced in the worst case, as a function of ε. Its purpose is to identify the key quantities that contribute to the bias, and to provide insights of what to optimize in practice.

Theorem 1. *For $T \geq 1$,*

$$\left|\mathbf{E}_\mu\left[\sum_{t=1}^{T} c_t R_{B(t)}\right] - \mathbf{E}_\pi\left[\sum_{t=1}^{T} r_t\right]\right| \leq \frac{T(T+1)}{2} \cdot \frac{\varepsilon}{1-\varepsilon} \ .$$

Intuitively, this theorem says that if a bias of ε is introduced in round t, its effect on the sum of rewards can be felt for $T - t$ rounds. Summing over rounds, we expect to get an $O(\varepsilon T^2)$ effect on the unnormalized bias in the worst case or equivalently a bias of $O(\varepsilon T)$ on the average reward. In general a very slight bias can result in a significantly better acceptance rate, and hence longer histories (or more replicates $R^{(i)}$).

This theorem is the first of this sort for policy evaluators, although the mechanics of proving correctness are related to the proofs for model-based reinforcement-learning agents in MDPs (e.g., [14]). A key difference here is that we depend on a context with unbounded complexity rather than a finite state space.

Before proving Theorem 1, we state two technical lemmas (for proofs see Appendix B). Recall that $\hat{\pi}$ denotes the distribution over target histories generated by our algorithm.

Lemma 1. *Let $t \geq 1$, $k \geq 1$ and let z_{k-1} be such that $\kappa(t-1) = k-1$. Let h_{t-1} and c_t be the target history and acceptance ratio induced by z_{k-1}. Then:*

$$\sum_{x,a} \left|\mathbf{P}_k^\mu[x_{\kappa(t)} = x, a_{\kappa(t)} = a] - \pi_t(x, a)\right| \leq \frac{2\varepsilon}{1-\varepsilon} \ ,$$

$$\left|c_t\mathbf{E}_k^\mu[R_{B(t)}] - \mathbf{E}_t^\pi[r_t]\right| \leq \frac{\varepsilon}{1-\varepsilon} \ .$$

Lemma 2. $\sum_{h_T} |\hat{\pi}(h_T) - \pi(h_T)| \leq (2\varepsilon T) / (1-\varepsilon)$.

Proof of Theorem 1. We first bound a single term $|\mathbf{E}_{z\sim\mu}[c_t R_{B(t)}] - \mathbf{E}_{h\sim\pi}[r_t]|$ using the previous two lemmas, the triangle inequality and Hölder's inequality:

$$\begin{aligned}
&\left|\mathbf{E}_{z\sim\mu}[c_t R_{B(t)}] - \mathbf{E}_{h\sim\pi}[r_t]\right| \\
&= \left|\mathbf{E}_{z\sim\mu}\left[c_t \mathbf{E}^{\mu}_{\kappa(t)}[R_{B(t)}]\right] - \mathbf{E}_{h\sim\pi}[r_t]\right| \\
&\leq \left|\mathbf{E}_{z\sim\mu}\left[\mathbf{E}^{\pi}_t[r_t]\right] - \mathbf{E}_{h\sim\pi}\left[\mathbf{E}^{\pi}_t[r_t]\right]\right| + \frac{\varepsilon}{1-\varepsilon} \\
&= \left|\mathop{\mathbf{E}}_{h_{t-1}\sim\hat{\pi}}\left[\mathbf{E}^{\pi}_t\left[r-\frac{1}{2}\right]\right] - \mathop{\mathbf{E}}_{h_{t-1}\sim\pi}\left[\mathbf{E}^{\pi}_t\left[r-\frac{1}{2}\right]\right]\right| + \frac{\varepsilon}{1-\varepsilon} \\
&\leq \frac{1}{2}\sum_{h_{t-1}}\left|\hat{\pi}(h_{t-1}) - \pi(h_{t-1})\right| + \frac{\varepsilon}{1-\varepsilon} \\
&\leq \frac{1}{2}\cdot\frac{2\varepsilon(t-1)}{1-\varepsilon} + \frac{\varepsilon}{1-\varepsilon} = \frac{\varepsilon t}{1-\varepsilon} \ .
\end{aligned}$$

The theorem now follows by summing over t and using the triangle inequality. □

3.3 Progressive Validation

While the bias analysis in the previous section qualitatively captures the bias-variance tradeoff, it cannot be used to construct an explicit error bound. The second and perhaps more severe problem is that even if we had access to a more explicit bias bound, in order to obtain *deviation* bounds, we would need to decrease the length of generated histories by a significant factor (at least according to the simple approach discussed at the beginning of the previous section).

In this section, we show how we can use a single run of our algorithm to construct a stationary policy, whose value is estimated with an error $O(1/\sqrt{n})$ where n is the number of original exploration samples. Thus, in this case we get explicit error bound and much better sample efficiency.

Assume that the algorithm terminates after fully generating T blocks. We will show that the value R/C returned by our algorithm is an unbiased estimate of the expected reward of the randomized stationary policy π_{PV} defined by:

$$\pi_{\text{PV}}(a|x) = \sum_{t=1}^{T} \frac{c_t|B(t)|}{C}\pi(a|x, h_{t-1}) \ .$$

Conceptually, this policy first picks among the histories h_0, $\ldots$, h_{T-1} with probabilities $c_1|B(1)|/C, \ldots, c_t|B(T)|/C$, and then executes the policy π given the chosen history. We extend π_{PV} to a distribution over triples

$$\pi_{\text{PV}}(x, a, r) = D(x)\pi_{\text{PV}}(a|x)D(r|x, a) \ .$$

To analyze our estimator, we need to assume that during the run of the algorithm, the ratio $\pi_t(a_k|x_k)/\mu_k(a_k|x_k)$ is bounded, i.e., we assume there exists $M < \infty$ such that

$$\forall z,\ \forall t \geq 1,\ \forall k \in B(t): \quad \frac{\pi_t(a_k|x_k)}{\mu_k(a_k|x_k)} \leq M \ .$$

Next, fix z_{k-1} and let $t = \tau(k)$ (note that $\tau(k)$ is a deterministic function of z_{k-1}). It is not too difficult to show that R_k is an unbiased estimator of $\mathbf{E}_{r\sim\pi_t}[r]$, and to bound its range and variance (for proofs see Appendices B and C):

Lemma 3. $\mathbf{E}^{\mu}_k[R_k] = \mathbf{E}_{r\sim\pi_t}[r]$.

Lemma 4. $|R_k| \leq 1 + M$.

Lemma 5. $\mathbf{E}^{\mu}_k[R^2_k] \leq 3 + M$.

Now we are ready to show that R/C converges to the expected reward of the policy π_{PV}:

Theorem 2. *Let n be the number of exploration samples used to generate T blocks, i.e., $n = \sum_{t=1}^{T}|B(t)|$. With probability at least $1-\delta$,*

$$\left|R/C - \mathop{\mathbf{E}}_{r\sim\pi_{\text{PV}}}[r]\right| \leq \frac{nc_{\max}}{C}\cdot 2\max\left\{\frac{(1+M)\ln(2/\delta)}{n}, \sqrt{\frac{(3+M)\ln(2/\delta)}{n}}\right\}.$$

Proof. The proof follows by Freedman's inequality (Theorem 3 in Appendix A), applied to random variables $c_t R_k$, whose range and variance can be bounded using Lemmas 4 and 5 and the bound $c_t \leq c_{\max}$. □

4 Experiments

We conduct experiments on two problems, the first is a public supervised learning dataset converted into an exploration learning dataset, and the second is a real-world proprietary dataset.

4.1 Classification with Bandit Feedback

In the first set of experiments, we illustrate the benefits of DR-ns (Algorithm 1) over naive rejection sampling using the public dataset `rcv1` [17]. Since `rcv1` is a multi-label dataset, an example has the form (x, c), where x is the feature and c is the set of corresponding labels. Following the construction of previous work [4, 8], an example (x, c) in a K-class classification problem may be interpreted as a bandit event with context x, action $a \in [K] := \{1, \ldots, K\}$, and loss $l_a := I(a \notin c)$, and a classifier as an arm-selection policy whose expected loss is its classification error. In this section, we aim at evaluating average policy loss, which can be understood as negative reward. For our experiments, we only use the $K = 4$ top-level classes in `rcv1`, namely $\{C, E, G, M\}$; a random selection of $40K$ data from the whole dataset were used. Call this dataset D.

Data Conversion. To construct a partially labeled exploration data set, we choose actions non-uniformly in the following manner. For an example (x, c), a uniformly random

score $s_a \in [0.1, 1]$ is assigned to arm $a \in [K]$, and the probability of action a is

$$\mu(a|x) = \frac{0.3 \times s_a}{\sum_{a'} s_{a'}} + \frac{0.7 \times I(a \in c)}{|c|}.$$

This kind of sampling ensures two useful properties. First, every action has a non-zero probability, so such a dataset suffices to provide an unbiased offline evaluation of any policy. Second, actions corresponding to correct labels have higher observation probabilities, emulating the typical setting where a baseline system already has a good understanding of which actions are likely best.

We now consider the two tasks of evaluating a static policy and an adaptive policy. The first serves as a sanity check to see how well the evaluator works in the degenerate case of static policies. In each task, a one-vs-all reduction is used to induce a multi-class classifier from either fully or partially labeled data. In fully labeled data, each example is included in data sets of all base binary classifiers. In partially labeled data, an example is included only in the data set corresponding to the action chosen. We use the LIBLINEAR [9] implementation of logistic regression. Given a classifier that predicts the most likely label a^*, our policy follows an ε-greedy strategy with ε fixed to 0.1; that is, with probability 0.9 it chooses a^*, otherwise a random label $a \in [K]$. The reward estimator $\hat{r}$ is directly obtained from the probabilistic output of LIBLINEAR (using the "`-b 1`" option). The scaling parameter is fixed to the default value 1 (namely, "`-c 1`").

Static Policy Evaluation. In this task, we first chose a random 10% of D and trained a policy π_0 on this fully labeled data. From the remainder, we picked a random "evaluation set" containing 50% of D. The average loss of π_0 on the evaluation set served as the ground truth. A partially labeled version of the evaluation set was generated by the conversion described above; call the resulting dataset D'. Finally, various offline evaluators of π_0 were compared against each other on D'.[1] We repeated the generation of evaluation set in 300 trials to measure the bias and standard deviation of each evaluator.

Adaptive Policy Evaluation. In this task, we wanted to evaluate the average online loss of the following adaptive policy π. The policy is initialized as a specific "offline" policy calculated on random 400 fully observed examples (1% of D). Then the "online" partial-feedback phase starts (the one which we are interested in evaluating). We update the policy after every 15 examples, until 300 examples are observed. On policy update, we simply use an enlarged training set containing the initial 400 fully labeled and additional partially labeled examples. The offline training set was fixed in all trials. The remaining data was split into two portions, the first containing a random 80% of D for evaluation, the second containing 19% of D to determine the ground truth. The evaluation set was randomly permuted and then transformed into a partially labeled set D' on which evaluators were compared. The generation of D' was repeated in 50 trials, from which bias and standard deviation of each evaluator were obtained. To estimate the ground-truth value of π, we simulated π on the randomly shuffled (fully labeled) 19% of ground-truth data 2 000 times to compute its average online loss.

[1] To avoid risks of overfitting, for evaluators that estimate $\hat{r}$ (all except RS), we split D' into two equal halves, one for training $\hat{r}$, the other for running the evaluator. The same approach was taken in the adaptive policy evaluation task.

Compared Evaluators. We compared the following evaluators described earlier: DM for direct method, RS for the unbiased evaluator in [19] combined with rejection sampling, and DR-ns as in Algorithm 1 (with $c_{\max} = 1$). We also tested a variant of DR-ns, which does not monitor the quantile, but instead uses c_t equal to $\min_D \mu(a|x)$; we call it WC since it uses the worst-case (most conservative) value of c_t that ensures unbiasedness of rejection sampling.

Results. Tables 1 and 2 summarize the accuracy of different evaluators in the two tasks, including rmse (root mean squared error), bias (the absolute difference between evaluation mean and the ground truth), and stdev (standard deviation of the evaluation results in different runs). It should be noted that, given the relatively small number of trials, the measurement of bias is not statistically significant. So for instance, it cannot be inferred in a statistically significant way from Table 1 that WC enjoys a lower bias than RS. However, the tables provide 95% confidence interval for the rmse that allows a meaningful comparison.

It is clear from both tables that although rejection sampling is guaranteed to be unbiased, its variance usually is the dominating source of rmse. At the other extreme is the direct method, which has the smallest variance but often suffers high bias. In contrast, our method DR-ns is able to find a good balance between these extremes and, with proper choice of q, is able to yield much more accurate evaluation results. Furthermore, compared to the unbiased variant WC, DR-ns's bias appears to be modest.

It is also clear that the main benefit of DR-ns is its low variance, which stems from the adaptive choice of c_t values. By slightly violating the unbiasedness guarantee, it increases the effective data size significantly, hence reducing the variance of its evaluation. In particular, in the first task of evaluating a static policy, rejection sampling was able to use only 264 examples (out of the 20K data in D') since the minimum value of $\mu(a|x)$ in the exploration data was very small; in contrast, DR-ns was able to use 523, 3 375, 4 279, and 4 375 examples for $q \in \{0, 0.01, 0.05, 0.1\}$, respectively. Similarly, in the adaptive policy evaluation task, with DR-ns($q > 0$), we could extract many more online tra-

Table 1: Static policy evaluation results.

evaluator	rmse ($\pm 95\%$ C.I.)	bias	stdev
DM	0.0151 ± 0.0002	0.0150	0.0017
RS	0.0191 ± 0.0021	0.0032	0.0189
WC	$\mathbf{0.0055 \pm 0.0006}$	0.0001	0.0055
DR-ns($q = 0$)	0.0093 ± 0.0010	0.0032	0.0189
DR-ns($q = 0.01$)	0.0057 ± 0.0006	0.0021	0.0053
DR-ns($q = 0.05$)	$\mathbf{0.0055 \pm 0.0006}$	0.0022	0.0051
DR-ns($q = 0.1$)	0.0058 ± 0.0006	0.0017	0.0055

Table 2: Adaptive policy evaluation results.

evaluator	rmse ($\pm 95\%$ C.I.)	bias	stdev
DM	0.0329 ± 0.0007	0.0328	0.0027
RS	0.0179 ± 0.0050	0.0007	0.0181
WC	0.0156 ± 0.0037	0.0086	0.0132
DR-ns($q = 0$)	0.0129 ± 0.0034	0.0046	0.0122
DR-ns($q = 0.01$)	$\mathbf{0.0089} \pm 0.0017$	0.0065	0.0062
DR-ns($q = 0.05$)	0.0123 ± 0.0017	0.0107	0.0061
DR-ns($q = 0.1$)	0.0946 ± 0.0015	0.0946	0.0053

jectories of length 300 for evaluating π, while RS and WC were able to find only one such trajectory out of the evaluation set. In fact, if we increased the trajectory length of π from 300 to 500, neither RS or WC could construct a full trajectory of length 500 and failed the task completely.

4.2 Content Slotting in Response to User Queries

In this set of experiments, we evaluate two policies on a proprietary real-world dataset consisting of web search queries, various content that is displayed on the web page in response to these queries, and the feedback that we get from the user (as measured by clicks) in response to the presentation of this content. Formally, this partially labeled data consists of tuples (x_k, a_k, r_k, p_k), where x_k is a query and corresponding features, $a_k \in \{$web-link,news,movie$\}$ (the content shown at slot 1 on the results page), r_k is a so-called click-skip reward ($+1$ if the result was clicked, -1 if a result at a *lower* slot was clicked), and p_k is the recorded probability with which the exploration policy chose the given action.

The page views corresponding to these tuples represent a small percentage of traffic for a major website; any given page view had a small chance of being part of this experimental bucket. Data was collected over a span of several days during July 2011. It consists of 1.2 million tuples, out of which the first 1 million were used for estimating $\hat{r}$ with the remainder used for policy evaluation. For estimating the variance of the compared methods, the latter set was divided into 10 independent test subsets of equal size.

Two policies were compared in this setting: *argmax* and *self-evaluation* of the exploration policy. For *argmax* policy, we first obtained a linear estimator $r'(x, a) = w_a \cdot x$ by importance-weighted linear regression (with importance weights $1/p_k$). The *argmax* policy chooses the action with the largest predicted reward $r'(x, a)$. Note that both $\hat{r}$ and r' are linear estimators obtained from the training set, but $\hat{r}$ was computed without importance weights (and we therefore expect it to be more biased). *Self-evaluation* of the exploration policy was performed by simply executing the exploration policy on the evaluation data.

Table 3: Estimated rewards reported by different policy evaluators on two policies for a real-world exploration problem. In the first column results are normalized by the (known) expected reward of the deployed policy. In the second column results are normalized by the reward reported by IPS. All $\pm$ are computed standard deviations over results on 10 disjoint test sets.

evaluator	*self-evaluation*	*argmax* policy
RS	0.986 ± 0.060	0.990 ± 0.048
IPS	0.995 ± 0.041	1.000 ± 0.027
DM	1.213 ± 0.010	1.211 ± 0.002
DR	0.967 ± 0.042	0.991 ± 0.026
DR-ns	0.974 ± 0.039	0.993 ± 0.024

Table 3 compares RS [19], IPS, DM, DR [8], and DR-ns($c_{\max} = 1$, $q = 0.1$). For business reasons, we do not report the estimated reward directly, but normalize to either the empirical average reward (for *self-evaluation*) or the IPS estimate (for the *argmax* policy evaluation).

In both cases, the RS estimate has a much larger variance than the other estimators. Note that the minimum observed p_k equals $1/13$, which indicates that a naive rejection sampling approach would suffer from the data efficiency problem. Indeed, out of approximately 20 000 samples per evaluation subset, about 900 are added to the history for the *argmax* policy. In contrast, the DR-ns method adds about 13 000 samples, a factor of 14 improvement.

The experimental results are generally in line with theory. The variance is smallest for DR-ns, although IPS does surprisingly well on this data, presumably because values $\hat{r}$ in DR and DR-ns are relatively close to zero, so the benefit of $\hat{r}$ is diminished. The Direct Method (DM) has an unsurprisingly huge bias, while DR and DR-ns appear to have a very slight bias, which we believe may be due to imperfect logging. In any case, DR-ns dominates RS in terms of variance as it was designed to do, and has smaller bias and variance than DR.

5 Conclusion and Future Work

We have unified best-performing stationary policy evaluators and rejection sampling by carefully preserving their best parts and eliminating the drawbacks. To our knowl-

edge, the resulting approach yields the best evaluation method for nonstationary and randomized policies, especially when reward predictors are available.

Yet, there are definitely opportunities for further improvement. For example, consider nonstationary policies which can devolve into a round-robin action choices when the rewards are constant (such as UCB1 [2]). A policy which cycles through actions has an expected reward equivalent to a randomized policy which picks actions uniformly at random. However, for such a policy, our policy evaluator will only accept on average a fraction of $1/K$ uniform random exploration events. An open problem is to build a more data-efficient policy evaluator for this kind of situations.

References

[1] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

[3] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Computing*, 32(1):48–77, 2002.

[4] Alina Beygelzimer and John Langford. The offset tree for learning with partial labels. In *KDD*, pages 129–138, 2009.

[5] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandit algorithms with supervised learning guarantees. In *AISTATS*, 2011.

[6] Claes M. Cassel, Carl E. Särndal, and Jan H. Wretman. Some results on generalized difference estimation and generalized regression estimation for finite populations. *Biometrika*, 63:615–620, 1976.

[7] David Chan, Rong Ge, Ori Gershony, Tim Hesterberg, and Diane Lambert. Evaluating online ad campaigns in a pipeline: Causal models at scale. In *KDD*, 2010.

[8] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *ICML*, 2011.

[9] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[10] David A. Freedman. On tail probabilities for martingales. *Annals of Probability*, 3(1):100–118, 1975.

[11] D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *J. Amer. Statist. Assoc.*, 47:663–685, 1952.

[12] Sham Kakade, Michael Kearns, and John Langford. Exploration in metric state spaces. In *ICML*, 2003.

[13] Joseph D. Y. Kang and Joseph L. Schafer. Demystifying double robustness: A comparison of alternative strategies for estimating a population mean from incomplete data. *Statist. Sci.*, 22(4):523–539, 2007. With discussions.

[14] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. In *ICML*, 1998.

[15] Diane Lambert and Daryl Pregibon. More bang for their bucks: Assessing new features for online advertisers. In *ADKDD*, 2007.

[16] John Langford, Alexander L. Strehl, and Jennifer Wortman. Exploration scavenging. In *ICML*, pages 528–535, 2008.

[17] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[18] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010.

[19] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, 2011.

[20] Jared K. Lunceford and Marie Davidian. Stratification and weighting via the propensity score in estimation of causal treatment effects: A comparative study. *Statistics in Medicine*, 23(19):2937–2960, 2004.

[21] Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *ICML*, pages 759–766, 2000.

[22] James M. Robins and Andrea Rotnitzky. Semiparametric efficiency in multivariate regression models with missing data. *J. Amer. Statist. Assoc.*, 90:122–129, 1995.

[23] James M. Robins, Andrea Rotnitzky, and Lue Ping Zhao. Estimation of regression coefficients when some regressors are not always observed. *J. Amer. Statist. Assoc.*, 89(427):846–866, 1994.

[24] Alex Strehl, John Langford, Lihong Li, and Sham Kakade. Learning from logged implicit exploration data. In *NIPS*, pages 2217–2225, 2011.

[25] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, March 1998.

Uniform Solution Sampling Using a Constraint Solver As an Oracle*

Stefano Ermon
Department of Computer Science
Cornell University
ermonste@cs.cornell.edu

Carla Gomes
Department of Computer Science
Cornell University
gomes@cs.cornell.edu

Bart Selman
Department of Computer Science
Cornell University
selman@cs.cornell.edu

Abstract

We consider the problem of sampling from solutions defined by a set of hard constraints on a combinatorial space. We propose a new sampling technique that, while enforcing a uniform exploration of the search space, leverages the reasoning power of a systematic constraint solver in a black-box scheme. We present a series of challenging domains, such as energy barriers and highly asymmetric spaces, that reveal the difficulties introduced by hard constraints. We demonstrate that standard approaches such as Simulated Annealing and Gibbs Sampling are greatly affected, while our new technique can overcome many of these difficulties. Finally, we show that our sampling scheme naturally defines a new approximate model counting technique, which we empirically show to be very accurate on a range of benchmark problems.

1 Introduction

In recent years, we have seen significant interest in probabilistic reasoning approaches that combine both hard, logical constraints and probabilistic information through soft, weighted constraints. Markov Logic Networks [1] are a prominent example of such a modeling language. The hard constraints are used to capture definitional and other irrefutable relationships in the underlying domain, while soft constraints capture less categorical information, generally better for modeling real-world data and dependencies.

Markov Chain Monte Carlo (MCMC) methods, such as Simulated Annealing (SA) and Gibbs Sampling [2], are among the most prominent approaches to probabilistic reasoning, especially when exact inference is beyond reach. In fact, they are guaranteed to asymptotically converge to a stationary distribution that can theoretically provide uniform samples. However, sampling from the solutions of a set of hard constraints is believed to be very hard in worst case, as it is closely related to #-P complete problems such as model counting [3, 4, 5, 6, 7]. In particular, the time required for a Markov Chain to reach the stationary distribution (mixing time) is often exponential in the number of problem variables when hard constraints are present. These difficulties led to the introduction of alternate sampling strategies. For example, in the SampleSAT approach [8], a Markov Chain is constructed that combines moves proposed according to a Simulated Annealing Markov Chain with so-called WalkSAT moves, inspired by local search constraint solvers. The SampleSAT method provides a significant advance, since SA sampling on constraint problems of practical interest generally does not reach any solutions or, at best, only a small subset of all possible solutions. SampleSAT was subsequently incorporated into MC-SAT in the Alchemy package for reasoning and learning for Markov Logic Networks [9]. A key limitation of SampleSAT is that it is not guaranteed to converge to the uniform distribution on the solution set. In fact, one can create examples where the stationary distribution of SampleSAT is arbitrarily biased (non-uniform) in terms of solution samples. Still, this does not negate the value of SampleSAT in many practical settings. For, although SA-like sampling does sample in the limit from the uniform stationary distribution on the solution set, reaching the stationary distribution often requires exponential time, making the SA strategy of little use in practice.

In this paper, we revisit the question of how to devise practical methods for sampling from solutions defined by a set of hard constraints on a combinatorial space. We present a series of challenging domains that reveal the difficulties introduced by such constraints. These include high energy[1] barriers, large energy plateaus ("golf-course" like energy landscapes), and highly asymmetric sampling spaces. We present data demonstrating that SA, Gibbs sampling, and SampleSAT [8] are all greatly affected by these problems.

*This work was supported by NSF Grant 0832782.

[1] Energy is defined as the number of violated constraints.

However, we also show that highly structured energy landscapes can actually present new opportunities for solution samplers. Combinatorially defined energy functions have a rich structure embedded, but traditional MCMC methods are very general and therefore treat the energy as a "black-box", effectively ignoring the underlying structure. On the other hand, modern day constraint solvers use clever heuristic to exploit constraint structure as much as possible, and can solve very large structured industrial problems with millions of variables [10]. However, these solvers cannot be used directly as solution samplers because they tend to oversample certain solutions, as they are designed just to find one satisfying assignment but not to be uniform [8].

In this paper, we propose a novel sampling scheme called `SearchTreeSampler`, which leverages the reasoning power of a systematic constraint solver while enforcing a uniform exploration of the search space. Constraint solvers have been previously applied by SampleSearch [11, 12, 13] in the context of importance sampling, a framework where the performance is known to heavily depend on the choice of the proposal distribution. In contrast, `SearchTreeSampler` introduces a new way of exploring the search space that does not rely on a heuristically chosen proposal distribution, and directly provides (approximately) uniform samples. The constraint solver is used as a black-box, so that any systematic solver can be plugged in, with no modifications required. We empirically demonstrate that by leveraging constraint structure, `SearchTreeSampler` can overcome many of the difficulties encountered by other solution samplers. In particular, we show it can be orders of magnitude faster than competing methods, while providing more uniform samples at the same time. Further, we show that our sampling scheme naturally defines a new technique for approximately counting the number of distinct solutions (model counting), that we empirically show to be very accurate on a range of benchmark problems.

2 Problem Definition

We consider the problem of sampling from a combinatorial search space defined by n Boolean variables and m constraints specified by a Boolean formula F in conjunctive normal form (CNF). A constraint or *clause* C is a logical disjunction of a set of (possibly negated) variables. A formula F is said to be in CNF form if it is a logical conjunction of a set of clauses $\mathcal{C}$.

We define V to be the set of propositional variables in the formula, where $|V| = n$. A variable assignment $\sigma : V \to \{0, 1\}$ is a function that assigns a value in $\{0, 1\}$ to each variable in V. As usual, the value 0 is interpreted as FALSE and the value 1 as TRUE. Let F be a formula in CNF over the set V of variables with $m = |\mathcal{C}|$ clauses and let σ be a variable or truth assignment. We say that σ satisfies a clause C if at least one signed variable of C is TRUE. We say that a truth assignment σ is a satisfying assignment for F (also called a model or a solution) if σ satisfies all the clauses $C \in \mathcal{C}$. Let $\mathcal{S}_F$ be the set of solutions of F, and let $Z = |\mathcal{S}_F|$ be the number of distinct solutions.

Given a Boolean formula F, we define a discrete probability distribution D over the set of all possible truth assignments $\{0, 1\}^n$ such that

$$D(\sigma) = \begin{cases} 1/Z & \text{if } \sigma \in \mathcal{S}_F \text{ (i.e., } \sigma \text{ is a solution)} \\ 0 & \text{otherwise} \end{cases}$$

In this paper we consider the problem of sampling from D. This problem is very hard and in fact simply deciding whether or not the support of D is empty is NP-complete (the CNF-SAT problem). Sampling is however believed to be even harder, as it is closely related to $\#$-P complete problems such as inference and model counting [3, 4]. For instance, SAT solvers cannot be directly used as solution samplers because they tend to oversample certain solutions, as they are designed just to find one satisfying assignment but not to be uniform [8].

3 Background on Solution Sampling

In this section, we briefly describe the main techniques for solution sampling.

3.1 Simulated Annealing

Simulated Annealing is a MCMC algorithm that defines a reversible Markov Chain on the space of truth assignments $\{0, 1\}^n$ to sample from a Boltzmann distribution [14]. The transition probabilities (and the steady state probability distribution) depend only on a property of the truth assignments called "energy". The energy $E : \{0, 1\}^n \to \mathbb{N}$ gives the number of clauses violated by a truth assignment σ and is defined as follows

$$E(\sigma) = |\{c \in \mathcal{C} | \sigma \text{ does not satisfy } c\}|.$$

The Boltzmann steady state probability distribution is given by

$$P_T(\sigma) = \frac{1}{Z(T)} e^{-\frac{E(\sigma)}{T}},$$

where T is a formal parameter called "temperature", and $Z(T)$ is the normalization constant. Notice that P_T assigns the same probability to all solutions, i.e. $P_T(\sigma) = \alpha$ for all $\sigma \in \mathcal{S}_F$, but is not necessarily zero for non-solutions. Given an algorithm $\mathcal{A}$ that produces samples from P_T, we can construct an algorithm $\mathcal{B}$ that samples from D (i.e. uniformly from the solution set) using rejection sampling. More specifically, we take samples s produced by $\mathcal{A}$ and we discard all the ones such that $s \notin \mathcal{S}_F$. The fundamental tradeoff involved is that we want the probability distribution P_T to be easier to sample from (compared to D), but

at the same time the probability mass should be concentrated on satisfying assignments (i.e. P_T should be close enough to D) so that we don't generate too many unwanted samples (i.e. non-solutions). Closely related to Simulated Annealing (SA) is the Gibbs Sampler for the Boltzmann distribution, that defines a similar Markov Chain with the same steady state probability distribution [2]. In our analysis below, we consider a fixed temperature annealing where T is fixed during the run of the Markov Chain.

3.2 SampleSAT

In [8] Wei et al. propose the use of an hybrid approach where they interleave SA moves with moves based on a focused random walk procedure (inspired by local search SAT solvers [15]). This approach is based on a result by Papadimitriou [16] proving that a random walk procedure for SAT will find a solution to any satisfiable 2CNF formula in $O(n^2)$ time, where n is the number of variables in the formula. While the role of focused random walk moves is to find solution "clusters", the role of SA moves is to heuristically provide some level of uniformity of the sampling. The drawback of this approach is that it does not maintain any detailed balance equation so there is no control on the resulting steady state probability distribution. SampleSAT therefore loses all the theoretical guarantees on the uniformity of the sampling provided by SA, and as we show below in the experimental section, it can lead to poor uniformity also in practice.

3.3 SampleSearch

SampleSearch [11, 12, 13] is an importance sampling technique that draws samples from the so-called *backtrack-free distribution*. Although it is not a uniform sampler, these samples can be used to compute expectations (using importance sampling) by correcting for the non uniformity of the *backtrack-free distribution*. The support of the *backtrack-free distribution* corresponds exactly to the set of solutions, and this greatly reduces the number of rejected samples. However, as other importance sampling schemes, the performance is highly dependent on the choice of the proposal distribution, which is usually precomputed using a generalized belief propagation scheme [11]. Sampling from the *backtrack-free distribution* can be achieved either by using a complete solver as a black-box, or by using the SampleSearch scheme, which integrates backtracking search with sampling [12]. Our approach is similar in the sense that we also use a complete solver as an oracle as we explore the search tree. However, the search tree is explored is a very different way. Specifically, our *recursive* approach aims at uniformly exploring of the search tree level by level, and directly provides (approximately) uniform samples without need for a heuristically chosen proposal distribution (e.g., using variational methods). We will compare the performance of the two methods for model counting below.

4 Black-Box Sampling

Modern day SAT solvers are very effective at finding *a solution*, but they explore the search space in a highly non-uniform way. This is because they use heuristics such assigning "Pure literals" (e.g., if a variable only appears with positive sign in a formula, then it can be safely set to true) that heavily bias the search towards certain parts of the search space.

Modifying a pure DPLL-style algorithm to produce uniform samples is a challenging task. Suppose the choice of the ordering in which variables are set is chosen uniformly at random, and the polarity (whether to assign true or false to a variable during search) is also chosen uniformly at random. Then in general the first solution found is not a uniform sample from S_F. A simple counterexample is the formula $x_1 \vee x_2$, where it can be seen that the solution $x_1 = x_2 = 1$ is less likely to be found in both possible variables orderings. In order to obtain uniform samples, one could choose the polarity of the variables according to their marginal probabilities (with respect to D), but this would defy the purpose of sampling, since we often want to obtain samples precisely to estimate quantities such as marginals.

Instead of modifying an existing search procedure to produce uniform samples, we introduce a novel recursive sampling scheme that aims at enforcing a level by level uniform exploration of the search tree, while leveraging the reasoning power of a complete SAT solver.

4.1 A Recursive Sampling Strategy

Our method is based on the notion of *pseudosolution*, defined as follows:

Definition 1. *Let π be an ordering of the variables. A* pseudosolution *of level i is a truth assignment to the first i variables that can be completed to form a solution (i.e. a node in the search tree at level i that has a solution as a descendant). We denote S_i the set of* pseudosolutions *of level i.*

Our recursive strategy (Algorithm 1) is based on the idea of dividing the search tree into L levels, and recursively sampling *pseudosolutions* using previously generated samples of *pseudosolutions* of a higher level (see Figure 1). In other words, we assume to have access to uniform samples of ancestors of solutions at level i, and we generate samples of ancestors of solutions at level $i + \ell$ using Algorithm 2. The procedure is initialized with a pseudosolution of level 0, i.e. an empty variable assignment. Note that generating samples from *pseudosolutions* of level n (if there are n variables) is equivalent to the original problem of sampling solutions from S_F.

A complete SAT solver is used in Algorithm 2 to generate

Algorithm 1 SearchTreeSampler(F, k, M)

Input: Formula F with n vars. Parameters k, $M = 2^\ell$
Output: A set S of solutions of F
if F is not satisfiable **then**
  return $\emptyset$
else
  Let $\Phi_0 = \{\top\}$ // True, empty variable assignment
  Let $L = \lceil \frac{n}{\ell} \rceil$ // number of levels
  for $i = 1, \cdots, L$ **do**
    $\Phi_i \leftarrow$ BlackBoxSampler(Φ_{i-1}, k, ℓ)
  end for
  return Φ_L
end if

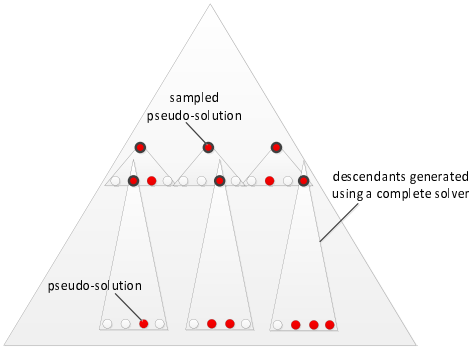


Figure 1: Representation of how Algorithm 1 explores the search tree.

$D(s_j)$, the set of all pseudosolutions of level $i + \ell$ with s_j as ancestor. This is accomplished by repeatedly checking the satisfiability of $F \wedge s_j \bigwedge_{d \in D} \neg d$ until it is provably unsatisfiable, adding a *pseudosolution* d to D each time one is found. Completeness is required in order to enumerate all elements of $D(s_j)$, i.e. to prove unsatisfiability when all elements have been found.

4.2 Analysis

It is easy to verify by induction that the sets Φ_i in Algorithm 1 satisfy the property $|\Phi_i| \geq \min\{k, |S_i|\}$. The key property of Algorithm 2 is that the set S returned is such that sampling from S is approximately equivalent to sampling from $S_{i+\ell}$, the set of *pseudosolutions* of level $i + \ell$. The parameter $k > 1$ controls the uniformity of the sampling. This is formalized by the following Theorem:

Theorem 1. *Let S be the output of Algorithm 2 with input Φ such that $|\Phi| \geq \min\{k, |S_i|\}$, and $s, s' \in S_{i+\ell}$ be any two pseudosolutions of level $i + \ell$. We have*

$$\frac{k}{M + k - 1} \leq \frac{P(s)}{P(s')} \leq \frac{M + k - 1}{k}, \qquad (1)$$

where $M = 2^\ell$ and $P(s)$ is the probability that a uniformly sampled element from S is equal to s.

Algorithm 2 BlackBoxSampler(Φ,k,ℓ)

Input: A set Φ of uniformly sampled pseudosolutions of level i. Parameters k, ℓ
Output: A set S of pseudosolutions of level $i + \ell$ (approximately uniformly sampled) with $2^\ell |\Phi| \geq |S| \geq |\Phi|$
$S = \emptyset$
for $j = 1, \cdots, \min\{k, |\Phi|\}$ **do**
  Sample s_j from Φ without replacement
  Generate $D(s_j)$, the set of all pseudosolutions of level $i+\ell$ with s_j as ancestor (using a complete SAT solver)
  $S = S \cup D(s_j)$
end for
return S

Proof. Suppose $k \leq |\Phi|$ (otherwise, it means that Φ contains all pseudosolutions of level i, hence by definition $S = S_{i+\ell}$ and therefore $P(s) = P(s')$), and that $\Phi \subseteq S_i$ is a set of uniformly sampled pseudosolutions of level i. We can think of each pseudosolution $s \in \Phi$ as an urn, that contains a certain number $1 \leq |D(s)| \leq M = 2^\ell$ of pseudosolutions at a lower level $i + \ell$ (its descendants). Let $|S_i| = N \geq |\Phi|$ be the total number of pseudosolutions of level i.

Since Φ contains uniform samples, $s_1, \cdots, s_k$ in Algorithm 2 are also uniform samples of *pseudosolutions* of level i. Let S as in Algorithm 2 (the union of the contents of the k urns selected). Let $s \in S_{i+\ell}$ be a pseudosolution of level $i + \ell$, and let $a(s) \in S_i$ be its unique ancestor at level i. Clearly, the probability that $s \in S$ is $P[s \in S] = \frac{\binom{N-1}{k-1}}{\binom{N}{k}}$ that is equal to the probability of selecting the ancestor $a(s)$ of s on level i. Let e be a randomly selected element of S. Ideally, we would like the probability $P(s) \triangleq P[e = s]$ to be a constant independent of s (uniform sampling). However, intuitively this is not exactly constant because there is a bias towards elements such that $|D(a(s))|$ is small. Specifically,

$$P(s) = \frac{1}{\binom{N}{k}} \sum_{\ell_1 \cdots \ell_{k-1}} \frac{1}{|D(a(s)) \cup D(s_{\ell_1}) \cdots \cup D(s_{\ell_{k-1}})|}.$$

Since the sets are disjoint,

$$P[e = s] = P(s) = \frac{\binom{N-1}{k-1}}{\binom{N}{k}} \sum_{t=1}^{\binom{N-1}{k-1}} \frac{1}{\binom{N-1}{k-1}} \frac{1}{|D(a(s))| + z_t},$$

where $z_t = |D(s_{\ell_1}) \cup \cdots \cup D(s_{\ell_{k-1}})| \geq k - 1$. Let $s' \in S_{i+\ell}$ be another pseudosolution of level $i + \ell$. We rewrite $P(s)$ as

$$\frac{1}{\binom{N}{k}} \left(\sum_{t=1}^{\binom{N-2}{k-2}} \frac{1}{|D(a(s))| + a_t + |D(a(s'))|} + \sum_{t=1}^{\binom{N-2}{k-1}} \frac{1}{|D(a(s))| + b_t} \right).$$

Suppose wlog that $|D(a(s))| \geq |D(a(s'))|$. Then,

$$\frac{1}{|D(a(s'))| + b_t} \leq \frac{|D(a(s))| + k - 1}{|D(a(s'))| + k - 1} \frac{1}{|D(a(s))| + b_t}$$

by Lemma 1, and using the fact that $1 \leq |D(a(s))| \leq M$,

$$\frac{1}{|D(a(s'))| + b_t} \leq \frac{M + k - 1}{k} \frac{1}{|D(a(s))| + b_t}.$$

Hence,

$$\sum_{t=1}^{\binom{N-2}{k-1}} \frac{1}{|D(a(s'))| + b_t} \leq \frac{M + k - 1}{k} \sum_{t=1}^{\binom{N-2}{k-1}} \frac{1}{|D(a(s))| + b_t}$$

and finally,

$$\left(\sum_{t=1}^{\binom{N-2}{k-2}} \frac{1}{|D(a(s))| + a_t + |D(a(s'))|} + \sum_{t=1}^{\binom{N-2}{k-1}} \frac{1}{|D(a(s'))| + b_t} \right) \leq$$

$$\frac{M + k - 1}{k} \left(\sum_{t=1}^{\binom{N-2}{k-2}} \frac{1}{|D(a(s))| + a_t + |D(a(s'))|} + \sum_{t=1}^{\binom{N-2}{k-1}} \frac{1}{|D(a(s))| + b_t} \right)$$

that gives

$$\frac{P(s')}{P(s)} \leq \frac{M + k - 1}{k}.$$

□

Lemma 1. *Given $d \geq c$ and a finite sequence $\{b_j\}_1^T$ such that $0 < k - 1 \leq b_j$ for all j, we have*

$$\frac{\sum_j \frac{1}{c + b_j}}{\sum_j \frac{1}{d + b_j}} \leq \frac{d + k - 1}{c + k - 1}.$$

Proof. Notice that for any j we have $\frac{\frac{1}{c+b_j}}{\frac{1}{d+b_j}} \leq \frac{1/(c+k-1)}{1/(d+k-1)}$ because when $c \leq d$ it is a monotonically decreasing function of b_j. The desired result follows by summing up both sides over j and dividing. □

Equation (1) shows that the sampling becomes more uniform as $k \to \infty$, and allows us to bound the uniformity of the output of Algorithm 2 as a function of k (a pseudosolution cannot be much more likely than another one). For instance, when $M = 2$ and $k = 100$, a pseudosolution cannot be more than 1% more likely to be sampled than another one. Notice that larger values of k would both improve the uniformity of the sampling and at the same time increase the number of output samples (sampling is without replacement). However, larger values of k would also require more calls to the SAT solver.

Remark: When Algorithm 2 is used recursively as in Algorithm 1, the samples received as input are usually not truly uniform unless k is larger than the total number of solutions[2]. Even though in general the sets Φ_i in Algorithm 1 do not meet the assumptions of Theorem 1, they tend to satisfy them as $k \to \infty$. The effect of rather small, practical values of k is investigated empirically below, where we evaluate the statistical properties of the output samples Φ_L.

[2]Uniformity is also guaranteed for the first b levels, until $|S_b| < k$.

4.3 Complexity

If n is the number of variables of the input formula F, Algorithm 1 requires $O(\lceil \frac{n}{\log M} \rceil Mk)$ calls to the SAT solver to get at least $\min\{\#\text{solutions of F}, k\}$ samples. Larger values M require more calls to the SAT solver, but intuitively also improve the uniformity of the sampling by reducing the number of recursions. In particular, in the extreme (impractical) case $M = 2^n$ we obtain truly uniform sampling because it corresponds to exact model counting (explicitly enumerating all solutions). Notice also that although in our experiments we use constant values for k and M, they could be chosen as a function of the level i.

5 Evaluating Sampling Methods

It can be verified that Simulated Annealing (SA) is ergodic for all "temperatures" $T > 0$ when new configurations are generated by randomly flipping a variable chosen uniformly at random (so that the chains are irreducible [2]). Since according to the steady state probability distribution all satisfying assignments have the same probability, in principle SA will provide uniform samples if we are willing to wait until the stationary distribution is reached. However, the amount of time we have to wait until the chain reaches its steady state and between two consecutive samples is of paramount importance for any practical application. On the other hand, SampleSAT is often much faster at finding solutions because of the focused random walk component, but it does not provide any guarantee on the uniformity of the sampling. To evaluate the practical utility of these methods it is therefore important to quantitatively measure the uniformity of the samples provided after finite amounts of time.

The experiments are run on a set of formulas F for which we know (analytically or using exact model counters [17]) that the number of distinct solutions $|S_F|$ is relatively small (≤ 1000). Therefore, by taking a sufficiently large number of samples, we are able to check the uniformity of the methods. In particular, let N_i be the number of times the ith solution has been sampled, for $i = 1, \ldots, |S_F|$. Given P samples from a truly uniform solution sampler, we expect N_i to be close to $P/|S_F|$ (since for the law of large numbers N_i/P converges to $1/|S_F|$ in probability as $P \to \infty$). To quantitatively measure the uniformity of the sampler, we use the Pearson's χ^2 statistic defined as

$$\chi^2 = \frac{1}{P/|S_F|} \sum_i (N_i - P/|S_F|)^2,$$

where $P/|S_F|$ is the expected theoretical frequency under

the hypothesis that the distribution is truly uniform. The χ^2 value can be used to test a null hypothesis stating that the frequencies N_i observed are consistent with a uniform distribution. In particular, the null hypothesis is rejected if the χ^2 value is larger than a cutoff value that depends on the number of distinct solutions $|S_F|$ (specifying the number of degrees of freedom of the distribution) and on the statistical significance desired (e.g., 0.05).

For our experiments, we use MiniSAT 2.2 [18] as a complete solver in Algorithm 1. Pseudosolutions are defined according to lexicographical variable ordering π_{LX}. For efficiency, all the experiments are run with $M = 2$. When we wish to obtain P samples, we keep running `SearchTreeSampler` with a parameter $k < |S_F| < P$ until we obtain at least P samples (taking exactly k samples without replacement from S_F per run), and we report the total running time. Note that choosing $k \geq |S_F|$ (e.g., $k = P$) wouldn't let us evaluate the performance of the method as an approximately uniform sampler, because it would correspond to enumerating all solutions (hence perfectly uniform sampling).

SA is evaluated as follows. We run the chain from a random initial truth assignment, discarding all the samples for the first 10^7 steps (burn-in phase). After the burn-in phase, we assume that the Markov Chain has reached its steady state distribution and we start taking samples. Every K steps, the current truth assignment σ_t is taken as a sample from the steady state distribution. We wait for K steps to ensure that consecutive samples are sufficiently independent. If the sample σ_t is a solution ($\sigma_t \in S_F$), we output it, otherwise we discard it. This process is carried out until we find a prescribed number of solutions P (non necessarily all distinct). SampleSAT is executed with default parameters, and all methods are run on the same 3GHz machine with 4Gb of memory.

6 Challenging Sample Spaces

6.1 Golf-Course energy landscape

We first consider a class of instances from [8] which we call $plateau(b)$, defined as

$$(x_1 \vee y_1) \wedge (x_1 \vee y_2) \wedge \cdots \wedge (x_1 \vee y_b) \wedge (\neg x_1 \vee z_1) \wedge (\neg x_1 \vee \neg z_1).$$

This class of instances is hard for local search methods such as SA because it has a very large plateau of truth assignments (of size at least 2^{b+1}) with energy $E(\sigma) = 1$, corresponding to assignments with $x_1 = 1$. The effect of the last two clauses is to enforce $x_1 = 0$, so that there are only two distinct solutions. Intuitively, this instance is difficult for SA because in order to reach one of the solutions, SA needs to set $x_1 = 0$. Setting $x_1 = 0$ is likely to violate several of the first b clauses, making the uphill move unlikely to be accepted. Formally, in [8] it is shown (Theorem 1) that SA at fixed temperature with probability going to 1 takes time exponential (in b) to find a satisfying assignment.

As shown in [8], SampleSAT is not affected by energy plateaus because it uses focused moves to quickly reach solutions. Since it is not based on local information, `SearchTreeSampler` is not affected by energy plateaus. In fact, since $plateau(b)$ instances belong to 2-CNF (formulas with at most 2 variables per clause) and has two solutions, we can use the following result:

Theorem 2. *Algorithm 1 provides uniform samples in polynomial time for 2-CNF instances F such that $S_F \leq k$.*

Proof. Follows from the fact that 2-SAT is solvable in polynomial time. Uniformity follows from the fact that Algorithm 1 will output all solutions when $S_F \leq k$. □

6.2 Energy barriers

We define an energy barrier between two variable assignments $\sigma, \sigma' \in \{0,1\}^n$ as the minimum over the set of all possible paths on the Boolean hypercube $\{0,1\}^n$ between σ and σ' of the maximum energy of the configurations in each path. Simulated Annealing is designed to deal with energy barriers (such as the ones encountered around a local minimum) by allowing occasional uphill moves (stochastic hill climbing), i.e. moves that lead to an increase of the energy function. In particular, larger values of the temperature parameter T make the acceptance of uphill moves more likely.

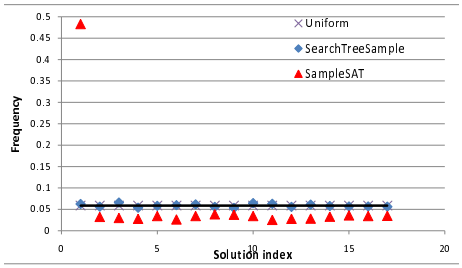
In this section, we will show that SA can only deal with relatively low energy barriers if we want to maintain a reasonable efficiency in the rejection sampling scheme. Consider the following CNF formula that we call $XORBarrier(b)$:

$$(x_1 \Rightarrow y_1) \wedge (x_1 \Rightarrow y_2) \wedge \cdots \wedge (x_1 \Rightarrow y_b) \wedge (\neg x_1 \Rightarrow \neg y_1) \wedge (\neg x_1 \Rightarrow \neg y_2) \wedge \cdots \wedge (\neg x_1 \Rightarrow \neg y_b).$$

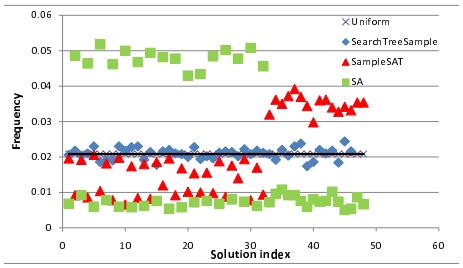
We use the term $XORBarrier(b)$ because it is equivalent to a conjunction of XOR constraints of the form $\neg(x_1\ XOR\ y_1) \wedge \cdots \wedge \neg(x_1\ XOR\ y_b)$. It is easy to see that for any value of b, these instances have only two solutions $(x_1, y_1, \cdots, y_b) = (0 \ldots 0)$ and $(x_1, y_1, \cdots, y_b) = (1 \cdots 1)$. It can be seen that $d(i) = |\{\sigma \in \{0,1\}^n | E(\sigma) = i\}| = 2\binom{b}{i}$. For any value of the temperature parameter T, the steady state probability of having SA in a configuration with energy i is

$$\mathbb{P}[E(\sigma) = i] = \sum_{\sigma | E(\sigma) = i} P_T(\sigma) \propto n(i) e^{-\frac{i}{T}} = 2\binom{b}{i} e^{-\frac{i}{T}}.$$

The key feature of this type of instances is the following. Since Simulated Annealing (and Gibbs Sampling) flips only one variable at a time, it is easy to see that any path in the Markov Chain graph that brings from one solution to



(a) Asymmetric space with energy barrier.



(b) Random formula with energy barrier.

Figure 2: Uniformity of sampling. Frequency each solution is sampled by different methods.

the other one will contain a configuration that violates $b/2$ constraints. Therefore we conclude that there is an energy barrier of "height" $b/2$ between the two solutions.

For large values of b, one cannot find a temperature T that provides both high enough $P_T(0)$ (i.e. small number of flips per solution) and $P_T(b/2)$ (i.e. probability of "jumping" over the barrier) to obtain a practical sampler. For instance, for $b = 80$, to get $P_T(40) > 10^{-9}$ (i.e. climbing the barrier on average once in about a billion samples) SA needs a temperature $T > 0.75$, but $P_{0.75}(0) = 7.4 \times 10^{-9}$ (which means that on average we need more than 10^8 samples to get a single solution). Similar results are obtained when using Gibbs sampling. This is because a Gibbs sampler has the same Boltzmann steady state probability distribution and it also proceeds by flipping a single variable at a time, so it cannot "jump" over the barrier with a single move.

Since instances in $XORBarrier(b)$ belong to 2-SAT and have 2 solutions, from Theorem 2 Algorithm 1 provides uniform samples in polynomial time.

A common strategy to partially overcome the ergodicity problems of Gibbs sampling and Simulated Annealing is the use of multiple parallel chains [9] or restarts [8]. These approaches can be quite effective and, for instance, are capable of producing uniform samples for the $XORBarrier(b)$ class of instances presented above. Intuitively, this is because on average 50% of the random initial assignments will be on one side of the barrier (i.e., the half-hypercube with $x_1 = 0$) and the other 50% on the other side (i.e., $x_1 = 1$).

6.3 Asymmetric spaces with energy barriers

These approaches are however not sufficient to sample from distributions where there are "solution clusters" (i.e., groups of solutions that are close in Hamming distance) of different sizes that are separated by energy barriers. Consider for instance the following class of instances that we call $AsymXORbarrier(b, \ell)$:

$$XORBarrier(b) \wedge (x_1 \vee z_1) \wedge (x_1 \vee z_2) \wedge \cdots \wedge (x_1 \vee z_\ell)\,.$$

The effect of the additional clauses $(x_1 \vee z_i)$ is to create a cluster of 2^ℓ solutions on one side of the barrier ($x_1 = 1$), while there exist only one solution with $x_1 = 0$.

As shown in Figure 2a and Table 1 (bottom rows), `SearchTreeSampler` provides a much more uniform sampling compared to SampleSAT on this type of instances. Specifically, the uniformity of sampling hypothesis is rejected for SampleSAT according to the χ^2-test. This is because SampleSAT employs restarts and therefore samples the first solution in Figure 2a (corresponding to the isolated solution with all variables set to 0) too often, i.e. about 50% of the times. Similarly biased results are obtained using a Gibbs Sampler with multiple parallel chains (as implemented in the Alchemy system [9]), while plain SA or Gibbs is not able to cross the barrier in a reasonable amount of time. Further, we see from Table 1 (4th column) that thanks to MiniSAT's reasoning power, `SearchTreeSampler` is about 3 orders of magnitude faster than SampleSAT.

6.4 Embedded energy barriers

In Table 1 (top rows), we evaluate the performance of the methods considered on three types of instances: a logistic one generated by SATPlan [19] (*logistic*, with 110 variables, 461 clauses, and 512 solutions), a graph Coloring problem from SatLib [20](*coloring*, with 90 variables, 300 clauses, and 900 solutions), and a random 3-SAT formula (*random*, with 75 variables, 315 clauses, and 48 solutions). We collect respectively $P = 50000$, $P = 200000$, and $P = 5000$ samples for each instance. We also artificially introduce an energy barrier in each of these instances by choosing a variable z such that there are roughly half solutions with $z = 1$ and half solutions with $z = 0$ in the original formula. We then introduce a new set clauses defined by $XORbarrier(40)$ where we substitute z to x_1 and $y_1, \ldots, y_{40}$ are fresh variables. Note that this does not change the number of solutions.

We experimented with several values of k and T and we provide a summary of the best results obtained in Table 1, both for the original formulas and the ones with an embedded energy barrier. By carefully choosing the tempera-

Method	Instance	Parameter	Time (s)	χ^2	P-value
SA	logistic	T=0.25	11028	550	**0.11**
SampleSAT	logistic	-	7598	534.9	**0.23**
SearchTreeSampler	logistic	k=5	9.8	545.82	**0.14**
SearchTreeSampler	logistic	k=20	10.1	487.43	**0.77**
SearchTreeSampler	logistic	k=50	9.6	407.01	**0.99**
SA	logistic+Barrier	T=0.25	42845	50495.8	0
SampleSAT	logistic+Barrier	-	7296	178860.2	0
SearchTreeSampler	logistic+Barrier	k=20	42.53	469.23	**0.90**
SA	coloring	T=0.25	7589	875.131	**0.71**
SampleSAT	coloring	-	28998	132559	0
SearchTreeSampler	coloring	k=50	184	844	**0.90**
SearchTreeSampler	coloring	k=100	204	808.3	**0.98**
SearchTreeSampler	coloring	k=200	228	672.15	**1.00**
SA	coloring+Barrier	T=0.25	29434	100905	0
SampleSAT	coloring+Barrier	-	29062	141027	0
SearchTreeSampler	coloring+Barrier	k=100	435	746.40	**0.99**
SA	random	T=0.3	8673	36.62	**0.88**
SampleSAT	random	-	740	970.21	0
SearchTreeSampler	random	k=10	2.5	76.84	0
SearchTreeSampler	random	k=15	3	31.28	**0.96**
SearchTreeSampler	random	k=20	5	29.26	**0.98**
SA	random+Barrier	T=0.3	71077	4211.40	0
SampleSAT	random+Barrier	-	744	1104.9	0
SearchTreeSampler	random+Barrier	k=10	7.0	64.55	0.04
SearchTreeSampler	random+Barrier	k=15	7.1	32.64	**0.94**
SearchTreeSampler	random+Barrier	k=20	7.1	32.33	**0.95**
SampleSAT	AsymXORBarrier(80,4)	-	290	6508	0
SearchTreeSampler	AsymXORBarrier(80,4)	k=2	0.7	51.54	0
SearchTreeSampler	AsymXORBarrier(80,4)	k=5	0.4	17.35	**0.36**
SearchTreeSampler	AsymXORBarrier(80,4)	k=10	0.3	7.84	**0.95**
SampleSAT	AsymXORBarrier(80,8)	-	4260	1893391	0
SearchTreeSampler	AsymXORBarrier(80,8)	k=25	1.9	220.14	**0.94**
SearchTreeSampler	AsymXORBarrier(80,8)	k=50	1.6	197.17	**0.99**

Table 1: Uniformity of sampling for instances with and without embedded energy barriers. P-value is the probability of observing an event at least as extreme under the null hypothesis (uniform sampling). In bold null hypothesis is not rejected.

ture parameter T, SA can provide uniform samples for the original formulas. However, it is very slow compared to the other methods (smaller T would make it faster, but the samples would not be uniform). On the other hand, SampleSAT is generally much faster, but the samples provided are far less uniform. SearchTreeSampler is superior to both SA and SampleSAT, both in terms of running time (at least 2 orders of magnitude faster) and uniformity. As expected, increasing k improves the uniformity and it does not affect the running time because it also produces more samples per run.

As expected, the introduction of energy barriers severely limits the ergodicity of SA, as shown in the last column of Table 1 and in Figure 2. As a result, both SA and SampleSAT fail to provide uniform samples (according to the χ^2 test) for the instances with embedded energy barriers. On the other hand, the uniformity of SearchTreeSampler is not affected, although the runtime increases (because the search tree is deeper with the introduced fresh variables).

7 A New Model Counting Technique

A typical way [21, 22] to estimate the number of solutions $|S_F| = Z$ (model count) of a formula F using a sample approximation is to estimate a series of multipliers

$$\frac{Z}{Z(x_1 = a_1)} \frac{Z(x_1 = a_1)}{Z(x_1 = a, x_2 = a_2)} \cdots \frac{Z(x_1 = a_1, \cdots, x_{n-1} = a_{n-1})}{1}.$$

For instance, $\frac{Z(x_1=a_1)}{Z}$ is given by the fraction of solutions that have $x_1 = a_1$ (the number of solutions in the green vs. yellow part of the search tree in Figure 3), and so on. This can be a challenging task because one has to heuristically select a "good" variable ordering and polarities $a_1, \cdots, a_n$ to condition on, and solve multiple sampling tasks with some of the variables clamped [22].

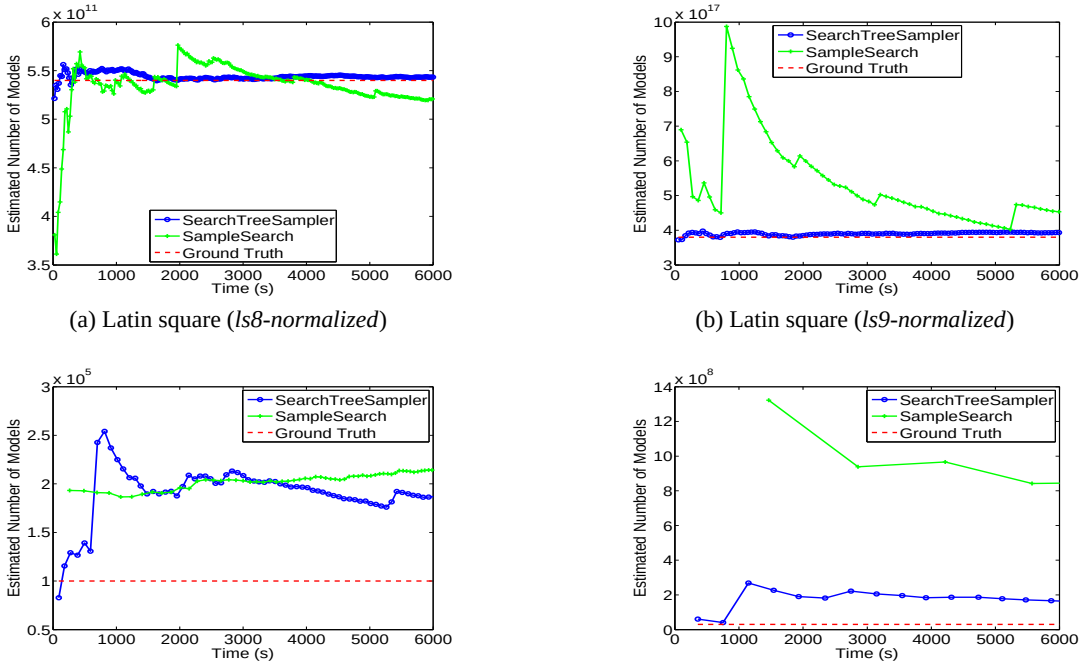
Algorithm 1 provides a new direct way of estimating the number of solutions. Let S_i be the set of pseudosolutions of level i. Suppose $Z = |S_n| > 0$ (the formula is satisfiable), and for simplicity let $M = 2^\ell = 2$. Then, we have

$$Z = |S_n| = \frac{|S_n|}{|S_{n-1}|} \frac{|S_{n-1}|}{|S_{n-2}|} \frac{|S_{n-2}|}{|S_{n-3}|} \cdots \frac{|S_1|}{1}$$

(see right panel of Figure 3), where we can estimate each multiplier using the relation

$$\frac{|S_i|}{|S_{i-1}|} = \frac{1}{|S_{i-1}|} \sum_{s \in S_{i-1}} |D(s)| \approx \frac{1}{|\Phi_{i-1}|} \sum_{\widetilde{s} \in \Phi_{i-1}} |D(\widetilde{s})|,$$

where elements $\widetilde{s} \in \Phi_{i-1}$ provided by Algorithm 1 are (approximately) uniformly sampled elements of S_{i-1}. Note that each multiplier corresponds to the average number of descendants at a given level i of the search tree, and we can estimate all of them in a single run of SearchTreeSampler (i.e. no need to solve multiple sampling tasks as in [22]).



(a) Latin square (*ls8-normalized*)

(b) Latin square (*ls9-normalized*)

(c) Langoford problem (*lang-12*)

(d) Langford problem (*lang-15*)

Figure 4: Estimated model count as a function of runtime.

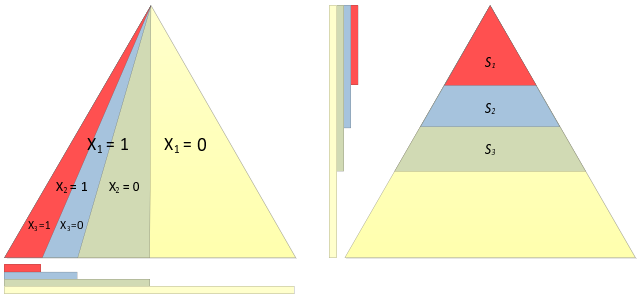


Figure 3: Pictorial representation of the traditional (left) and new model counting techniques (right). Notice the vertical vs. horizontal division of the search tree.

7.1 Experimental Results

We compare our new counting method with SampleSearch [23], currently the best model counter based on (importance) sampling [12], on several large instances (with known ground truth) from a standard model counting benchmark [3]. In Figure 4 we plot the estimated model count over time (as more samples are being collected) for the two methods, for 4 representative instances (encodings of Latin Squares and Langford's problems). Note that both methods rely on MiniSAT as internal constraint solver. Both methods are run on the same 3GHz machine, `SearchTreeSampler` with $k = 500$ and SampleSearch with parameters as in [23].

Our empirical results show that `SearchTreeSampler` converges faster and provides more accurate estimates. Similar results are obtained for other instances in the benchmark (not reported for space reasons). Note that these instances have a rather large number of solutions so it is difficult to evaluate the uniformity of the sampling using a χ^2 test. In this case, the accuracy of the estimates of Z is an indication of the of the quality of the sampling scheme.

8 Conclusions

We presented `SearchTreeSampler`, a new method to sample solutions of Boolean formulas and to estimate their count. Our method uses a constraint solver as a black-box, and can therefore leverage the reasoning power of state-of-the-art constraint solving technology, while maintaining an approximately uniform exploration of the search tree. We presented several challenging domains, such as energy barriers and asymmetric search spaces, that reveal the difficulties of sampling. We presented data demonstrating that standard methods such as Simulated Annealing, Gibbs Sampling, and SampleSAT seriously suffer from these difficulties in terms of runtime and uniformity of the sampling. Our results also demonstrate that `SearchTreeSampler` can overcome many of these difficulties and is considerably faster on a set of benchmark problems. Further, we show that the intermediate samples of pseudosolutions provided by our method can be used in a new model counting technique, which is shown to be very effective in practice.

References

[1] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.

[2] N.N. Madras. *Lectures on Monte Carlo Methods*. American Mathematical Society, 2002.

[3] C.P. Gomes, J. Hoffmann, A. Sabharwal, and B. Selman. From sampling to model counting. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[4] L.G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410, 1979.

[5] C.P. Gomes, A. Sabharwal, and B. Selman. Near-uniform sampling of combinatorial spaces using xor constraints. *Advances In Neural Information Processing Systems*, 19:481, 2007.

[6] R. Dechter, K. Kask, E. Bin, and R. Emek. Generating random solutions for constraint satisfaction problems. In *Proc. 18th National Conference on Artificial Intelligence (AAAI)*, pages 15–21, 2002.

[7] V. Gogate and R. Dechter. Approximate solution sampling (and counting) on and/or spaces. In *Principles and Practice of Constraint Programming*, pages 534–538. Springer, 2008.

[8] W. Wei, J. Erenrich, and B. Selman. Towards efficient sampling: exploiting random walk strategies. In *Proc. 19th National Conference on Artifical Intelligence (AAAI)*, pages 670–676, 2004.

[9] H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proc. 21st National Conference on Artifical Intelligence (AAAI)*, volume 21, page 458, 2006.

[10] H. Kautz and B. Selman. The state of sat. *Discrete Applied Mathematics*, 155(12):1514–1524, 2007.

[11] V. Gogate and R. Dechter. Samplesearch: A scheme that searches for consistent samples. In *Proc. 10th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[12] V. Gogate and R. Dechter. Samplesearch: Importance sampling in presence of determinism. *Artificial Intelligence*, 175(2):694–729, 2011.

[13] V. Gogate and R. Dechter. Studies in solution sampling. In *Proc 23rd National Conference on Artificial Intelligence (AAAI)*, volume 1, page 271. AAAI Press, 2008.

[14] N. Metropolis, AW Rosenbluth, MN Rosenbluth, AH Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1091, 1953.

[15] B. Selman, H.A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1996.

[16] C.H. Papadimitriou. On selecting a satisfying truth assignment. In *Proc. 32nd Annual Symposium on Foundations of Computer Science*, pages 163–169, 1991.

[17] T. Sang, F. Bacchus, P. Beame, H. Kautz, and T. Pitassi. Combining component caching and clause learning for effective model counting. In *Theory and Applications of Satisfiability Testing (SAT)*, 2004.

[18] N. Eén and N. Sörensson. An extensible sat-solver. In *Theory and Applications of Satisfiability Testing (SAT)*, pages 333–336. Springer, 2004.

[19] H. Kautz and B. Selman. Planning as satisfiability. In *Proc. 10th European Conference on Artificial Intelligence (ECAI)*, pages 359–363, 1992.

[20] H.H. Hoos and T. Stiitzle. SATLlB: An Online Resource for Research on SAT. *Sat2000: highlights of satisfiability research in the year 2000*, page 283, 2000.

[21] M. Jerrum and A. Sinclair. The Markov chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1997.

[22] W. Wei and B. Selman. A new approach to model counting. In *Theory and Applications of Satisfiability Testing (SAT)*, pages 324–339, 2005.

[23] V. Gogate and R. Dechter. Approximate counting by sampling the backtrack-free search space. In *Proc. 22nd National Conference on Artifical Intelligence (AAAI)*, volume 22, page 198, 2007.

Spectral Estimation of Conditional Random Graph Models for Large-Scale Network Data

Antonino Freno **Mikaela Keller*** **Gemma C. Garriga** **Marc Tommasi***
INRIA Lille – Nord Europe, Villeneuve d'Ascq (France)
{antonino.freno, mikaela.keller, gemma.garriga, marc.tommasi}@inria.fr

Abstract

Generative models for graphs have been typically committed to strong prior assumptions concerning the form of the modeled distributions. Moreover, the vast majority of currently available models are either only suitable for characterizing some particular network properties (such as degree distribution or clustering coefficient), or they are aimed at estimating joint probability distributions, which is often intractable in large-scale networks. In this paper, we first propose a novel network statistic, based on the Laplacian spectrum of graphs, which allows to dispense with any parametric assumption concerning the modeled network properties. Second, we use the defined statistic to develop the *Fiedler random graph* model, switching the focus from the estimation of joint probability distributions to a more tractable conditional estimation setting. After analyzing the dependence structure characterizing Fiedler random graphs, we evaluate them experimentally in edge prediction over several real-world networks, showing that they allow to reach a much higher prediction accuracy than various alternative statistical models.

1 INTRODUCTION

Arising from domains as diverse as bioinformatics and web mining, large-scale data exhibiting network structure are becoming increasingly available. Network models are commonly used to represent the relations among data units and their structural interactions. Recent studies, especially targeted at social network modeling, have focused on random graph models of those networks. In the simplest form, a social network is a configuration of binary random variables X_{uv} such that the value of X_{uv} stands for the presence or absence of a link between nodes u and v in the network. The general idea underlying random graph modeling is that network configurations are generated by a stochastic process governed by specific probability laws, so that different models correspond to different families of distributions over graphs.

The simplest random graph model is the Erdős-Rényi (ER) model [Erdős and Rényi, 1959], which assumes that the probability of observing a link between two nodes in a given graph is constant for any pair of nodes in that graph, and it is independent of which other edges are being observed. In preferential attachment models [Barabási and Albert, 1999], the probability of linking to any specified node is proportional to the degree of the node in the graph, leading to "rich get richer" effects. Small-world models [Watts and Strogatz, 1998] try to capture instead some phenomena often observed in real networks such as high clustering coefficients and small diameters [Newman, 2010]. A sophisticated attempt to model complex dependences between edges in the form of Gibbs-Boltzmann distributions is made by exponential random graph (ERG) models [Snijders et al., 2006], which subsume the ER model as a special case. Finally, a recent attempt at modeling real networks through a stochastic generative process is made by Kronecker graphs [Leskovec et al., 2010], which try to capture phenomena such as heavy-tailed degree distributions and shrinking diameter properties while paying attention to the temporal dynamics of network growth.

While some of these models behave better than others in terms of computational tractability, one basic limitation affecting all of them is what we may call a *parametric assumption* concerning the probability laws underlying the observed network properties. In other words, currently available models of network structure assume that the probability distribution gener-

*Université Charles de Gaulle – Lille 3, Villeneuve d'Ascq (France).

ating the network can be expressed through a particular parametric form $P(\mathcal{G}|\,\boldsymbol{\theta})$, where $\mathcal{G}$ is the observed graph and $\boldsymbol{\theta}$ is a parameter vector. For example, typical formulations of exponential random graph models assume that the building blocks of real networks are given by such structures as k-stars and k-triangles, with different weights assigned to different structures, whereas Kronecker graphs assume that the number of edges and the number of nodes in a given network are related by a densification power law with a suitable densification parameter. In such frameworks, estimating the model from data reduces to fitting the model parameters, whereas the model structure remains fixed from the very beginning. The problem is that, in order for a parametric model to deliver an accurate estimate of the distribution at hand, its prior assumption concerning the form of the modeled distribution must be satisfied by the given data, which is something that we can rarely assess a priori. To date, the knowledge we have concerning real-world network phenomena does not allow to assume that any particular parametric assumption is really capturing in depth any network-generating law, although some observed properties may happen to be modeled fairly well.

The aim of this paper is twofold. On the one hand, we take a first step toward *nonparametric* modeling of random networks by developing a novel network statistic, which we call the *Fiedler delta* statistic. The Fiedler delta function allows to model different graph properties at once in an extremely compact form. This statistic is based on the spectral analysis of the graph Laplacian. In particular, it is based on the smallest non-zero eigenvalue of the Laplacian matrix, which is known as Fiedler value [Fiedler, 1973, Mohar, 1991]. On the other hand, we systematically adopt a *conditional* approach to random graph modeling, i.e. we focus on the conditional distribution of edges given some neighboring portion of the network, while setting aside the problem of estimating joint distributions. The resulting conditional random graph model is what we call *Fiedler random graph* (FRG). Roughly speaking, to model the conditional distribution of an edge variable X_{uv} with respect to its neighborhood, we compute the difference in Fiedler values for the vicinity subgraph including or excluding the edge $\{u, v\}$. The underlying intuition is that the variations encapsulated in the Fiedler delta for each particular edge will give a measure of the role of that edge in determining the algebraic connectivity of its neighborhood. As part of our contributions, we theoretically prove that FRGs capture edge dependencies at any distance within a given neighborhood, hence defining a fairly general class of conditional probability distributions over graphs.

Experiments on the estimation of (conditional) link distributions in large-scale networks show that FRGs are well suited for estimation problems on very large networks, especially small-world and scale-free networks. Our results reveal that the FRG model supersedes other approaches in terms of edge prediction accuracy, while allowing for efficient computation in the large-scale setting. In particular, it is known that the computation of the Fiedler delta scales polynomially in the size of the analyzed neighborhood [Bai et al., 2000]. And the experiments show that even for small sizes of neighborhoods (which allow for extremely fast computation), our model regularly outperforms the alternative ones—which we reformulate here in terms of conditional models so as to allow for transparent comparison.

The paper is organized as follows. Sec. 2 reviews some preliminary notions concerning the Laplacian spectrum of graphs. The FRG model is presented in Sec. 3, where we also show how to estimate FRGs from data, and we analyze the dependence structures involved in FRGs. In Sec. 4 we review (and to some extent develop) a few alternative conditional approaches to random graph estimation, starting from some well-known generative models. All considered models are then evaluated experimentally in Sec. 5, while Sec. 6 draws some conclusions and sketches a few directions for further work.

2 GRAPHS, LAPLACIANS, AND EIGENVALUES

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with n nodes. In the following we assume that the graph is unweighted with adjacency matrix $\mathbf{A}$. For each (unordered) pair of nodes $\{u, v\}$ such that $u \neq v$, we take X_{uv} to denote a binary random variable such that $X_{uv} = 1$ if $\{u, v\} \in \mathcal{E}$, and $X_{uv} = 0$ otherwise. Since the graph is undirected, $X_{uv} = X_{vu}$.

The degree d_u of a node $u \in \mathcal{V}$ is defined as the number of connections of u to other nodes, that is $d_u = |\{v : \{u, v\} \in \mathcal{E}\}|$. Accordingly, the degree matrix $\mathbf{D}$ of a graph $\mathcal{G}$ corresponds to the diagonal matrix with the vertex degrees $d_1, \ldots, d_n$ on the diagonal. The main tools exploited by the random graph model proposed here are the graph Laplacian matrices. Different graph Laplacians have been identified in the literature [von Luxburg, 2007]. In this paper, we use consistently the *unnormalized graph Laplacian*, defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The basic facts related to the unnormalized Laplacian matrix can be summarized as follows:

Proposition 1 (Mohar [1991])**.** *The unnormalized graph Laplacian* $\mathbf{L}$ *of an undirected graph* $\mathcal{G}$ *with* n

nodes has the following properties: (i) $\mathbf{L}$ *is symmetric and positive semi-definite; (ii)* $\mathbf{L}$ *has* n *non-negative, real-valued eigenvalues* $0 = \lambda_1(\mathcal{G}) \leq \ldots \leq \lambda_n(\mathcal{G})$*; (iii) the multiplicity of the eigenvalue 0 of* $\mathbf{L}$ *equals the number of connected components in the graph, that is,* $\lambda_2(\mathcal{G}) > 0$ *if and only if* $\mathcal{G}$ *is connected.*

In the following, the (algebraic) multiplicity of an eigenvalue λ will be denoted by $M(\lambda, \mathcal{G})$. If the graph has one single connected component, then $M(0, \mathcal{G}) = 1$, and the second smallest eigenvalue $\lambda_2(\mathcal{G}) > 0$ is called *Fiedler eigenvalue*. The Fiedler eigenvalue provides insight into several graph properties: when there is a nontrivial spectral gap, i.e. $\lambda_2(\mathcal{G})$ is clearly separated from 0, the graph has good expansion properties, stronger connectivity, and rapid convergence of random walks in the graph [Mohar, 1991]. For example, it is known that $\lambda_2(\mathcal{G}) \leq \mu(\mathcal{G})$, where $\mu(\mathcal{G})$ is the edge connectivity of the graph (i.e. the size of the smallest edge cut whose removal makes the graph disconnected). Notice that if the graph has more than one connected component, then $\lambda_2(\mathcal{G})$ will be also equal to zero, thus implying that the graph is not connected. Without loss of generality, we abuse the term Fiedler eigenvalue to denote the smallest eigenvalue different from zero, regardless of the number of connected components. In this paper, by Fiedler value we will mean the eigenvalue $\lambda_{k+1}(\mathcal{G})$, where $k = M(0, \mathcal{G})$.

For any pair of nodes u and v in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define two corresponding graphs $\mathcal{G}^+$ and $\mathcal{G}^-$ in the following way: $\mathcal{G}^+ = (\mathcal{V}, \mathcal{E} \cup \{\{u, v\}\})$, and $\mathcal{G}^- = (\mathcal{V}, \mathcal{E} \setminus \{\{u, v\}\})$. Clearly, we have that either $\mathcal{G}^+ = \mathcal{G}$ or $\mathcal{G}^- = \mathcal{G}$. A basic property concerning the Laplacian eigenvalues of $\mathcal{G}^+$ and $\mathcal{G}^-$ is the following [Mohar, 1991, Anderson and Morley, 1985, Cvetković et al., 1979]:

Lemma 1. *If* $\mathcal{G}^+$ *and* $\mathcal{G}^-$ *are two graphs with* n *nodes, such that* $\{u, v\} \subseteq \mathcal{V}$*,* $\mathcal{G}^+ = (\mathcal{V}, \mathcal{E} \cup \{\{u, v\}\})$*, and* $\mathcal{G}^- = (\mathcal{V}, \mathcal{E} \setminus \{\{u, v\}\})$*, then we have that:*

1. $\sum_{i=1}^{n} \lambda_i(\mathcal{G}^+) - \lambda_i(\mathcal{G}^-) = 2$*;*
2. $\lambda_i(\mathcal{G}^+) \leq \lambda_i(\mathcal{G}^-)$ *for any* i *such that* $1 \leq i \leq n$*.*

3 FIEDLER RANDOM GRAPHS

Fiedler random graphs are defined in Sec. 3.1, while in Secs. 3.2 and 3.3 we deal with the problems of estimating them from data and characterizing their statistical dependence structure respectively.

3.1 CONDITIONAL DISTRIBUTIONS

Given the notions reviewed above, we introduce the Fiedler delta function $\Delta\lambda_2$, which is defined as follows. Let $k = M(0, \mathcal{G}^+)$. Then,

$$\Delta\lambda_2(u, v, \mathcal{G}) = \lambda_{k+1}(\mathcal{G}^+) - \lambda_{k+1}(\mathcal{G}^-) \qquad (1)$$

In other words, for any pair of nodes u and v in graph $\mathcal{G}$, the Fiedler delta value of the pair $\{u, v\}$ in $\mathcal{G}$ is the (absolute) variation in the Fiedler eigenvalue of the graph Laplacian that would result from removing edge $\{u, v\}$ from $\mathcal{G}^+$.

Lemma 1 immediately implies the following proposition:

Proposition 2. *For any graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *and any pair of nodes* u *and* v *such that* $\{u, v\} \in \mathcal{E}$*, we have that* $0 \leq \Delta\lambda_2(u, v, \mathcal{G}) \leq 2$*.*

Proof. Let $\mathcal{G}^- = (\mathcal{V}, \mathcal{E} \setminus \{\{u, v\}\})$ and $k = M(0, \mathcal{G})$. The theorem follows straightforwardly from Lemma 1, given that $\Delta\lambda_2(u, v, \mathcal{G}) = \lambda_{k+1}(\mathcal{G}) - \lambda_{k+1}(\mathcal{G}^-)$. □

Using the Fiedler delta statistic, we can proceed to define a Fiedler random graph. Given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $\mathcal{X}(\mathcal{G})$ denote the set of random variables defined on $\mathcal{G}$, that is $\mathcal{X}(\mathcal{G}) = \{X_{uv} \colon u \neq v \wedge \{u, v\} \subseteq \mathcal{V}\}$. Then:

Definition 1. *Given a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, suppose we associate a subgraph* $\mathcal{G}_{uv}$ *of* $\mathcal{G}$ *to each variable* $X_{uv} \in \mathcal{X}(\mathcal{G})$*. We say that* $\mathcal{G}$ *is a* Fiedler random graph *if, for any* $X_{uv} \in \mathcal{X}(\mathcal{G})$*, we have that* $P(X_{uv} | \, \mathcal{X}(\mathcal{G}_{uv}) \setminus \{X_{uv}\}) = P(X_{uv} | \, \Delta\lambda_2(u, v, \mathcal{G}_{uv}))$*.*

Concerning Definition 1, we stress the importance of two points. First, the key intuition motivating FRGs is to treat the Fiedler delta function as a real-valued random variable defined over graph configurations, and to exploit this random variable as a compact representation of those configurations. Second, FRGs are *conditional* models, i.e. they only define conditional distributions of edges in a random graph given the possible configurations of other edges. The model definition does not specify whether and how the considered conditional distributions can be combined to obtain a consistent factorization of the joint distribution of $\mathcal{X}(\mathcal{G})$. In order to obtain that result, the model should take into account the global dependence structure of $\mathcal{X}(\mathcal{G})$, but this goal is not pursued by FRG modeling. Two interesting questions that can be raised on the nature of the FRG model are then the following. First, how does the Fiedler delta statistic behave with respect to the Markov dependence property [Besag, 1974, Frank and Strauss, 1986]? Second, how useful can be the information enclosed in the Fiedler delta statistic for learning and mining applications over large-scale networks? One basic result related to the first question is presented in Sec. 3.3, whereas Sec. 5 will address the second point.

3.2 MODEL ESTIMATION

The goal of estimating a FRG from data is to obtain an estimate of the conditional distribution $P(X_{uv}|\,\mathcal{X}(\mathcal{G}_{uv}) \setminus \{X_{uv}\})$, defined over subgraphs $\mathcal{G}_{uv}$ of $\mathcal{G}$. Starting from Definition 1, this quantity can be manipulated as follows:

$$\begin{aligned} P(X_{uv}|\,\mathcal{X}(\mathcal{G}_{uv}) \setminus \{X_{uv}\}) &= P(X_{uv}|\,\Delta\lambda_2(u,v,\mathcal{G}_{uv})) \\ &= \frac{p(\Delta\lambda_2(u,v,\mathcal{G}_{uv})|\,X_{uv})P(X_{uv})}{p(\Delta\lambda_2(u,v,\mathcal{G}_{uv}))} \\ &= \frac{p(\Delta\lambda_2(u,v,\mathcal{G}_{uv})|\,X_{uv})P(X_{uv})}{\sum\limits_{x_{uv}} p(\Delta\lambda_2(u,v,\mathcal{G}_{uv})|\,x_{uv})P(x_{uv})} \end{aligned} \tag{2}$$

In Eq. 2, $P(X_{uv})$ is the prior probability of observing an edge between any two nodes u and v, whereas $p(\Delta\lambda_2(u,v,\mathcal{G}_{uv})|\,X_{uv})$ is the conditional density of the Fiedler delta $\Delta\lambda_2(u,v,\mathcal{G}_{uv})$ given the value of X_{uv}. Suppose we have a network $\mathcal{G} = (\mathcal{V},\mathcal{E})$, and that our training sample is a dataset $\mathcal{D}$, defined as $\mathcal{D} = \big\{(x_{u_1v_1},\mathcal{G}_{u_1v_1}),\ldots,(x_{u_nv_n},\mathcal{G}_{u_nv_n})\big\}$. In other words, the training data are n observations of node pairs $\{u_i,v_i\}$ such that, for each one of them, we know both the value of $X_{u_iv_i}$ and the configuration of $\mathcal{G}_{u_iv_i}$ in $\mathcal{G}$. A simple estimate for $P(X_{uv})$ can then be obtained by computing the proportion of linked/unlinked node pairs $\{u_i,v_i\}$ in $\mathcal{D}$. Let $\mathcal{D}_{x_{uv}}$ denote the set $\big\{\{u_i,v_i\}\colon (x_{u_iv_i},\mathcal{G}_{u_iv_i}) \in \mathcal{D} \wedge x_{u_iv_i} = x_{uv}\big\}$. Then, $\widehat{P}(x_{uv}) = \frac{1}{n}\left|\mathcal{D}_{x_{uv}}\right|$. On the other hand, in order to estimate $p(\Delta\lambda_2(u,v,\mathcal{G}_{uv})|\,X_{uv})$, we need to make a choice concerning the form of the modeled density function. In the lack of prior knowledge concerning the form of the density, a reasonable choice is to adopt a nonparametric estimation technique. In particular, we use a kernel density estimator [Rosenblatt, 1956, Parzen, 1962], defined as follows:

$$\begin{aligned} &\hat{p}(\Delta\lambda_2(u,v,\mathcal{G}_{uv})|\,x_{uv}) = \\ &= \frac{1}{|\mathcal{D}_{x_{uv}}|} \sum_{\{u_i,v_i\}\in\mathcal{D}_{x_{uv}}} K_{\Delta\lambda_2}\big(\{u,v\},\{u_i,v_i\}\big) \end{aligned} \tag{3}$$

where

$$\begin{aligned} &K_{\Delta\lambda_2}\big(\{u,v\},\{u_i,v_i\}\big) = \\ &= \frac{1}{h}\,K\left(\frac{\Delta\lambda_2(u,v,\mathcal{G}_{uv}) - \Delta\lambda_2(u_i,v_i,\mathcal{G}_{u_iv_i})}{h}\right) \end{aligned} \tag{4}$$

and K is a kernel function with bandwidth h [Silverman, 1986]. In our implementation of FRGs, we use the Epanechnikov kernel [Epanechnikov, 1969], namely $K(t) = \frac{3}{4}\big(1-t^2\big)\,\mathbb{1}_1(|t|)$, where $\mathbb{1}_y(x) = 1$ if $x \le y$, and $\mathbb{1}_y(x) = 0$ otherwise. The Epanechnikov kernel is more appealing than other possible functions (such as the Gaussian kernel) because of its computational convenience. At the same time, it displays analogous properties in terms of statistical consistency [Silverman, 1986].

3.3 DEPENDENCE STRUCTURE

In order to illustrate the behavior of FRGs with respect to conditional independence, we first recall the definition of Markov dependence for random graph models [Frank and Strauss, 1986]. Let $\mathcal{N}(X_{uv})$ denote the set $\{X_{wz}\colon \{w,z\} \in \mathcal{E} \wedge |\{w,z\} \cap \{u,v\}| = 1\}$. A Markov random graph is then defined as follows:

Definition 2. *A random graph $\mathcal{G}$ is said to be a* Markov graph *(or to have a* Markov dependence structure*) if, for any pair of variables X_{uv} and X_{wz} in $\mathcal{G}$ such that $\{u,v\} \cap \{w,z\} = \emptyset$, we have that $P(X_{uv}|\,X_{wz},\mathcal{N}(X_{uv})) = P(X_{uv}|\,\mathcal{N}(X_{uv}))$.*

Based on Definition 2, we say that the dependence structure of a random graph $\mathcal{G}$ is *non-Markovian* if, for disjoint pairs of nodes $\{u,v\}$ and $\{w,z\}$, it does not imply that $P(X_{uv}|\,X_{wz},\mathcal{N}(X_{uv})) = P(X_{uv}|\,\mathcal{N}(X_{uv}))$, i.e. if it is consistent with the inequality $P(X_{uv}|\,X_{wz},\mathcal{N}(X_{uv})) \neq P(X_{uv}|\,\mathcal{N}(X_{uv}))$. We can then prove the following proposition:

Proposition 3. *There exist Fiedler random graphs with non-Markovian dependence structure.*

Proof. To prove the proposition, we make use of the following equalities [Fiedler, 1973]: if graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are, respectively, a path and a circuit of size n, then $\lambda_2(\mathcal{G}_1) = 2\,(1-\cos(\pi/n))$ and $\lambda_2(\mathcal{G}_2) = 2\,(1-\cos(2\pi/n))$. Consider a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ such that $\mathcal{V} = \{u,v,w,z\}$ and $\mathcal{E} = \{\{u,v\},\{v,w\},\{w,z\},\{u,z\}\}$, and let $\mathcal{G}_{uv} = (\mathcal{V}_{uv},\mathcal{E}_{uv})$ be the subgraph of $\mathcal{G}$ incident to $\mathcal{E}_{uv} = \mathcal{E}\setminus\{\{w,z\}\}$. If $\mathcal{G}$ is a Markov graph, it must be the case that $P(X_{uv}|\,X_{wz},X_{vw},X_{uz}) = P(X_{uv}|\,X_{vw},X_{uz})$. Now, suppose that $\mathcal{G}$ is a FRG. In this case, we have that

$$\begin{aligned} P(X_{uv}|\,X_{wz},X_{vw},X_{uz}) &= P(X_{uv}|\,\mathcal{X}(\mathcal{G}) \setminus \{X_{uv}\}) \\ &= \frac{p(\Delta\lambda_2(u,v,\mathcal{G})|\,X_{uv})P(X_{uv})}{p(\Delta\lambda_2(u,v,\mathcal{G}))} \end{aligned} \tag{5}$$

and, similarly,

$$P(X_{uv}|\,X_{vw},X_{uz}) = \frac{p(\Delta\lambda_2(u,v,\mathcal{G}_{uv})|\,X_{uv})P(X_{uv})}{p(\Delta\lambda_2(u,v,\mathcal{G}_{uv}))} \tag{6}$$

Eqs. 5–6 imply that, if the dependence structure of $\mathcal{G}$ is Markovian, then the following equality must hold:

$$\frac{p(\Delta\lambda_2(u,v,\mathcal{G})|\,X_{uv})}{p(\Delta\lambda_2(u,v,\mathcal{G}))} = \frac{p(\Delta\lambda_2(u,v,\mathcal{G}_{uv})|\,X_{uv})}{p(\Delta\lambda_2(u,v,\mathcal{G}_{uv}))} \tag{7}$$

Since the configuration of $\mathcal{G}$ and $\mathcal{G}_{uv}$ is given by a circuit and a path respectively, where both have size 4, we know that $\lambda_2(\mathcal{G}) = 2\,(1-\cos(\pi/2))$ and $\lambda_2(\mathcal{G}_{uv}) = 2\,(1-\cos(\pi/4))$. Also, if $\mathcal{G}^- = (\mathcal{V}, \mathcal{E} \setminus \{\{u,v\}\})$ and $\mathcal{G}^-_{uv} = (\mathcal{V}_{uv}, \mathcal{E}_{uv} \setminus \{\{u,v\}\})$, notice that $\lambda_2(\mathcal{G}^-) = \lambda_2(\mathcal{G}_{uv})$, since $\mathcal{G}^-$ is also a path of size 4, and that $M(0,\mathcal{G}^-_{uv}) = M(0,\mathcal{G}_{uv}) + 1$, since $\mathcal{G}^-_{uv}$ has one more connected component than $\mathcal{G}_{uv}$. Therefore, we have that $\Delta\lambda_2(u,v,\mathcal{G}) = 2\cos(\pi/4)$ and $\Delta\lambda_2(u,v,\mathcal{G}_{uv}) = 2\,(1-\cos(\pi/4))$, i.e. $\Delta\lambda_2(u,v,\mathcal{G}) \neq \Delta\lambda_2(u,v,\mathcal{G}_{uv})$. Because of this inequality, there will exist parameterizations of p which do not satisfy Eq. 7, which means that the dependence structure of $\mathcal{G}$ is non-Markovian. □

Note that the proof of Proposition 3 can be straightforwardly generalized to the dependence between X_{uv} and X_{wz} in circuits/paths of arbitrary size n, since the expression used for the Fiedler eigenvalues of such graphs holds for any n. In fact, suppose that the 4-nodes circuit $\mathcal{G}$ used in the proof is replaced with a circuit $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ of size n, where $\mathcal{V}^* = \mathcal{V} \cup \{s_1,\ldots,s_1,\ldots,s_m,t_1,\ldots,t_m\}$ and $\mathcal{E}^*$ is obtained from $\mathcal{E}$ by replacing $\{u,z\}$ and $\{v,w\}$, respectively, with a path from u to z going through $s_1,\ldots,s_m$ and a path from v to w going through $t_1,\ldots,t_m$, so that $n = 2m+4$. In this case, if $\mathcal{G}^*_{uv}$ is the subgraph of $\mathcal{G}^*$ incident to $\mathcal{E}^*_{uv} = \mathcal{E}^* \setminus \{\{w,z\}\}$, we have again that $\Delta\lambda_2(u,v,\mathcal{G}^*) \neq \Delta\lambda_2(u,v,\mathcal{G}^*_{uv})$, which means that there exist FRGs such that $\frac{p(\Delta\lambda_2(u,v,\mathcal{G}^*)|\,X_{uv})}{p(\Delta\lambda_2(u,v,\mathcal{G}^*))} \neq \frac{p(\Delta\lambda_2(u,v,\mathcal{G}^*_{uv})|\,X_{uv})}{p(\Delta\lambda_2(u,v,\mathcal{G}^*_{uv}))}$.

4 OTHER CONDITIONAL RANDOM NETWORKS

In this section, we adapt three families of random graph models—namely the ERG, Watts-Strogatz (WS), and Barabási-Albert (BA) models respectively—to the task of conditional estimation of edge distributions, and we also review the entailed dependence structures. This will allow for a clear comparison of such models to FRGs in Sec. 5. Kronecker graphs are not considered here because it is not straightforward how they could be turned into conditional estimators.

4.1 EXPONENTIAL RANDOM GRAPH MODELS

Let $G(\mathcal{V})$ be the set of all possible (undirected) graphs $\mathcal{G}$ with fixed vertex set $\mathcal{V}$, where $|G(\mathcal{V})| = \sqrt{2^{|\mathcal{V}|^2-|\mathcal{V}|}}$. Suppose that Φ denotes a collection of potential functions φ_i over graphs, where each $\varphi_i(\mathcal{G})$ is a graph statistic such as the number of edges or the number of triangles in $\mathcal{G}$. Then, an ERG with parameter vector $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_{|\Phi|})$ defines the following Boltzmann distribution [Robins et al., 2007]:

$$P(\mathcal{X}(\mathcal{G})|\,\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\exp\left\{\sum_{i=1}^{|\Phi|}\theta_i\varphi_i(\mathcal{G})\right\} \tag{8}$$

where each θ_i is a real-valued parameter associated to the potential function φ_i, and $Z(\boldsymbol{\theta})$ denotes the partition function [Koller and Friedman, 2009], given by $Z(\boldsymbol{\theta}) = \sum_{\mathcal{G}_j\in G(\mathcal{V})}\exp\big\{\sum_{i=1}^{|\Phi|}\theta_i\varphi_i(\mathcal{G}_j)\big\}$. Computing the partition function exactly is clearly intractable for graphs with more than a few nodes, which makes it quite expensive to compute joint probabilities in large networks using ERGs.

ERGs entail a simple form for the conditional distribution of edges given the remainder of the graph:

$$P(X_{uv}|\,\mathcal{X}(\mathcal{G})\setminus\{X_{uv}\};\boldsymbol{\theta}) = \frac{\exp\left\{\sum_{i=1}^{|\Phi|}\theta_i\varphi_i(\mathcal{G})\right\}}{\sum_{x_{uv}}\exp\left\{\sum_{i=1}^{|\Phi|}\theta_i\varphi_i(\mathcal{G}_{x_{uv}})\right\}} \tag{9}$$

where $\mathcal{G}_{x_{uv}}$ denotes the configuration of $\mathcal{G}$ that we obtain by clamping the value of X_{uv} to x_{uv}. Therefore, if we want to estimate the parameters of an ERG so as to maximize the model log-likelihood with respect to a data sample $\mathcal{D} = \big\{(x_{u_1v_1},\mathcal{G}_{\mathcal{S}_1}),\ldots,(x_{u_nv_n},\mathcal{G}_{\mathcal{S}_n})\big\}$, we can optimize each parameter θ_i by taking the derivative $\frac{\partial}{\partial\theta_i}\sum_{j=1}^{n}\log P(x_{u_jv_j}|\,\mathcal{X}(\mathcal{G}_{\mathcal{S}_j})\setminus\{X_{uv}\};\boldsymbol{\theta})$ and applying any gradient-based optimization technique.

4.1.1 Markov Random Graphs

Given a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$, define a k-star (for $k \geq 2$) as a set $\mathcal{E}_{\mathcal{S}_k} \subseteq \mathcal{E}$ such that $|\mathcal{E}_{\mathcal{S}_k}| = k$ and, for any $\{u,v\}$ and $\{w,z\}$ in $\mathcal{E}_{\mathcal{S}_k}$, $\{u,v\}\cap\{w,z\} = 1$. Also, let a k-triangle (for $k \geq 1$) be a set $\mathcal{E}_{\mathcal{T}_k} \subseteq \mathcal{E}$ of size $2k+1$ such that the elements of $\mathcal{E}_{\mathcal{T}_k}$ are edges $\{u,v\},\{u,w_1\},\{v,w_1\},\ldots,\{u,w_k\},\{v,w_k\}$, where $w_i \neq w_j$ for any i and j such that $1 \leq i,j \leq k$. If we use $E(\mathcal{G})$, $S_k(\mathcal{G})$, and $T_k(\mathcal{G})$ to denote, respectively, the number of edges, the number of k-stars, and the number of k-triangles in $\mathcal{G}$, then an *exponential Markov random graph* (MRG) model with parameter vector $\boldsymbol{\theta}$ is defined by the following probability distribution [Frank and Strauss, 1986]:

$$P(\mathcal{G}|\,\boldsymbol{\theta}) = \frac{1}{Z}\exp\left\{\eta\,E(\mathcal{G}) + \sum_{k=2}^{K}\sigma_k S_k(\mathcal{G}) + \tau\,T(\mathcal{G})\right\} \tag{10}$$

where K is the maximum value that we want to consider for k in the k-star statistics, $\boldsymbol{\theta} = (\eta,\sigma_2,\ldots,\sigma_K,\tau)$, and $T(\mathcal{G}) = T_1(\mathcal{G})$ is simply the number of triangles in $\mathcal{G}$. That is to say, a MRG defines a Boltzmann distribution over graphs, with parameters corresponding to edge, k-star, and triangle

statistics. Interestingly, the ER model can be characterized as a special case of MRG, where all the parameters except for η are set to zero.

MRGs are known to satisfy Definition 2 [Robins et al., 2007]. Therefore, under the probability model defined by Eq. 10, we have that, for any pair of nodes $\{u, v\}$ and any subgraph sample $\mathcal{G}_{\mathcal{S}}$ from $\mathcal{G}$, $P(X_{uv} | \mathcal{X}(\mathcal{G}_{\mathcal{S}}) \setminus \{X_{uv}\}; \boldsymbol{\theta}) = P(X_{uv} | \mathcal{N}_{\mathcal{S}}(X_{uv}); \boldsymbol{\theta})$, where $\mathcal{N}_{\mathcal{S}}(X_{uv})$ is the set of all variables X_{wz} from $\mathcal{G}_{\mathcal{S}}$ such that $|\{w, z\} \cap \{u, v\}| = 1$. Note that while Definition 2 specifies a general class of random graph models, MRGs in the strict sense refer to the class of ERGs defined above.

4.1.2 Higher-Order Models

Higher-order ERG models (HRGs) are readily obtained from MRGs by adding k-triangle counts (for $k \geq 1$) to the Boltzmann distribution of Eq. 10 [Robins et al., 2007]. Moreover, in order to avoid fixing in advance the maximum value of the k parameter, a general formulation has been proposed for HRGs through the *alternating k-star* and the *alternating k-triangle* statistics [Snijders et al., 2006], obtaining the following probability model:

$$P(\mathcal{G} | \boldsymbol{\theta}) = \frac{1}{Z} \exp \{\eta\, E(\mathcal{G}) + \sigma\, S^*(\mathcal{G}) + \tau\, T^*(\mathcal{G})\} \quad (11)$$

where, if n is the number of nodes in $\mathcal{G}$ and S_k and T_k are the usual k-stars and k-triangle counts, $S^*(\mathcal{G})$ and $T^*(\mathcal{G})$ are defined as $S^*(\mathcal{G}) = \sum_{k=2}^{n-1} (-1)^k \frac{S_k}{\rho^{k-2}}$ and $T^*(\mathcal{G}) = \sum_{k=2}^{n-2} (-1)^k \frac{T_k}{\rho^{k-2}}$ (with $\rho \geq 1$ acting as a sort of regularization parameter).

It is known that, under the distribution given by Eq. 11, $P(X_{uv} | \mathcal{X}(\mathcal{G}_{\mathcal{S}}) \setminus \{X_{uv}\}; \boldsymbol{\theta}) = P(X_{uv} | \mathcal{N}^*_{\mathcal{S}}(X_{uv}); \boldsymbol{\theta})$ [Robins et al., 2007], where $\mathcal{N}^*_{\mathcal{S}}(X_{uv})$ is the set containing any X_{wz} from $\mathcal{G}$ such that $X_{wz} \neq X_{uv}$ and, for at least one edge $\{s, t\}$, we have that $|\{s, t\} \cap \{u, v\}| = 1$ and $\{s, t\} \cap \{w, z\} \neq \emptyset$. Clearly, $\mathcal{N}_{\mathcal{S}}(X_{uv}) \subseteq \mathcal{N}^*_{\mathcal{S}}(X_{uv})$, which is why HRGs are 'higher-order' than MRGs.

4.2 WATTS-STROGATZ MODEL

The WS network model [Watts and Strogatz, 1998] defines a random network as a regular ring lattice which is randomly 'rewired' so as to introduce a certain amount of disorder, which typically leads to small-world phenomena. Given nodes $u_1, \ldots, u_n$, a WS network is generated by constructing a regular ring lattice such that each node is connected to exactly $2\,\delta$ other nodes. Network edges are then scanned sequentially, and each one of them is rewired with probability β, where, if $i < j$, rewiring an edge $\{u_i, u_j\}$ means to replace it with another edge $\{u_i, u_k\}$ such that $k \neq i$ and u_k is chosen uniformly at random from the set of all nodes that are not already linked to u_i.

Interestingly, the degree distribution corresponding to a WS network $\mathcal{G}$ with parameters δ and β takes the following form [Barrat and Weigt, 2000], for any degree $k \geq \delta$:

$$P(k) = \sum_{i=0}^{I} \binom{\delta}{i} (1-\beta)^i\, \beta^{\delta - i}\, \frac{(\delta\,\beta)^{k-\delta-i}}{(k-\delta-i)!} \exp(-\beta\,\delta) \quad (12)$$

where $I = \min\{k - \delta, \delta\}$. Given Eq. 12, we model the conditional distribution of a variable X_{uv} given the remainder of $\mathcal{G}$ through the following quantity:

$$P(X_{uv} | \mathcal{X}(\mathcal{G}) \setminus \{X_{uv}\}) = \frac{P(d_u(\mathcal{G}))\, P(d_v(\mathcal{G}))}{\sum\limits_{x_{uv}} P(d_u(\mathcal{G}_{x_{uv}}))\, P(d_v(\mathcal{G}_{x_{uv}}))} \quad (13)$$

where $d_u(\mathcal{G})$ denotes the degree of node u in $\mathcal{G}$, and each P is implicitly conditional on the values of δ and β. We refer to the conditional random graph model specified in Eq. 13 as the *conditional Watts-Strogatz (CWS) model*.

For the CWS model, the following proposition follows straightforwardly from Eq. 13:

Proposition 4. *The dependence structure of CWS models is Markovian.*

A simple estimate of the δ parameter for a data sample $\mathcal{D} = \big\{(x_{u_1 v_1}, \mathcal{G}_{u_1 v_1}), \ldots, (x_{u_n v_n}, \mathcal{G}_{u_n v_n})\big\}$ is the following:

$$\hat{\delta} = \frac{1}{2\,n} \sum_{i=1}^{n} d_{u_i}(\mathcal{G}_{\mathcal{S}_i}) + d_{v_i}(\mathcal{G}_{u_i v_i}) \quad (14)$$

On the other hand, a simple strategy for estimating the rewiring probability by a maximum likelihood approach consists in parameterizing it as a sigmoid function $\beta = \frac{1}{1+\exp(-\theta_\beta)}$, where θ_β can be optimized by exploiting the derivative $\frac{\partial}{\partial \theta_\beta} \sum_{i=1}^{n} \log P(x_{u_i v_i} | \mathcal{X}(\mathcal{G}_{u_i v_i}) \setminus \{X_{uv}\}; \delta, \theta_\beta)$.

4.3 BARABÁSI-ALBERT MODEL

The BA model was originally proposed for explaining the scale-free degree distributions often observed in real-world networks [Barabási and Albert, 1999]. In the BA model, the probability $P(u)$ of linking to any particular node u in a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ takes the form $P(u) = \frac{d_u(\mathcal{G})^\alpha}{\sum_{v \in \mathcal{V}} d_v(\mathcal{G})^\alpha}$ [Albert and Barabási, 2002], where α is a real-valued parameter affecting the shape of the degree distribution. Given $P(u)$, we can use the following expression to characterize the conditional probability of observing edge $\{u, v\}$ given the

remainder of $\mathcal{G}$ [Newman, 2001, Barabási et al., 2002]:

$$P(X_{uv} = 1 | \mathcal{X}(\mathcal{G}) \setminus \{X_{uv}\}; \alpha) = \frac{d_u(\mathcal{G})^\alpha \, d_v(\mathcal{G})^\alpha}{\left(\sum_{w \in \mathcal{V}} d_w(\mathcal{G})^\alpha\right)^2} \tag{15}$$

We refer to the random graph model of Eq. 15 as the *conditional Barabási-Albert (CBA) model*. The α parameter can be optimized straightforwardly by gradient-based methods, so as to maximize the objective $\sum_{i=1}^{n} \log P(x_{u_i v_i} | \mathcal{X}(\mathcal{G}_{u_i v_i}) \setminus \{X_{uv}\}; \alpha)$ over an observed sample of size n.

The following property holds for CBA networks:

Proposition 5. *The dependence structure of CBA models is non-Markovian.*

Proof. Consider a CBA network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\{u, v\}$ and $\{w, z\}$ are edges in $\mathcal{E}$ such that $\{u, v\} \cap \{w, z\} = \emptyset$. $\mathcal{G}$ is a Markov graph if and only if $P(X_{uv} | X_{wz}, \mathcal{N}(X_{uv})) = P(X_{uv} | \mathcal{N}(X_{uv}))$. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be the same as defined in the proof of Proposition 4. Although $d_u(\mathcal{G}_2) = d_u(\mathcal{G}_1)$ and $d_v(\mathcal{G}_2) = d_v(\mathcal{G}_1)$, we have that $\sum_{w \in \mathcal{V}} d_w(\mathcal{G}_2)^\alpha > \sum_{w \in \mathcal{V}} d_w(\mathcal{G}_1)^\alpha$. Therefore,

$$\begin{aligned} P(X_{uv} | X_{wz}, \mathcal{N}(X_{uv})) &= P(X_{uv} | \mathcal{X}(\mathcal{G}_2) \setminus \{X_{uv}\}) \\ &= \frac{d_u(\mathcal{G}_2)^\alpha \, d_v(\mathcal{G}_2)^\alpha}{\left(\sum_{w \in \mathcal{V}} d_w(\mathcal{G}_2)^\alpha\right)^2} \neq \frac{d_u(\mathcal{G}_1)^\alpha \, d_v(\mathcal{G}_1)^\alpha}{\left(\sum_{w \in \mathcal{V}} d_w(\mathcal{G}_1)^\alpha\right)^2} \\ &= P(X_{uv} | \mathcal{X}(\mathcal{G}_1) \setminus \{X_{uv}\}) = P(X_{uv} | \mathcal{N}(X_{uv})) \end{aligned} \tag{16}$$

□

5 EXPERIMENTAL EVALUATION

In order to assess experimentally the accuracy of FRGs as models of conditional edge distributions in large-scale networks, we test them in the following link prediction setting. First, we take a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with a number of nodes ranging from a few thousands to a couple of millions, and a number of edges going up to several millions. We then sample n training pairs $\{u_i, v_i\}$ uniformly at random from $\mathcal{V}$, with $u_i \neq v_i$. For each sampled pair of nodes $\{u_i, v_i\}$, we recover the subgraph $\mathcal{G}_{u_i v_i}$ induced on $\mathcal{G}$ by the vertex set $\mathcal{V}_{u_i v_i} = \{u_i, v_i\} \cup \{w_i : \{u_i, w_i\} \in \mathcal{E} \vee \{v_i, w_i\} \in \mathcal{E}\}$, so as to get a training set $\mathcal{D} = \left\{(x_{u_1 v_1}, \mathcal{G}_{u_1 v_1}), \ldots, (x_{u_n v_n}, \mathcal{G}_{u_n v_n})\right\}$. We then estimate from $\mathcal{D}$ each one of the conditional random graph models described in the previous sections, using the respective learning techniques. Given the estimated models, we sample a test set $\mathcal{T} = \left\{(x_{u_1 v_1}, \mathcal{G}_{u_1 v_1}), \ldots, (x_{u_m v_m}, \mathcal{G}_{u_m v_m})\right\}$ from $\mathcal{G}$ such that $\mathcal{T} \cap \mathcal{D} = \emptyset$, where each element of $\mathcal{T}$ is sampled through the same technique used for $\mathcal{D}$. For each estimated model, we assess its accuracy by first using it to compute the conditional probability of observing a link for each pair of nodes $\{u_j, v_j\}$ in $\mathcal{T}$, and then calculating the area under the ROC curve (AUC) for the predictions delivered on $\mathcal{T}$. Since the problem of predicting the presence of edges in real-world networks is severely unbalanced, i.e. $|\mathcal{E}| \ll \frac{1}{2}\left(|\mathcal{V}|^2 - |\mathcal{V}|\right)$, the ROC curve is a particularly appropriate evaluation measure [Fawcett, 2006]. Also, in order to assess the sensitivity of each model to the size of the training sample, we ran some preliminary experiments with several different values of $|\mathcal{D}|$. The result of this preliminary investigation was that, provided that the training sample contains at least a few pairs of linked nodes (in the order of ten or twenty), then the behavior of the different models is not significantly affected by increasing sample sizes.

Table 1: General statistics for the network datasets used in the experiments. $CC_{\mathcal{G}}$ and $D_{\mathcal{G}}$ denote the average clustering coefficient and the network diameter respectively.

Network	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$CC_{\mathcal{G}}$	$D_{\mathcal{G}}$
AstroPh	18,772	396,160	0.63	14
GrQc	5,242	28,980	0.52	17
HepPh	12,008	237,010	0.61	13
HepTh	9,877	51,971	0.47	17
Enron	36,692	367,662	0.49	12
RoadCA	1,965,206	5,533,214	0.04	850
RoadTX	1,379,917	3,843,320	0.04	1,049

We compare the performance of the MRG, HRG, CWS, CBA, and FRG models on seven different networks, drawn from the arXiv e-print repository, the Enron email corpus, and the US road network respectively [Leskovec et al., 2007, 2009]. The ER model is not considered in these experiments, since the involved independence assumption makes that model simply unusable for conditional estimation tasks. Some representative network statistics for the used datasets are summarized in Table 1. For the arXiv datasets (i.e. the AstroPh, GrQc, HepPh, and HepTh networks), we report results for a choice of $10,000$ node pairs both for the training and the test set, whereas for the Enron and US road (RoadCA and RoadTX) networks we have $|\mathcal{D}| = |\mathcal{T}| = 100,000$ and $|\mathcal{D}| = |\mathcal{T}| = 10,000,000$ respectively. However, notice that nearly identical results can be obtained in each case after reducing the number of training samples down to even 20%–40% of the indicated size (as discovered through preliminary analysis). ROC curves for the described datasets are plotted in Figs. 1–3, whereas the corresponding AUC values are given in Table 2. As a general remark, we point out that AUC values less

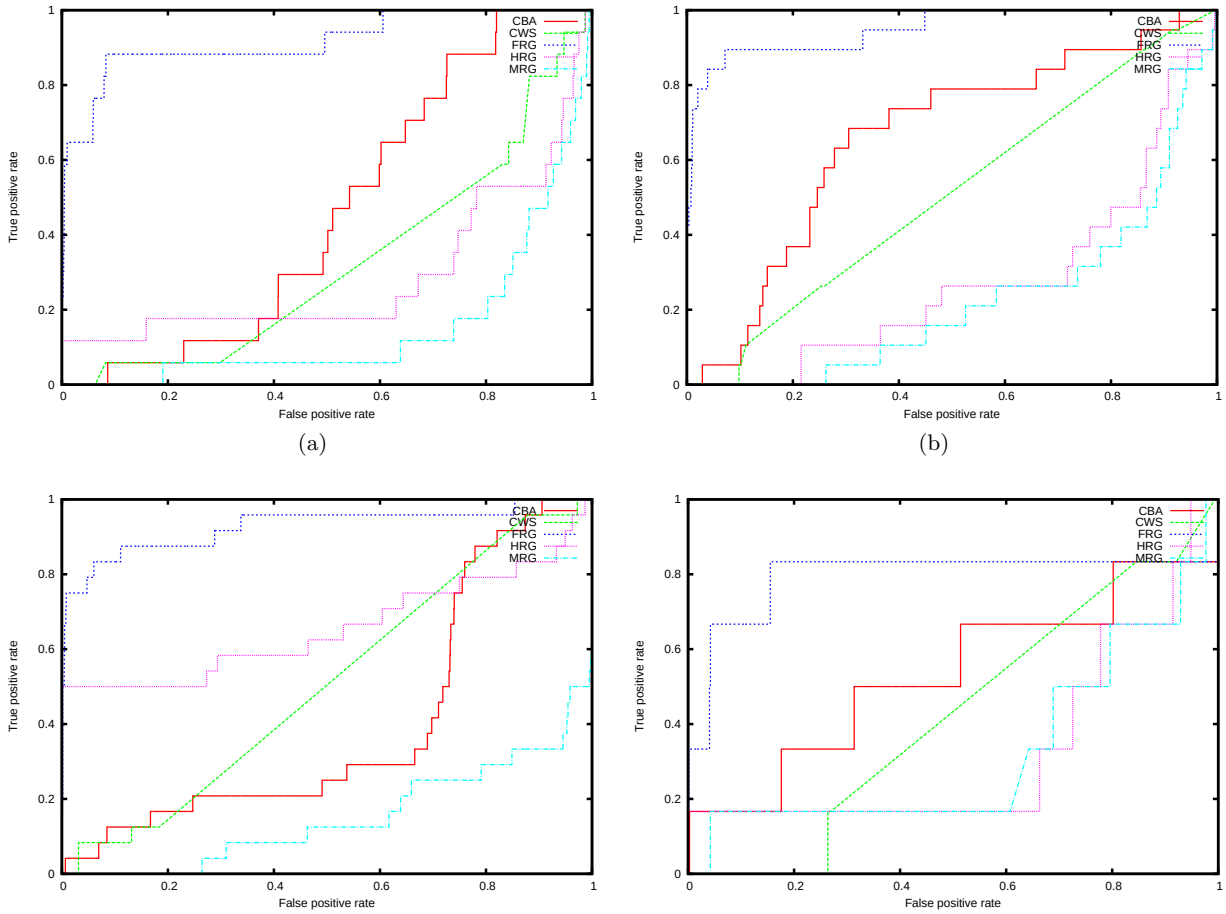


Figure 1: ROC curves for the AstroPh (a), GrQc (b), HepPh (c), and HepTh (d) networks.

than 0.5 (as recorded for some models) do not necessarily mean that prediction accuracy is being worse than random guessing. In fact, recent studies show that, depending on signal strength and stratification errors between training and test sets, random guessing can drop to AUC values even lower than 0.3 [Parker et al., 2007]. Therefore, AUC values less than 0.5 should be considered simply as *no better* than random guessing, rather than *worse* than random.

FRGs significantly outperform all other models both in the arXiv and in the Enron networks. Such networks are generally characterized by power law degree distributions, contrary to the road networks, which exhibit instead a regular topology, with degree distribution peaked around its mean. This explains the better behavior of the CBA model with respect to the remaining options in most of the scale-free networks. Interestingly, this trend is reverted for the HepPh data, where not only CWS outperforms CBA, but we also observe that HRG behaves much better than it does on the other scale-free networks. The fact that, in this particular case, FRGs remain the best available option provides evidence for their higher stability (i.e. robustness to variation in the underlying network properties) with

Table 2: AUC values for the ROC curves plotted in Figs. 1–3. $CC_{\mathcal{G}}$ and $D_{\mathcal{G}}$ denote the average clustering coefficient and the network diameter respectively.

	AUC (%)				
Network	**CBA**	**CWS**	**FRG**	**HRG**	**MRG**
AstroPh	46.04	32.39	**91.66**	28.69	14.78
GrQc	66.22	50.86	**94.90**	27.10	22.32
HepPh	40.04	50.84	**92.73**	65.62	15.02
HepTh	53.21	42.47	**79.34**	32.82	32.36
Enron	48.45	49.03	**85.92**	19.94	11.23
RoadCA	30.84	**91.21**	66.70	77.90	81.12
RoadTX	34.67	**93.01**	63.75	62.51	68.36

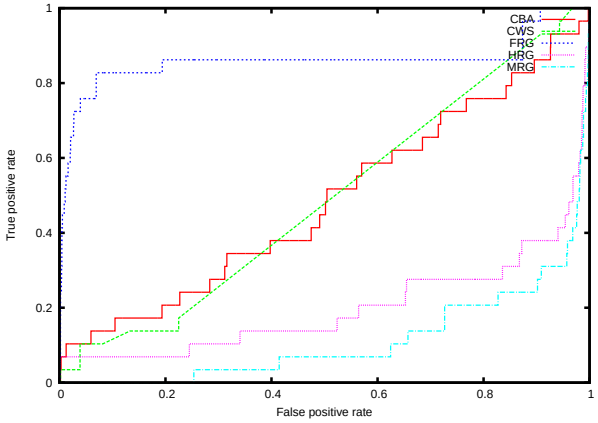
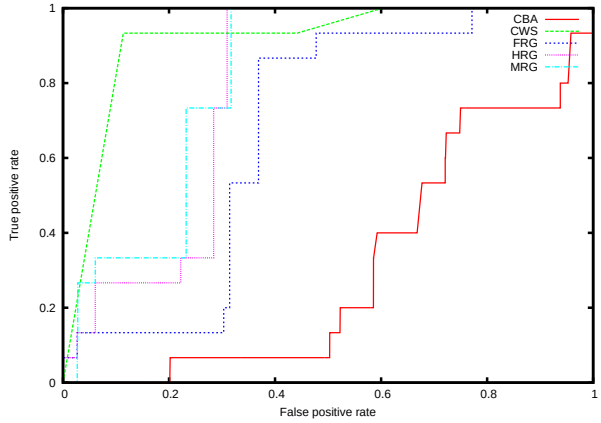
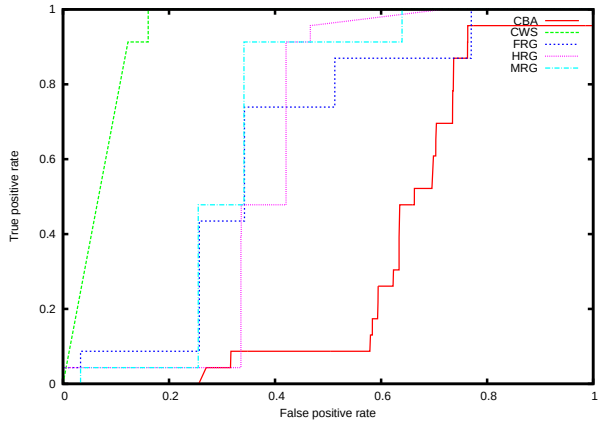


Figure 2: ROC curves for the Enron network.

respect to the other models. On the other hand, for the US road networks we observe a completely different pattern of results. This time, the FRG model performs worse than the CWS model or the two variants of ERG. The reason is related to the lattice-shaped regularity exhibited by such graphs. This is not surprising, as the WS model subsumes regular lattices as special case scenarios. A natural explanation for the superiority of the CWS model on the road networks lies in the fact that their parametric assumption concerning the observed degree distribution happens to be satisfied for such data, while it is clearly violated by the scale-free networks, which is exactly the opposite of what happens for the CBA model. It is worth noting, however, that FRGs are again performing significantly better than the CBA model. Therefore, FRGs appear to be a more stable class of conditional random network models, since their predictions ensure a regularly accurate behavior across networks displaying significantly different statistical properties. Overall, the presented results encourage the hypothesis that a genuinely nonparametric approach to conditional random graph modeling can offer dramatic advantages over parametric approaches.



(a)



(b)

Figure 3: ROC curves for the RoadCA (a) and RoadTX (b) networks.

6 CONCLUSION

The work presented in this paper started from two motivations. On the one hand, we remarked that statistical modeling of networks cries for nonparametric estimation, because of the inaccuracy often resulting from fallacious parametric assumptions. In this respect, we showed that statistics derived from the Laplacian spectrum of graphs (such as the Fiedler delta function) offer practical ways of developing nonparametric estimators. On the other hand, we suggested that a conditional approach to random graph modeling can be very effective in the large-scale setting, since it allows e.g. to dispense with the intractable partition functions often involved in joint distributions. With respect to this point, we showed how FRGs allow us to tackle very large datasets, processing effectively even millions of queries over networks with millions of nodes and edges. Interesting options for future work consist in analyzing the effect of the subgraph sampling mechanism in determining the distribution of the Fiedler delta, as well as investigating alternative ways of exploiting the information enclosed in the graph spectrum.

Acknowledgements

This work has been supported by the French National Research Agency (ANR-09-EMER-007). The authors are grateful to Romaric Gaudel, Rémi Gilleron, and Michal Valko for their suggestions and comments concerning this work.

References

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

William N. Anderson and Thomas D. Morley. Eigenvalues of the Laplacian of a graph. *Linear and Multilinear Algebra*, 18:141–145, 1985.

Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst, editors. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia (PA), 2000.

A.L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientic collaborations. *Physica A*, 311:590–614, 2002.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

A. Barrat and M. Weigt. On the properties of small-world network models. *The European Physical Journal B*, 13:547–560, 2000.

Julian Besag. Spatial Interaction and the Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society. Series B*, 36:192–236, 1974.

Dragoš M. Cvetković, Michael Doob, and Horst Sachs, editors. *Spectra of Graphs: Theory and Application*. Academic Press, New York (NY), 1979.

V.A. Epanechnikov. Nonparametric Estimation of a Multidimensional Probability Density. *Theory of Probability and its Applications*, 14:153–158, 1969.

P. Erdős and A. Rényi. On Random Graphs, I. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

Tom Fawcett. An introduction to ROC analaysis. *Pattern Recognition Letters*, 27:861–874, 2006.

Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

Ove Frank and David Strauss. Markov Graphs. *Journal of the American Statistical Association*, 81:832–842, 1986.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge (MA), 2009.

Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *Transactions on Knowledge Discovery from Data*, 1(1), 2007.

Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. *Internet Mathematics*, 6(1):29–123, 2009.

Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11: 985–1042, 2010.

Bojan Mohar. The Laplacian Spectrum of Graphs. In Y. Alavi, G. Chartrand, O. R. Oellermann, and A. J. Schwenk, editors, *Graph Theory, Combinatorics, and Applications*, pages 871–898. Wiley, 1991.

Mark E.J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64:025102, 2001.

Mark E.J. Newman. *Networks. An Introduction*. Oxford University Press, New York (NY), 2010.

Brian J. Parker, Simon Günter, and Justin Bedo. Stratification bias in low signal microarray studies. *BMC Bioinformatics*, 8, 2007.

Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

Garry L. Robins, Tom Snijders, Peng Wang, Mark S. Handcock, and Philippa E. Pattison. Recent developments in exponential random graph (p^*) models for social networks. *Social Networks*, 29:192–215, 2007.

Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

Tom A. B. Snijders, Philippa E. Pattison, Garry L. Robins, and Mark S. Handcock. New Specifications for Exponential Random Graph Models. *Sociological Methodology*, 36:99–153, 2006.

Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393: 440–442, 1998.

Mechanism Design for Cost Optimal PAC Learning in the Presence of Strategic Noisy Annotators

Dinesh Garg
IBM India Research Lab,
New Delhi, 110070 India
garg.dinesh@in.ibm.com

Sourangshu Bhattacharya
Yahoo! Labs,
Bangalore, 560071 India
sourangb@yahoo-inc.com

S. Sundararajan
Yahoo! Labs,
Bangalore, 560071 India
ssrajan@yahoo-inc.com

Shirish Shevade
Dept. of CSA, IISc
Bangalore, 560012 India
shirish@csa.iisc.ernet.in

Abstract

We consider the problem of Probably Approximate Correct (PAC) learning of a binary classifier from noisy labeled examples acquired from multiple annotators (each characterized by a respective classification noise rate). First, we consider the complete information scenario, where the learner knows the noise rates of all the annotators. For this scenario, we derive sample complexity bound for the Minimum Disagreement Algorithm (MDA) on the number of labeled examples to be obtained from each annotator. Next, we consider the incomplete information scenario, where each annotator is strategic and holds the respective noise rate as a private information. For this scenario, we design a cost optimal procurement auction mechanism along the lines of Myerson's optimal auction design framework in a non-trivial manner. This mechanism satisfies incentive compatibility property, thereby facilitating the learner to elicit true noise rates of all the annotators.

1 Background

In supervised learning, it is usually assumed that the training set is sampled i.i.d. from some fixed distribution, and the true labels are readily available. In contrast to this, there are many real-world applications in which obtaining true labels is a time consuming or costly process. However, for such applications, acquiring non-expert labels is easy, fast, and inexpensive. Web based *crowdsourcing* [Howe, 2008] platforms like *Rent-A-Coder* and *Galaxy Zoo* allow any web-user to perform various data annotation tasks. Amazon's Mechanical Turk allow any individual to publish a crowdsourcing task. In such cases, labels obtained from such sources are typically noisy, as the annotators can be careless or even deceitful. Further, the annotators can act strategically if it helps them fetch better rewards.

The problem of learning with noisy labeled examples has been studied mostly in two different contexts: (1) studying the learnability of the problem and developing learning algorithms under the PAC learning framework [Valiant, 1984], [Angluin and Laird, 1988], [Aslam and Decatur, 1996], [Blum et al., 1994], [Decatur and Gennaro, 1995], [Decatur, 1997], [Kearns, 1993], [Littlestone, 1991], and (2) estimating the noise rates of the annotators and building robust classifier models (independent or joint estimation and learning) [Dawid and Skene, 1979], [Raykar et al., 2009], [Yan et al., 2010], [Donmez et al., 2010]. In the former context, most of the work has been done in a single noisy annotator scenario with different kinds of noise models including *Malicious Noise* [Valiant, 1985], [Kearns and Li, 1993], [Goldman and Sloan, 1995] and *Nasty Noise* [Bshouty et al., 2002]. In the latter context, the focus has been mainly using different classifier models and presenting different noise rate estimation techniques. Our work differs from these works in the following way: (1) we extend PAC learning sample complexity bound

results to the multiple noisy annotators scenario as this problem is becoming more prevalent in recent times [Dekel and Shamir, 2009a], [Dekel and Shamir, 2009b], and (2) we design an optimal auction mechanism that facilitates eliciting (instead of estimating) true noise rates from the annotators followed by a cost-effective purchase of labeled examples satisfying the PAC learning constraint. We assume that annotators know their true noise rate. This is practical in many scenarios, e.g. many litigation service providing companies outsource document labeling tasks to paralegal agencies who in turn hire human editors. These agencies know the competence level of the editors through long standing relationship. The agencies bid for securing outsourcing contract while fully knowing the quality of its editors. The closest work in this direction is due to [Dekel et al., 2008] where they have focused on regression problem and took the elicitation approach via providing incentives to know the distribution information (privately held by the agents for evaluation). In contrast, our work is classification focused; furthermore, unlike the noisy annotators who charge labeling cost in our scenario, the strategic agents do not charge any price for annotation in their case. Instead, in their model, the agents have other vested interest in influencing the outcome of the learning; similar framework has also been used in [Meir et al., 2008], [Meir et al., 2009], [Dalvi et al., 2004], [L'Huillier et al., 2009], and [Kantarcioglu et al., 2008]. The main contributions of our work are:

- We consider the problem of PAC learning a binary classifier using noisy labeled examples obtained from multiple noisy annotators (in contrast to the conventional single noisy annotator scenario). We introduce the notion of annotation plan, and derive sample complexity bounds for PAC learning of finite concept class using the well-known minimum disagreement algorithm (MDA), in the known noise rates information scenario.
- We present an optimal auction mechanism to purchase labeled examples from strategic noisy annotators by eliciting true noise rate information in a more realistic scenario of unknown noise rates. Our approach is inspired by Myerson's Nobel prize winning work on optimal auction design [Myerson, 1981]. We derive allocation and payment rules for purchasing labeled examples at a near-optimal cost from strategic noisy annotators, satisfying the PAC constraint. To the best of our knowledge there has been no prior art with such results.

2 PAC Learning and Bounds

In this section, we provide basic definitions related to the PAC learning model [Valiant, 1984], [Angluin and Laird, 1988] with n noisy annotators, and derive sample complexity bounds for PAC learning with n noisy annotators. The PAC learning model comprises of an *instance space* $\mathscr{X}$ and a *concept class* $\mathscr{C}$. The instance space $\mathscr{X}$ is a fixed set which can be finite, countably infinite, $\{0,1\}^d$, or $\mathbb{R}^d$ for some $d \geq 1$. The concept class $\mathscr{C}$ is a set of *concepts*. A concept $\mathbf{c}$ is a subset of $\mathscr{X}$, which can equivalently be expressed as a boolean function from $\mathscr{X}$ to $\{0,1\}$, and it should be clear from the context whether $\mathbf{c}$ is referring to a subset or to a function. The task of the learner is to determine a close approximation to an unknown target (or true) concept $\mathbf{c}_t$, from the labeled examples. We assume that $\mathbf{c}_t \in \mathscr{C}$. The learner has access to n noisy annotators as the sources of its training data. Each call to an annotator returns a labeled example $\langle x, y \rangle$, where instance $x \in \mathscr{X}$ is drawn randomly and independently according to some unknown (to the learner) sampling distribution D. The learner gets $m_i \geq 0$ labeled examples from annotator i, where $i = 1, \ldots, n$, which together constitute the training dataset. Finally, the learner employs a learning algorithm to output a hypothesis $h \in \mathscr{C}$, based on the training data. The annotator $i, (i = 1, \ldots, n)$ reports the label y which is subject to an independent random mistake with a known probability η_i. So, the reported label is $y = \neg\, \mathbf{c}_t(x)$ with probability η_i and $y = \mathbf{c}_t(x)$ with probability $(1 - \eta_i)$. This noise model is known as *random classification noise* and was first studied by [Angluin and Laird, 1988] and [Laird, 1988] for the single noisy annotator case. The probability $\eta_i, i = 1, \ldots, n$ is known as *noise rate* of the annotator i. In this paper, we assume that $0 < \eta_i < 1/2, i = 1, \ldots, n$.

2.1 PAC learning with n noisy annotators

For any hypothesis $h \in \mathscr{C}$, the error rate (or generalization error) is defined to be the probability that $h(x) \neq \mathbf{c}_t(x)$ for an instance $x \in \mathscr{X}$ that is randomly drawn according to D. The error rate of a hypothesis h is given by $\mathbb{P}^D(\mathbf{c}_t \Delta h)$, where $\mathbf{c}_t \Delta h \subseteq \mathscr{X}$ is the symmetric difference between sets $\mathbf{c}_t$ and h, and $\mathbb{P}^D(\cdot)$ is the probability of this event (calculated with respect to D). A hypothesis h is said to be ϵ-bad if its error rate is more than ϵ, i.e. $\mathbb{P}^D(\mathbf{c}_t \Delta h) > \epsilon$. In the classical PAC model of [Valiant, 1984], the learner's goal is to come up with a learning algorithm which outputs an ϵ-bad hypothesis h with probability at most δ, where the probability is defined with respect to the distribution of training examples of a fixed size. Such a learning algorithm is known as PAC learning algorithm. In general, the error rate of the hypothesis chosen by a learning algorithm critically depends on the number of training examples supplied to the algorithm. Thus, a learning algorithm with single annotator is said to satisfy PAC bound with respect to the *sample size* $m(\epsilon, \delta)$ if the following condition holds true: $\mathbb{P}^{m(\epsilon,\delta)}(\mathbb{P}^D(\mathbf{c}_t \Delta h) > \epsilon) < \delta$, where h is the hypothesis output by the learning algorithm when trained on the $m(\epsilon, \delta)$ number of training examples. The sample size $m(\epsilon, \delta)$ is a non-negative integer valued function of the parameters ϵ and δ. The probability $\mathbb{P}^{m(\epsilon,\delta)}(\cdot)$ is taken over the distribution of $m(\epsilon, \delta)$ training examples (noisy or non-noisy). For a given algorithm, the smallest sample size $m^*(\epsilon, \delta)$ for which it still satisfies PAC bound is known as its *sample complexity*. Now, we extend the PAC learning framework to the case of n noisy annotators; starting with the following definitions:

- An **instance** of the PAC learning problem is a set of specifications of instance space $\mathscr{X}$, concept class $\mathscr{C}$, true concept $\mathbf{c}_t$, and sampling distribution D.

- An **annotation plan**, denoted by $\mathbf{m}(\epsilon, \delta) = (m_1(\epsilon, \delta), \ldots, m_n(\epsilon, \delta))$, is a vector of sample sizes (number of examples) annotated by the n annotators. This quantity is analogous to sample size $m(\epsilon, \delta)$ in the single annotator case. In rest of the paper, we use m_i and $m_i(\epsilon, \delta), i = 1, 2, \ldots, n$, interchangeably.

- A learning algorithm for n noisy annotators is said to satisfy **PAC bound** for annotation plan $\mathbf{m} = (m_1, \ldots, m_n)$ if following holds true:

$$\mathbb{P}^{(m_1,\ldots,m_n)}(\mathbb{P}^D(\mathbf{c}_t \Delta h) > \epsilon) < \delta \tag{1}$$

Note that the noise rates $\eta_1, \ldots, \eta_n$ of the annotators could be very different. Hence the PAC bound depends not just on $\sum_{i=1}^{n} m_i$, but on the individual numbers $m_1, \ldots, m_n$ also. This motivates us to define the notions of **feasible and infeasible annotation plans**, as:

- For a given learning algorithm, an annotation plan $\mathbf{m} = (m_1, \ldots, m_n)$ is said to be *feasible* if the learning algorithm satisfies PAC bound (1) for every instance of the problem when training data is supplied as per this plan.

- Given an algorithm, an annotation plan $\mathbf{m} = (m_1, \ldots, m_n)$ is said to be *infeasible* if the algorithm **fails** to satisfy PAC bound (1) for at least one instance of the problem when training data is supplied as per this plan.

2.2 Feasible annotation plans for MDA

In this section, we consider a simple learning algorithm, namely *Minimum Disagreement Algorithm (MDA)* and derive PAC learnability bound on annotation plan complexity for this algorithm in the presence of n noisy annotators. [Laird, 1988] analyzed this algorithm for single noisy annotator case. MDA outputs the hypothesis h, which minimizes the empirical loss, $L_e(h)$, on the training dataset. We describe MDA for multiple annotators, below.

Algorithm 1 (MDA) *Let* $\mathscr{D} = \{\langle x_j^i, y_j^i \rangle \;\; i = 1, \ldots, n; \; j = 1, \ldots, m_i\}$ *be the input training data, where* $\langle x_j^i, y_j^i \rangle$ *is supplied by annotator* i *in* j^{th} *call. The empirical loss* $L_e(h)$ *for hypothesis* h *is given as:*

$$L_e(h) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathbf{1}(h(x_j^i) \neq y_j^i) \tag{2}$$

where $\mathbf{1}(\cdot)$ *is an indicator variable. Output hypothesis* $h^* \in \mathscr{C}$*, such that* $L_e(h^*) \leq L_e(h), \forall h \in \mathscr{C}$ *(use any tie breaking rule).*

Next we derive a characterization of feasible annotation plans for MDA. For this, we define a few events and their corresponding probabilities, assuming a finite concept class $\mathscr{C}$ having $|\mathscr{C}| = N < \infty$. The events are defined for an annotation plan $(m_1, \ldots, m_n)$ and a hypothesis h. We assume that m_i samples of x^i are drawn randomly and independently, according to the distribution D by annotator i, and labels y^i are flipped independently with noise rates η_i. The events E_1, E_2, E_3, and E_4, of our interest are defined as:

- $E_1(h, m_1, \ldots, m_n)$: The empirical error of a given hypothesis $h \in \mathscr{C}$ is no more than the empirical error of the true hypothesis $\mathbf{c}_t$, i.e. $L_e(h) \leq L_e(\mathbf{c}_t)$.
- $E_2(h, m_1, \ldots, m_n)$: The empirical error of a given hypothesis $h \in \mathscr{C}$ is the minimum across all hypotheses in the class $\mathscr{C}$, i.e. $L_e(h) \leq L_e(h')\ \forall h' \in \mathscr{C}$.
- $E_3(h, m_1, \ldots, m_n)$: MDA outputs a given hypothesis h.
- $E_4(\epsilon, m_1, \ldots, m_n)$: MDA outputs an ϵ-bad hypothesis.

The probabilities of events $E_i, i = 1, 2, 3$ and E_4 are denoted by $\mathbb{P}^{(m_1,\ldots,m_n)}[E_i(h)]$ and $\mathbb{P}^{(m_1,\ldots,m_n)}[E_4(\epsilon)]$, respectively. Next, we show the following useful lemmas:

Lemma 1 *Given a concept class $\mathscr{C}$ such that $\mathbf{c}_t \in \mathscr{C}$ and $N = |\mathscr{C}|$, the following holds true for any given annotation plan $(m_1, \ldots, m_n)$:*

$$\mathbb{P}^{(m_1,\ldots,m_n)}[E_4(\epsilon)] \leq (N-1)\times \left[\max_{h \in \mathscr{C}, h \text{ is } \epsilon\text{-bad}} \mathbb{P}^{(m_1,\ldots,m_n)}[E_1(h)] \right]$$

Proof: By definition of the events, for any hypothesis $h \in \mathscr{C}$ that is ϵ-bad (for any $\epsilon > 0$), we have $E_3(h, m_1, \ldots, m_n) \subseteq E_2(h, m_1, \ldots, m_n) \subseteq E_1(h, m_1, \ldots, m_n)$. Also, $E_4(\epsilon, m_1, \ldots, m_n) = \bigcup_{h \in \mathscr{C}; h \text{ is } \epsilon\text{-bad}} E_3(h, m_1, \ldots, m_n)$. The lemma follows from taking probabilities of these events. *Q.E.D.*

Now, we state our main result regarding characterization of the feasible annotation plans for MDA.

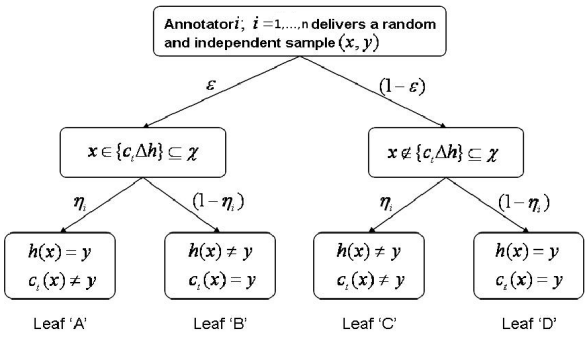


Figure 1: Probability Tree for $\mathbf{c}_t$ and h.

Theorem 1 *Consider the PAC learning model with n noisy annotators and the MDA (Algorithm 1). Let $N = |\mathscr{C}| < \infty$. Then, for any given $0 < \epsilon, \delta < 1$ and $0 < \eta_i < 1/3, i = 1, \ldots, n$, if $m_i, i = 1, \ldots, n$, satisfy the following inequality then the MDA will satisfy PAC bound.*

$$\log(N/\delta) \leq \sum_{i=1}^{n} m_i \psi(\eta_i) \tag{3}$$

where $\psi(\eta_i) = \log\left[1 - \epsilon\left(1 - \exp\left(-(1-3\eta_i)/8\right)\right)\right]^{-1}$ for $i = 1, \ldots, n$

Remark: The inequality (3) characterizes a subset of the feasible annotation plans. This characterization is independent of the problem instance and the tie breaking rule of the MDA.

Proof: MDA satisfies PAC bound iff there exists an annotation plan $(m_1, \ldots, m_n)$ such that $\mathbb{P}^{(m_1,\ldots,m_n)}[E_4(\epsilon)] < \delta$. From Lemma 1, it can be seen that for any $0 < \epsilon, \delta < 1$, if an annotation plan $(m_1, \ldots, m_n)$ satisfies the following condition, then MDA will satisfy PAC bound.

$$\left[\max_{h \text{ is } \epsilon\text{-bad}} \mathbb{P}^{(m_1,\ldots,m_n)}[E_1(h)] \right] \leq \delta/N \tag{4}$$

Notice a change in the LHS expression as compared to Lemma 1. The LHS in the above expression is an upper bound for the RHS of the expression in Lemma 1 (excluding $N-1$), because here h may not belong to $\mathscr{C}$. This makes the bound independent of the problem instance, although MDA will only output $h \in \mathscr{C}$. Now, we upper bound the LHS of (4). To do this, we derive an upper bound for $\mathbb{P}^{(m_1,\ldots,m_n)}[E_1(h)]$ when hypothesis h has an error rate of ϵ (for any $\epsilon \in (0, 1)$).

To derive this bound, we note that for any random and independent sample (x, y) that is delivered by

an annotator i, the probability of its agreeing (or disagreeing) with hypotheses $\mathbf{c}_t$ and h (having error rate ϵ) is given by a probability tree shown in Figure 1. From the tree, it can be seen that $\mathbb{P}^{(m_1,\ldots,m_n)}[E_1(h)]$ is same as the probability that the number of samples that fall under leaf B in the probability tree is at most the number of samples that fall under leaf A.

To compute the above quantity, first we compute the conditional probability, $\mathbb{P}^{k_1,\ldots,k_n}(L_e(h) \leq L_e(\mathbf{c}_t))$, defined as: if k_i examples, $(0 \leq k_i \leq m_i)$, from annotator i $(i = 1,\ldots,n)$ come from the set $(\mathbf{c}_t \Delta h)$, then the probability that empirical error of h (given by $L_e(h)$) is less than or equal to empirical error of $\mathbf{c}_t$ (given by $L_e(\mathbf{c}_t)$).

Consider the random variable $Z_i^j, i = 1,\ldots,n$ $j = 1,\ldots,k_i$, which is the indicator of whether j^{th} sample from i^{th} annotator is from leaf node B, given that all the data points are from $\mathbf{c}_t \Delta h$ region. Thus, $\mathbb{P}(Z_i^j = 1) = (1-\eta_i)$ and $\mathbb{P}(Z_i^j = 0) = \eta_i$. Let $Z = \sum_{i=1}^{n}\sum_{j=1}^{k_i} Z_i^j$. Then the event $L_e(h) \leq L_e(\mathbf{c}_t)$ is same as the event $Z \leq \sum_{i=1}^{n} k_i/2$. Hence, we are interested in finding an upper bound on $\mathbb{P}(Z \leq \sum_{i=1}^{n} k_i/2)$. We can use the multiplicative form of Chernoff bound (see e.g. Theorem 4.2 in [Motwani and Raghavan, 1995]), which says $\mathbb{P}[Z \leq (1-\nu)\mu] \leq \exp(-\mu\nu^2/2)$, where $\mu = E[Z] = \sum_{i=1}^{n}(1-\eta_i)k_i$. Hence, by letting $\nu = \frac{\sum_{i=1}^{n} k_i(1-2\eta_i)}{2\sum_{i=1}^{n} k_i(1-\eta_i)}$, we get the following bound:

$$\mathbb{P}^{(k_1,\ldots,k_n)}(L_e(h) \leq L_e(\mathbf{c}_t)) \leq e^{\frac{-(\sum_{i=1}^{n} k_i(1-2\eta_i))^2}{8\sum_{i=1}^{n} k_i(1-\eta_i)}}$$

Simplifying this bound, for $0 \leq \eta_i \leq 1/3$ we get:

$$\mathbb{P}^{(k_1,\ldots,k_n)}(L_e(h) \leq L_e(\mathbf{c}_t)) \leq e^{\frac{-\sum_{i=1}^{n} k_i(1-3\eta_i)}{8}} \quad (5)$$

Summing up the above conditional probability bound over all possible values of k_i, the total probability $\mathbb{P}^{(m_1,\ldots,m_n)}[E_1(h)]$ becomes:

$$\sum_{k_1=0}^{m_1}\cdots\sum_{k_n=0}^{m_n}\left(\prod_{i=1}^{n}\left(\binom{m_i}{k_i}\epsilon^{k_i}(1-\epsilon)^{m_i-k_i}\right)\right. \quad (6)$$
$$\left.\mathbb{P}^{(k_1,\ldots,k_n)}(L_e(h) \leq L_e(\mathbf{c}_t))\right)$$

Using the bound in (5), we get the following upper bound on $\mathbb{P}^{(m_1,\ldots,m_n)}[E_1(h)]$:

$$\prod_{i=1}^{n}\left(\sum_{k_i=0}^{m_i}\binom{m_i}{k_i}\epsilon^{k_i}(1-\epsilon)^{m_i-k_i}\exp\left(-k_i(1-3\eta_i)/8\right)\right)$$

Using the moment generating function of the Binomial distribution, the bound becomes

$$\prod_{i=1}^{n}\left[1-\epsilon(1-\exp(-(1-3\eta_i)/8))\right]^{m_i}$$

In above bound, h has an error rate exactly equal to ϵ. However, this bound is valid for an ϵ-bad hypothesis also because the expression decrease as ϵ increases. Substituting this upper bound on the LHS of (4), we get the desired claim. *Q.E.D.*

Note that the Theorem 1 is valid only for the range of $0 < \eta_i < 1/3$. However, we can extend the definition of $\psi(\cdot)$ to the boundary points in a manner that the same relation (3) holds true. For this, observe that minimum number of examples required from a single **non-noisy** annotator would be $m_0 = \log(N/\delta)/\log\left[1-\epsilon\right]^{-1}$. This is because in such a case, we have $\mathbb{P}(L_e(h) \leq L_e(\mathbf{c}_t) \mid x \in \mathbf{c}_t\Delta h) = 0$ and $\mathbb{P}(x \notin \mathbf{c}_t\Delta h) = (1-\epsilon)$. Hence, we can let $\psi(0) = \log\left[1-\epsilon\right]^{-1}$. Also, we let $\psi(1/3) = \log\left[1-\epsilon\left(1-\exp(-(1/18))\right)\right]^{-1}$ and $m_{1/3} = \log(N/\delta)/\psi(1/3)$ from [Laird, 1988].

3 Cost Optimal Mechanism Design for PAC Learning

We consider the problem of procuring a feasible annotation plan when the learner needs to pay annotators for their efforts, under known and unknown noise rate scenarios. In the unknown noise rate scenario, we propose an auction model and present an optimal auction mechanism.

We assume that each annotator i (with noise rate η_i) incurs an internal cost $c(\eta_i)$ of annotation for labeling one data point; note that the cost is dependent on the noise rate, and the cost function is same for all the annotators. The cost function is assumed to be a bounded, continuously differentiable, strictly decreasing function in $0 \leq \eta_i < 1/2\ \forall\ i = 1,\ldots,n$. If an annotator is more competent (i.e. less noisy) then he can make more money by selling his services and time to somewhere else in the market, which translates to saying that his internal cost of annotation is high.

Consider a simplistic scenario of complete information where the learner knows noise rates of all the annotators. In such a case, the goal of the learner is purchase an annotation plan $m =$

$(m_1, m_2, \ldots, m_n)$ in such a way that the procurement cost (that is, cost of annotation), given by $\sum_{i=1}^{n} m_i c(\eta_i)$, is minimized subject to the PAC learning constraint (3). This is an integer linear programming problem. An approximate solution can be obtained by relaxing the integer constraint and rounding off the optimal solution to the nearest integer value.

Now, let us consider a more realistic scenario of incomplete information where the learner does not know noise rates $\eta = (\eta_1, \ldots, \eta_n)$. There are two possible approaches: (1) estimation, and (2) elicitation. In the estimation approach, the learner estimates η_i using previously acquired examples (say, for example, comparing labels from different annotators). In the elicitation approach, the learner gets η_i directly from the annotators. The former approach has the disadvantage that poor estimates result in either paying more (when overestimated) or not satisfying the PAC bound (when underestimated). Due to this reason, we are interested in elicitation. In this approach, the learner pays an incentive (*a.k.a.* price of information) to get η_i from the annotators. Note that the learner needs to pay this price of information to elicit true noise rates. (Otherwise, the annotators can falsely report the noise rate.) For this purpose, we propose to design a procurement auction mechanism to procure a feasible annotation plan with minimum cost; now, the procurement cost also includes the price of information. This problem is challenging because from annotator's perspective, he would like to maximize his utility (i.e., the payment received minus the internal cost for annotation). The choice of mechanism depends crucially on various design parameters such as $N, \epsilon, \delta, \eta_i, c(\cdot)$, and the choice of the learning algorithm. We assume that $N, \epsilon, \delta, c(\cdot)$, and the choice of the learning algorithm are public knowledge, and only η_i is the private information of i^{th} annotator.

3.1 Procurement Auction Model

The learner solicits simultaneous and confidential bids for the noise rates from annotators. Let $\hat{\eta}_i$ be the bid of i^{th} annotator that can possibly be a false noise rate. Assume that annotator i draws his true noise rate η_i in an independent random manner using a density function ϕ_i in the interval $I_i = [0, 1/3]$ with the corresponding cumulative distribution function (Φ_i), and let $\phi_i(\eta_i) > 0$ for all $\eta_i \in I_i$ and $i = 1, 2, \ldots, n$. Let $I = I_1 \times I_2 \times \ldots \times I_n$ and $\phi = \phi_1 \times \phi_2 \ldots \times \phi_n$ denote respective joint spaces. We use the subscript $-i$ to exclude i^{th} annotator in any variable (e.g. I_{-i}, η_{-i}) and, we also use the notation $\hat{\eta} = (\hat{\eta}_i, \hat{\eta}_{-i})$.

After receiving the bids (i.e. $\hat{\eta} = (\hat{\eta}_1, \ldots, \hat{\eta}_n)$), the learner allocates a contract of supplying certain number of labeled examples to each annotator and an associated payment. Thus, a procurement auction mechanism is a pair of mappings $\mathcal{M} = (a, p)$, where $a : I \mapsto \mathbb{N}_0^n$ [1] is the **allocation rule** and $p : I \mapsto \mathbb{R}^n$ is the **payment rule**.

Given an auction mechanism $\mathcal{M} = (a, p)$, an annotator i, having noise rate η_i, gets the following utility when all the annotators report their bids $\hat{\eta}$:

$$u_i(\hat{\eta}; \eta_i) = p_i(\hat{\eta}) - a_i(\hat{\eta}) c(\eta_i) \tag{7}$$

Note that the first and the second term denote the payment received from the learner and the internal cost in supplying the labeled examples, respectively. Since each annotator i does not know η_{-i} and moreover, others' bids $\hat{\eta}_{-i}$ affect his utility, it is useful to define **expected allocation rule** α and the **expected payment rule** π for any mechanism $\mathcal{M} = (a, p)$ in the following manner (from i^{th} annotator's perspective).

$$\alpha_i(\hat{\eta}_i) = \int_{I_{-i}} a_i(\hat{\eta}_i, \hat{\eta}_{-i}) \phi_{-i}(\hat{\eta}_{-i}) d\hat{\eta}_{-i} \tag{8}$$

$$\pi_i(\hat{\eta}_i) = \int_{I_{-i}} p_i(\hat{\eta}_i, \hat{\eta}_{-i}) \phi_{-i}(\hat{\eta}_{-i}) d\hat{\eta}_{-i} \tag{9}$$

The **expected utility** of annotator i, when he bids $\hat{\eta}_i$ while having true value η_i, can now be given by

$$U_i(\hat{\eta}_i; \eta_i) = \pi_i(\hat{\eta}_i) - \alpha_i(\hat{\eta}_i) c(\eta_i) \tag{10}$$

When both arguments in (10) are same, we use $U_i(\eta_i)$ to mean $U_i(\eta_i; \eta_i)$ (for notational simplicity). Given this background, we first present several definitions that are essential to prove our results. A Mechanism $\mathcal{M} = (a, p)$ is said to be:

[1] The symbol $\mathbb{N}_0$ denotes the set of natural numbers inclusive of zero. The allocation and the payment rules are functions of $N, \epsilon, \delta, c(\cdot)$, and the algorithm. For notational simplicity, we drop these parameters.

- **Dominant Strategy Incentive Compatible** (DSIC) if for every annotator i and for every possible true noise rate $\eta_i \in I_i$, the utility $u(\cdot)$ is maximized when $\hat{\eta}_i = \eta_i$ irrespective of what others are bidding, i.e., $u_i(\eta_i, \hat{\eta}_{-i}; \eta_i) \geq u_i(\hat{\eta}_i, \hat{\eta}_{-i}; \eta_i)\ \forall\ \hat{\eta}_i \in I_i, \hat{\eta}_{-i} \in I_{-i}$.
- **Bayesian Incentive Compatible** (BIC) if for every annotator i and for every possible true noise rate $\eta_i \in I_i$, the expected utility $U_i(\cdot)$ is maximized when $\hat{\eta}_i = \eta_i$, i.e., $U_i(\eta_i) \geq U_i(\hat{\eta}_i; \eta_i)\ \ \forall \hat{\eta}_i \in I_i$. Note, any mechanism satisfying DSIC will also satisfy BIC but the other way is not necessarily true.
- **PAC compatible** if the annotation plan procured by this mechanism satisfies the PAC bound condition (3) whenever all the annotators report their true noise rates, i.e., $\log(N/\delta) \leq \sum_{i=1}^{n} a_i(\eta)\psi(\eta_i)$.
- **Individually Rational** (IR) if no annotator loses (in expected sense) anything by reporting true noise rates, i.e., $\pi_i(\eta_i) - \alpha_i(\eta_i)c(\eta_i) \geq 0\ \forall\ \eta_i \in I_i$.

Our goal is to design a procurement auction mechanism that satisfies BIC, PAC Compatibility, and IR properties; and minimizes the expected cost of procurement for the learner. We call such an auction mechanism as *Optimal Auction for Data Labeling*. Our design is inspired from the Nobel Prize winning work of Roger Myerson on optimal auction design [Myerson, 1981]. For a comprehensive treatment of this topic, readers are referred to [Krishna, 2002] and [Mishra, 2008].

3.2 Characterization of Incentive Compatibility

To design an optimal auction mechanism for data labeling problem, we need to first characterize the space of auction rules that satisfy BIC property. For this, we begin with the following definitions. An allocation rule a is said to be:

- **weakly monotone (WM)** if for every annotator i and for every $\hat{\eta}_{-i} \in I_{-i}$, we have $a_i(\eta_i, \hat{\eta}_{-i}) \geq a_i(\hat{\eta}_i, \hat{\eta}_{-i})$ for all $\eta_i, \hat{\eta}_i \in I_i$, with $\eta_i > \hat{\eta}_i$.
- **weakly monotone in expectation (WME)** if for every annotator i and for every $\eta_i, \hat{\eta}_i \in I_i$ with $\eta_i > \hat{\eta}_i$, we have $\alpha_i(\eta_i) \geq \alpha_i(\hat{\eta}_i)$.

[Myerson, 1981] showed that BIC is characterized by the WME allocation rules in the setting of single object auction. Interestingly, a similar characterization holds in our problem setting also. We state this result as an important theorem.

Theorem 2 *A Mechanism $\mathcal{M} = (a, p)$ is a BIC mechanism iff (i) the allocation rule $a(\cdot)$ is WME, and (ii) the expected payment rule $\pi(\cdot)$ satisfies:*

$$\pi_i(\eta_i) = \gamma_i + \alpha_i(\eta_i)c(\eta_i) - z_i(\eta_i) \tag{11}$$

where $\gamma_i = \pi_i(0) - \alpha_i(0)c(0)$ and $z_i(\eta_i) = \int_0^{\eta_i} \alpha_i(t_i)c'(t_i)dt_i$.

Proof: Suppose $\mathcal{M} = (a, p)$ is a BIC mechanism. Then, we show that the allocation rule is WME and (11) holds. Consider an annotator i and $\eta_i, \hat{\eta}_i \in I_i$ with $\eta_i > \hat{\eta}_i$. Then, it follows from (10) and BIC that $\pi_i(\eta_i) - \alpha_i(\eta_i)c(\eta_i) \geq \pi_i(\hat{\eta}_i) - \alpha_i(\hat{\eta}_i)c(\eta_i)$ and $\pi_i(\hat{\eta}_i) - \alpha_i(\hat{\eta}_i)c(\hat{\eta}_i) \geq \pi_i(\eta_i) - \alpha_i(\eta_i)c(\hat{\eta}_i)$. Adding these two inequalities, we get $[\alpha_i(\eta_i) - \alpha_i(\hat{\eta}_i)]\,[c(\hat{\eta}_i) - c(\eta_i)] \geq 0$. Since $\eta_i > \hat{\eta}_i$ and $c(\cdot)$ is a strictly decreasing function, this implies that $\alpha_i(\eta_i) \geq \alpha_i(\hat{\eta}_i)$ (i.e. α_i is non-decreasing). Now, since $\mathcal{M}$ is BIC, for every $\eta_i, \hat{\eta}_i \in I_i$, we have $U_i(\eta_i) \geq U_i(\hat{\eta}_i) - \alpha_i(\hat{\eta}_i)\,[c_i(\eta_i) - c_i(\hat{\eta}_i)]$. This is obtained from adding and subtracting $\alpha_i c(\hat{\eta}_i)$ to $U_i(\hat{\eta}_i; \eta_i)$ (Equation (10)) and rearranging the terms. Similarly, switching the roles of η_i and $\hat{\eta}_i$, we get $U_i(\hat{\eta}_i) \geq U_i(\eta_i) - \alpha_i(\eta_i)\,[c_i(\hat{\eta}_i) - c_i(\eta_i)]$. On combining these two inequalities, we get $-\alpha_i(\eta_i)\,[c_i(\hat{\eta}_i) - c_i(\eta_i)] \leq U_i(\hat{\eta}_i) - U_i(\eta_i) \leq -\alpha_i(\hat{\eta}_i)\,[c_i(\hat{\eta}_i) - c_i(\eta_i)]$. Now, by dividing with $(\hat{\eta}_i - \eta_i)$ and letting $\hat{\eta}_i \to \eta_i$, the two-sided inequality implies that $-\alpha_i(\cdot)c'(\cdot)$ is the derivative of $U_i(\cdot)$. Since $c'(\cdot)$ is continuous and $\alpha_i(\cdot)$ is non-decreasing, the function $\alpha_i(\cdot)c'(\cdot)$ has finitely many points of discontinuity and hence is Riemann integrable in the interval $[0, 1/3]$. Thus, we have $\int_0^{\eta_i} -\alpha_i(t_i)c'(t_i)dt_i = U_i(\eta_i) - U_i(0)$. On substituting $U_i(\eta_i) = \pi_i(\eta_i) - \alpha_i(\eta_i)c(\eta_i)$ and $U_i(0) = \pi_i(0) - \alpha_i(0)c(0)$, we get (11).

Now, suppose that a is WME and (11) holds. We will show that $\mathcal{M} = (a, p)$ is a BIC mechanism. For any annotator i and any $\eta_i, \hat{\eta}_i \in I_i$, we have $\pi_i(\eta_i) - \pi_i(\hat{\eta}_i) = \alpha_i(\eta_i)c(\eta_i) - \alpha_i(\hat{\eta}_i)c(\eta_i) +$

$\alpha_i(\hat{\eta_i})\left[c(\eta_i) - c(\hat{\eta_i})\right] - z_i(\eta_i) + z_i(\hat{\eta_i})$. Using the facts $\alpha_i(\cdot)$ is non-decreasing and $c'(\cdot) \leq 0$, it can be shown that $\alpha_i(\hat{\eta_i})\left[c(\eta_i) - c(\hat{\eta_i})\right] - z_i(\eta_i) + z_i(\hat{\eta_i}) \geq 0$. Therefore,$\pi_i(\eta_i) - \pi_i(\hat{\eta_i}) \geq \alpha_i(\eta_i)c(\eta_i) - \alpha_i(\hat{\eta_i})c(\eta_i)$ which is the condition for BIC. *Q.E.D*

A similar characterization result can be derived for the DSIC case also. Due to lack of space, we skip the results. Note that Theorem (2) suggests that the learner can only increase the contract size with higher noise rate. This is a bit counter intuitive as the learner is buying more examples from a more noisy annotator (in a relative sense). However, this is essentially the key to enforce truthful elicitation of the noise rates. Even if an annotator misreports higher noise rate to get a bigger size contract, the payment rule would make sure that the additional payment is not enough to cover the cost of labeling the required additional examples. A similar argument holds for the other direction as well.

3.3 Optimal Auction Mechanism

We pose the optimization problem of designing the auction mechanism as follows:

$$\min_{a(\cdot),p(\cdot)} \Pi(a,p) = \sum\nolimits_{i=1}^{n} \int_0^{1/3} \pi_i(t_i)\phi_i(t_i)dt_i \quad s.t. \quad (12)$$

$$\alpha_i(\cdot) \text{ is non-decreasing} \quad (13)$$

$$\pi_i(\eta_i) = \gamma_i + \alpha_i(\eta_i)c(\eta_i) - z_i(\eta_i)\ \forall \eta_i \in I_i, \forall i \quad (14)$$

$$\pi_i(\eta_i) \geq \alpha_i(\eta_i)c(\eta_i)\ \forall \eta_i \in I_i, \forall i \quad (15)$$

$$\log(N/\delta) \leq \sum\nolimits_i a_i(\eta_i, \eta_{-i})\psi(\eta_i)\ \forall (\eta_i, \eta_{-i}) \in I \quad (16)$$

Note that the objective function (12) constitutes the total expected payment made to all the annotators. The constraints (13) and (14) are BIC constraints, (15) is the IR constraint, and (16) is the PAC compatibility constraint. Recall, $c(\cdot)$ is a strictly decreasing function. If (14) is satisfied then (15) will be satisfied iff $\gamma_i \geq 0$ (i.e. $\pi_i(0) \geq \alpha_i(0)c(0))\ \forall i$. Because our goal is to minimize (12), we must set $\gamma_i = 0$. Then, by setting $\gamma_i = 0$ and using the definition of $\alpha_i(\cdot)$, we can rewrite the objective function after some algebraic manipulations as:

$$\Pi(a,p) = \int_I \left(\sum\nolimits_{i=1}^{n} v_i(x_i)a_i(x)\right)\phi(x)dx \quad (17)$$

where $v_i(\eta_i) = c(\eta_i) - \frac{1-\Phi_i(\eta_i)}{\phi_i(\eta_i)}c'(\eta_i)$ is called as **virtual cost function**. Note that since $c'(\eta_i)$ is negative, $\phi_i(\cdot) > 0$ for all i and for all $\eta_i \in I_i$, the virtual cost function is always non-negative, well defined and is higher than $c(\eta_i)$. Note that (17) is essentially a function of the allocation rule $a(\cdot)$ since $p(\cdot)$ is dictated by $a(\cdot)$ via (14). We need to minimize (17) subject to the constraints (13) and (16). It seems difficult to solve this problem, particularly with the constraint (13) without imposing additional regularity condition. To arrive at this condition, we consider solving (17) by ignoring the constraint (13) momentarily. So, we consider minimizing (17) subject to the constraint (16) alone for the moment. Note, for minimizing (17), it suffices to minimize $\sum_{i=1}^{n} v_i(\eta_i)a_i(\eta)$ for every possible profile η subject to the constraint (16). For a fixed η, this is an integer linear programming (ILP) problem whose approximate solution can be obtained by relaxing the integer constraint and rounding off the optimal solution to the nearest integer value. This approximate solution is a near-optimal way of purchasing examples from noisy annotators for PAC learning (ignoring (13)). By looking at the dual of such a relaxed LP, one can verify that in this near-optimal scheme, the learner should purchase $\lceil \log(N/\delta)/\psi(\eta_{i^*}) \rceil$ number of examples from only that annotator, say i^*, for whom the ratio $v_i(\eta_i)/\psi(\eta_i)$ is the minimum. Let us call this rule as the **minimum allocation rule** whose approximation guarantee is given below.

Theorem 3 *Let ALG be the total cost of purchase incurred by the min allocation rule. Let OPT be the optimal value of the ILP and m_0 be non-noisy sample complexity. Then, we must have*

$$ALG \leq OPT + v_{i^*}(\eta_{i^*}) \leq OPT(1 + 1/m_0) \quad (18)$$

Proof: The optimal solution of the linear relaxation is always a lower bound on the optimal solution of the ILP. Therefore, we must have $\log(N/\delta)v_{i^*}(\eta_{i^*})/\psi(\eta_{i^*}) \leq OPT$. This implies

$$\begin{aligned} ALG &= v_{i^*}(\eta_{i^*})\lceil \log(N/\delta)/\psi(\eta_{i^*}) \rceil \\ &\leq \log(N/\delta)v_{i^*}(\eta_{i^*})/\psi(\eta_{i^*}) + v_i(\eta_{i^*}) \\ &\leq OPT + v_i(\eta_{i^*}) \end{aligned} \quad (19)$$

To get the other bound, note that the number of examples suggested by the minimum allocation rule is at least as much as m_0. Therefore, we must have $\log(N/\delta)/\psi(\eta_{i^*}) \geq m_0$. Thereby, we get:

$$OPT \geq \log(N/\delta)v_{i^*}(\eta_{i^*})/\psi(\eta_{i^*}) \geq m_0\ v_i(\eta_{i^*}) \quad (20)$$

Substituting the bound on $v_i(\eta_{i^*})$ from (20) into (19) will give us the second term. *Q.E.D*

Regularity Condition: So far, we considered the approximate optimal auction mechanism design without the constraint (13). Therefore, the minimum allocation rule need not satisfy the WME property. However, it is WME under the regularity condition that $v_i(\cdot)/\psi(\cdot)$ *is a non-increasing function.* Under this condition, as η_i increases, the annotator i remains the winner if he/she is already the winner (with an increased contract size) or becomes the winner as per the minimum allocation rule. Therefore, the allocation rule satisfies the WM property (hence, WME). This implies that the allocation rule would give an approximate optimal mechanism satisfying BIC + IR+ PAC compatibility properties. For every (η_i, η_{-i}) and every i, the associated payment rule can be given by

$$p_i(\eta_i, \eta_{-i}) = a_i(\eta_i, \eta_{-i})c(\eta_i) - w_i(\eta_i) \qquad (21)$$

where $w_i(\eta_i) = \int_0^{\eta_i} a_i(t_i, \eta_{-i})c'(t_i)dt_i$. One can verify that the corresponding expected payment rule $\pi_i(\cdot)$ satisfies BIC and IR constraints. In fact, it turns out that the minimum allocation rule and the payment rule (21) together satisfy the DSIC property. We skip this proof as it follows the same line of arguments given in the proof of Theorem 2.

Simplified Payment Rule: We define for every annotator i, the smallest bid value sufficient to win the contract as per the minimum allocation rule as:

$$q_i(\eta_{-i}) = \inf\left\{\hat{\eta_i} \mid \frac{v_i(\eta_i)}{\psi(\eta_i)} \leq \frac{v_j(\eta_j)}{\psi(\eta_j)}\ \forall j \neq i\right\} \qquad (22)$$

Then, the minimum allocation rule and simplified payment rule can be written as:

$$a_i(\eta) = \begin{cases} \lceil \log(N/\delta)/\psi(\eta_i) \rceil & : \text{ if } \eta_i \geq q_i(\eta_{-i}) \\ 0 & : \text{ otherwise} \end{cases} \qquad (23)$$

$$p_i(\eta) = \begin{cases} \left\lceil \frac{\log(N/\delta)}{\psi(\eta_i)} \right\rceil c(q_i(\eta_{-i})) & : \text{ for winner} \\ 0 & : \text{ otherwise} \end{cases} \qquad (24)$$

Thus, we now have the following mechanism.

Algorithm 2 (Approx_Mechanism) *The learner should choose an annotator i^* for whom the score $v_i(\eta_i)/\psi(\eta_i)$ is minimum (breaking ties arbitrarily), and award a contract of supplying $\lceil \log(N/\delta)/\psi(\eta_{i^*}) \rceil$ labeled examples. The learner must pay an amount equal to $c(q_{i^*}(\eta_{-i^*}))$ per example to this annotator, where $q_{i^*}(\eta_{-i^*})$ denotes the smallest noise rate bidding which the winning annotator i^* still stays as the winner. The other annotators are not paid any amount.*

For the winning annotator i^*, we have $q_{i^*}(\eta_{-i^*}) \leq \eta_{i^*}$ (under the regularity condition). This implies that $c(q_{i^*}(\eta_{-i^*})) \geq c(\eta_{i^*})$. Note that the right hand side of this inequality is the cost involved when η is known. Therefore, the learner needs to pay some extra cost to annotators for eliciting the true noise rates. The following theorem is now apparent from the analysis done so far.

Theorem 4 *Suppose the regularity condition holds. Then,* **Approx_Mechanism** *is an approximate optimal mechanism satisfying DSIC, IR, and PAC compatibility properties. The approximation guarantee of this mechanism is given by $ALG \leq OPT + v_{i^*}(\eta_{i^*}) \leq OPT(1 + 1/m_0)$.*

Note, DSIC is a preferred condition than BIC and DSIC implies BIC. The above result says that under the regularity condition, a DSIC mechanism comes very close to the optimal BIC mechanism.

4 Conclusion

To the best of our knowledge, this is the first paper to model and analyze the problem of acquiring labeled examples from multiple noisy **strategic** annotators for PAC learning. For such a setting, we have proposed an approximate cost optimal auction mechanism for the unknown noise rates scenario, by extending Myerson's optimal auction design framework in a non-trivial manner. As future enhancements, (1) the assumption of finite concept class can be relaxed by making use of VC-dimension, (2) PAC bound can be derived for an improved Weighted MDA (WMDA) algorithm, where we give more importance to the samples from less noisy annotator while computing the loss $L_e(\cdot)$, and (3) one can design better approximate algorithms to solve the underlying ILP problems.

References

J. Howe. *Crowdsourcing: why the power of the crowd is driving the future of business*. Crown Business, 2008.

L.G. Valiant. A theory of learnable. *Communications of the ACM*, 27:1134–1142, 1984.

D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

J.A. Aslam and S.E. Decatur. On the sample complexity of noise-tolerant learning. *Information Processing Letters*, 57(4):189–195, 1996.

A. Blum, M. Furst, J. Jackson, M. J. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using fourier analysis. In *STOC*, 1994.

S. E. Decatur and R. Gennaro. On learning from noisy and incomplete examples. In *COLT*, pages 353–360, 1995.

S. E. Decatur. PAC learning with constant-partition classification noise and applications to decision tree induction. In *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 147 – 156, 1997.

M. Kearns. Efficient noise-tolerant learning from statistical queries. In *25th ACM Symposium on the Theory of Computing (STOC)*, pages 392 – 401, 1993.

N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. In *Conference on Learning Theory (COLT)*, 1991.

A. P. Dawid and A.M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979.

V. C. Raykar, S. Yu, L. H. Zhao, A. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: Whom to trust when everyone lies a bit. In *International Conference on Machine Learning (ICML)*, 2009.

Y. Yan, G. Hermosillo, R. Rosales, L. Bogoni, G. Fung, L. Moy, M. Schmidt, and J. G. Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

P. Donmez, J. Carbonell, and J. Schneider. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *Proceedings of the SIAM International Conference on Data Mining*, 2010.

L. G. Valiant. Learning disjunctions of conjunctions. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 560 – 566, 1985.

M. Kearns and M. Li. Learning in the presence of malicious error. *SIAM Journal on Computing*, 22(4):807–837, 1993.

S.A. Goldman and R.H. Sloan. Can PAC learning algorithms tolerate random attribute noise? *Algorithmica*, 14(1):70–84, 1995.

N. H. Bshouty, N. Eiron, and E. Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002.

O. Dekel and O. Shamir. Vox populi: Collecting high-quality labels from a crowd. In *Conference on Learning Theory (COLT)*, 2009a.

O. Dekel and O. Shamir. Good learners for evil teachers. In *International Conference in Machine Learning (ICML)*, 2009b.

O. Dekel, F. Fischer, and A. D. Procaccia. Incentive Compatible Regression Learning. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 277–286, 2008.

R. Meir, A. D. Procaccia, and J. S. Rosenschein. Strategyproof Classification under Constant Hypotheses: A tale of two functions. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*., 2008.

R. Meir, A. D. Procaccia, and J. S. Rosenschein. Strategyproof Classification with Shared Inputs. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial Classification. In *International Conference on Knowledge Discovery*

and Data Mining (KDD). ACM, August 22-25 2004.

G. L'Huillier, R. Weber, and N. Figueroa. Online Phishing Classification Using Adversarial Data Mining and Signaling Games. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2009.

M. Kantarcioglu, B. Xi, and C. Clifton. A Game Theoretical Framework for Adversarial Learning. In *CERIAS 9th Annual Information Security Symposium*, 2008.

R. B. Myerson. Optimal auction design. *Math. Operations Res.*, 6(1):58–73, Feb. 1981.

P. D. Laird. *Learning from good and bad data*. Kluwer Academic Publishers, Norwell, MA, USA, 1988.

R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, NY, USA, 1995.

V. Krishna. *Auction Theory*. Academic Press, 2002.

D. Mishra. An introduction to mechanism design theory. Unpublished Manuscript. URL: `http://www.isid.ac.in/~dmishra/doc/survey.pdf`, May 2008.

Combining local search techniques and path following for bimatrix games

Nicola Gatti, Giorgio Patrini, Marco Rocco
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Milano, Italy

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
Pittsburgh, USA

Abstract

Computing a Nash equilibrium (NE) is a central task in computer science. An NE is a particularly appropriate solution concept for two–agent settings because coalitional deviations are not an issue. However, even in this case, finding an NE is PPAD–complete. In this paper, we combine path following algorithms with local search techniques to design new algorithms for finding exact and approximate NEs. We show that our algorithms largely outperform the state of the art and that almost all the known benchmark game classes are easily solvable or approximable (except for the GAMUT CovariantGame–Rand class).

1 Introduction

Finding a Nash equilibrium (NE) of a normal–form (aka strategic–form aka bimatrix) game is PPAD–complete [8] even with just two players [6]. Although PPAD $\subseteq$ NP (but PPAD $\not\subseteq$ NP–complete unless NP = co–NP) it is generally believed that PPAD $\neq$ P. Thus the worst–case complexity of finding an NE is exponential in the size of the game. This pushed researchers to study approximate solution concepts, providing some polynomial–time algorithms that compute solutions with a guaranteed approximation bound [2, 11, 22]. However, bimatrix games do not have a fully polynomial–time approximation scheme unless PPAD $\subseteq$ P [6].

Worst–case complexity, being too pessimistic, is often a bad indicator of the actual performance of an algorithm, and *average–case* complexity is difficult to determine. A new metric of complexity, called *smoothed* complexity, has been gaining interest in recent years [21]. It studies how the introduction of small perturbations affects the worst–case complexity. Unfortunately, finding an NE in two–player games is not smoothed polynomial unless PPAD $\subseteq$ RP [5].

In this paper, we focus on two–player general–sum strategic–form games. An NE is a particularly appropriate solution concept in two–agent settings because coalitional deviations are not an issue. The main known algorithms are LH [14] based on linear complementarity mathematical programming (LCP), PNS [18] based on support enumeration, and MIP Nash [19] based on mixed–integer linear programming (MILP). None of these beats the others on all games, and all of them have worst–case complexity exponential in the size of the game. In particular, there are game instances, called hard–to–solve games (HtSG), that always require exponential time when solved with LH and PNS [20].

The integration of local search techniques with support enumeration algorithms, called LS–PNS, can be effective on games that are hard for all three algorithms above [3, 4]. (Other local search algorithms, based on simulated annealing or homotopy, that are slower but have convergence guarantees, have also been designed [12, 13].) In this paper, we combine local search techniques with path following algorithms (e.g., LH) instead. This opens new promising opportunities, allowing a dramatic reduction of the solution space: path following algorithms work on a solution space that is $O(2.6^m)$ where m is the number of actions per agent, while the number of supports is $O(4^m)$. Our contributions include the following.

(i) We design new algorithms: (a) an LH version with random restarts (rrLH)[1], (b) a version of the adaptation of the Lemke algorithm (L) proposed in [24] with random restarts (rrL), (c) a local search algorithm moving on the best response vertices (LS–v), and (d) an anytime LH version based on iteratively decreasing perturbation of the payoff matrices (ip–LH).

[1] A similar algorithm is presented in [7]. We extend such algorithm with an iterative deepening cutoff, we theoretically analyze its properties, and we provide a more thorough experimental analysis, developing a floating–point implementation that is about two times faster and developing a new arbitrary–precision implementation.

(*ii*) We prove that rrLH is asymptotically optimal in the space of algorithms that randomize over LH paths (the result can be extended to rrL).

(*iii*) We experimentally show that LH requires arbitrary precision with some classes of GAMUT games [17] and HtSG. L requires arbitrary precision for almost all the game instances.

(*iv*) rrLH outperforms all our and state–of–the–art algorithms, requiring an average number of steps that is linear in the size of the game except for CovariantGame–Rand and HtSG. rrL works asymptotically as rrLH, but its compute time is usually larger.

(*v*) LS–v outperforms the other algorithms in approximating hard instances, except HtSG.

(*vi*) HtSG can be easily approximated by ip–LH even with a perturbation of 10^{-10}. Thus there must exist a different worst case for LH unless PPAD $\subseteq$ RP. (In contrast, CovariantGame–Rand stays hard even when perturbed, thus it is a possible candidate.)

2 Game model and solution concepts

A bimatrix game is a tuple (N, A, U), where $N = \{1, 2\}$ is the set of agents (we denote by $i \in N$ a generic agent); $A = (A_1, A_2)$ where A_i is the set of $m_i = |A_i|$ actions available to agent i (we denote by $a \in A_1 \cup A_2$ a generic action); $U = (U_1, U_2)$ where U_i is the utility matrix of agent i [9]. Without loss of generality, we assume that $\max_{j,k}\{U_i(j,k)\} = 1$ and $\min_{j,k}\{U_i(j,k)\} = 0$ for every $i \in N$.

We denote by $\mathbf{x}_i$ the strategy (vector of probabilities) of agent i and by $x_{i,a}$ the probability with which agent i plays action $a \in A_i$. We denote by Δ_i the space of strategies over action space A_i, i.e., vectors where the probabilities sum to 1. Given strategy $\mathbf{x}_i$, the *support* S_i is the set of actions $a \in A_i$ played with a non–zero probability in $\mathbf{x}_i$.

The central solution concept is NE. A profile of strategies $(\mathbf{x}_1, \mathbf{x}_2)$ is an NE if and only if, for each $i \in N$, $\mathbf{x}_i^T U_i \mathbf{x}_{-i} \geq \mathbf{x}_i'^T U_i \mathbf{x}_{-i}$ for every $\mathbf{x}_i' \in \Delta_i$. The problem of finding an NE can be expressed as an LCP:

$$\mathbf{x}_i \geq 0 \qquad \forall i \in \{1,2\} \qquad (1)$$

$$\mathbf{1}v_i - U_i\mathbf{x}_{-i} \geq 0 \qquad \forall i \in \{1,2\} \qquad (2)$$

$$\mathbf{x}_i^T(\mathbf{1}v_i - U_i\mathbf{x}_{-i}) = 0 \qquad \forall i \in \{1,2\} \qquad (3)$$

$$\mathbf{1}^T\mathbf{x}_i = 1 \qquad \forall i \in \{1,2\} \qquad (4)$$

Here v_i is the expected utility of agent i. Constraints (1) and (4) state that every $\mathbf{x}_i \in \Delta_i$. Constraints (2) state that no pure strategy of agent i gives expected utility greater than v_i. Constraints (3) state that each agent plays only optimal actions.

The most common approximate solution concept is ϵ–Nash equilibrium (ϵ–NE): $(\mathbf{x}_1, \mathbf{x}_2)$ is an ϵ–NE if, for each $i \in N$, $\mathbf{x}_i^T U_i \mathbf{x}_{-i} \geq \mathbf{x}_i'^T U_i \mathbf{x}_{-i} - \epsilon$ for every $\mathbf{x}_i' \in \Delta_i$. Informally, $(\mathbf{x}_1, \mathbf{x}_2)$ is an ϵ–NE if at most each player loses a utility of ϵ w.r.t. playing her best response. Other approximate solution concepts that we use in this paper are ϵ–Well Supported–Nash equilibrium (ϵ_{WS}–NE) and regret–Nash equilibrium (r–NE). $(\mathbf{x}_1, \mathbf{x}_2)$ is an ϵ_{WS}–NE if, for each $i \in N$, $\mathbf{e}_k^T U_i \mathbf{x}_{-i} \geq \mathbf{e}_j^T U_i \mathbf{x}_{-i} - \epsilon$ for every $k \in S_i, j \in A_i$ ($\mathbf{e}_k$ is a vector of 0 but 1 in position k). Thus playing singularly each action of the support, the agent loses a utility of at most ϵ w.r.t. playing the her best response. Finally, $(\mathbf{x}_1, \mathbf{x}_2)$ is an r–NE if, for each $i \in N$, $r = \sum_{i=\{1,2\}} \sum_{l \in S_i} r_{i,l}$ and $\mathbf{e}_k^T U_i \mathbf{x}_{-i} \geq \mathbf{e}_j^T U_i \mathbf{x}_{-i} - r_{i,k}$ for every $k \in S_i, j \in A_i$.

3 Randomizing over paths

In this section we develop NE algorithms that randomize over *almost complementary* paths.

3.1 Randomizing over Lemke–Howson (LH) paths

For every agent i, define the best response polyhedron $P_i = \{\tilde{\mathbf{x}}_i \in \mathbb{R}^{m_i} | U_{-i}\tilde{\mathbf{x}}_i \leq \mathbf{1}, \tilde{\mathbf{x}}_i \geq \mathbf{0}\}$. Let V_i be the vertices of P_i. The space Θ of solutions traversed by LH is a subset of pairs of vertices of $V = V_1 \times V_2$. Let $\mathbf{s}_{-i}$ be the slack variables of $U_{-i}\tilde{\mathbf{x}}_i + \mathbf{s}_{-i} = \mathbf{1}$. Variables $s_{i,a}$ and $x_{i,a}$ are called *complementary*. A basic solution of the tableaux associated with P_1 and P_2 is *complementary* if the basis contains exactly one complementary variable for all i and a, while it is *almost complementary* if both variables of a single pair of complementary variables are in the basis. A completely complementary solution is an NE.

The initial solution of the LH algorithm [14] is generated by starting from the artificial solution $(\mathbf{0}, \mathbf{0})$ (corresponding to the case in which all the slack variables $\mathbf{s}_i$, for every i, are in the basis), and putting in the basis of one of the two tableaux a variable $x_{i,a}$ associated with some action $a \in A_1 \cup A_2$. Thus, there are $m_1 + m_2$ different possible initial solutions. LH follows a path of almost complementary solutions by repeatedly applying complementary pivoting steps. At each step, the current basis is changed by putting in the basis the variable that is complementary to the variable that has left the basis in the previous step, until a completely complementary solution has been found. The leaving variable is determined by the minimum ratio test [15] and it is unique except for degenerate games; however, degeneracy can be removed by introducing lexicographic perturbation [24].

For each initial solution, there is a different path leading to a (potentially different) NE: LH partitions the solution space in $m_1 + m_2$ different paths. If the path that the algorithm is following is an exponentially long one, it may be convenient to have restart.

We observe that any random restart policy over the solutions traversed by the LH algorithm can be formulated as a two–stage randomization policy: (1) ran-

domization over the $m_1 + m_2$ possible paths and (2) randomization, given a path, over its solutions. We initially present an algorithm, called rrLH, that adopts only a blind randomization of type (1) and, subsequently, we investigate randomization of type (2).

rrLH is described in Algorithm 1. It randomly chooses one of the paths (Step 2) and follows it (Step 4) until an NE is found; if the length (in terms of number of pivoting steps) of the path is larger than a given *cutoff* and there is a path that has not been visited till *cutoff* (Steps 5, 6, and 8), the algorithm restarts with a new path, otherwise (Step 7) *cutoff* is updated in an iterative deepening fashion and a restart is done.

Algorithm 1 Lemke–Howson with random restart (rrLH)

```
1: cutoff = cutoff_0
2: randomly choose one path non–visited till cutoff
3: repeat
4:     apply complementary pivoting
5:     if the path is longer than cutoff then
6:         if all the paths are visited till cutoff then
7:             cutoff = cutoff + cutoff_0
8:         go to Step 2
9: until a completely complementary solution is found
10: return current solution
```

(To improve the efficiency of the algorithm, we save the variables of the current basis before making a restart and, when a path is re–visited, we derive the last visited basis by matrix inversion and we restart from it.) The advantages of rrLH are simplicity and completeness. The potential drawbacks are that the number of possible paths is limited and that the algorithm is forced to move along fixed paths. Call l^* the shortest LH path. The compute time of rrLH is linear in l^*. When l^* is known or it is possible to estimate a tight upper bound, $cutoff_0$ can be conveniently fixed. Assigning, e.g., $cutoff_0 = l^*$, we have that the worst case compute time is $l^*(m_1 + m_2)$, while the average is $l^*(m_1 + m_2 + 1)/2$. Thus, if l^* grows in length exponentially in $m_1 + m_2$, also rrLH compute time grows in length exponentially. Adopting a non–blind randomization of type (1) would keep the compute time, even in the best case, linear in l^* and hence is useless.

We focus on randomization of type (2). This type of randomization cannot be efficiently achieved with LH. Indeed, differently from what happens with support–enumeration algorithms [3, 4] where it is possible to start from any support profile, no every possible almost complementary basis corresponds to a feasible starting solution for LH: some sets of almost complementary variables are not feasible basic solutions, while the feasible ones can belong to LH paths or not, and in this latter case they may belong to cyclic paths that do not lead to any equilibrium (the membership of a solution to a path can be discovered only during the traversing of the path itself and therefore, if the algorithm starts from an arbitrary solution, it cannot know the path over which it is moving and cannot distinguish paths from cycles). Thus, an initial solution must be searched by using pivoting and the number of pivoting steps may be exponential in $m_1 + m_2$. However, we show that any algorithm with randomization of type (2) has a compute time that is asymptotically the same of our rrLH and therefore rrLH is (asymptotically) optimal in the space of the algorithms making random restarts over the LH paths. This holds even dropping completeness and considering algorithms that find an NE with a probability p. Initially, we state the following lemma.

Lemma 1. *The best* cutoff *of a generic randomized algorithm that finds the terminal vertex of an l–step–long path with probability p and is able to position blindly in every vertex of the path is $l \cdot p$.*

Proof. The best configuration of the algorithm can be obtained by minimizing the number of steps of the algorithm (i.e., $cutoff \cdot res$, where res is the number of restarts), under the constraint that the probability to find the terminal is p, i.e., $p = 1 - (1 - cutoff/l)^{res}$, $res \geq 1$, and $cutoff \geq 1$. From $p = 1-(1-cutoff/l)^{res}$ it follows that $res = \log(1-p)/\log(1-cutoff/l) \geq 1$, thus $\log(1-p) \geq \log(1 - cutoff/l)$, that means $p \leq cutoff/l$. $res \cdot cutoff$ can be rewritten as $\log(1-p)/\log(1 - cutoff/l) \cdot cutoff$. After having removed the negative constant $\log(1-p)$ the objective is to maximize the monotonic decreasing function $cutoff/\log(1-cutoff/l)$ under the constraint $p \leq cutoff/l$. The optimum is $cutoff = l \cdot p$ and $res = 1$. □

From Lemma 1 it follows that, even when it is possible to perform a blind randomization over the solutions composing a single path, like stage (2) prescribes, the optimal configuration of the algorithm is such that this path is traversed only once without making restarts (i.e. $res = 1$). Thus, this path can be safely removed from the set of the available paths at stage (1). From Lemma 1, we can easily derive the following lemma.

Lemma 2. *The worst case compute time of a randomized algorithm finding the terminal vertex of an l–step–long path and able to position blindly in every vertex of the path is l, while the average time is $l/2$.*

Finally, we show that including randomization of type (2) the compute time keeps to be linear in l^* as stated by Lemma 3. From Lemma 2 we can state the following.

Lemma 3. *LH with blind randomization policy of type (2) has a compute time $O(l^*)$.*

We can evaluate for specific interesting cases the ratio between the average compute time of an algorithm Π adopting randomization of type (2) and the one of an algorithm Π' that does not. Assign $cutoff = l^*$ and assume, for simplicity, that the length of all the non–shortest paths is $l > l^*$ and $m_1 = m_2 = m$.

The average compute time of Π is $(2m+1)/2 \cdot l^*$. The average case compute time of Π' is given by $1/2m \cdot l^*/2 \cdot \sum_{k=0}^{2m}(1-l^*/2)^k + (l^*)^2/2l \cdot 1/2n \cdot \sum_{k=0}^{2m-1}(1-l^*/l)^k(2m-1-k)$. The two opposite possible situations are: $l^*/l \to 1$ and $l^*/l \to 0$. In the best situation $l^*/l \to 1$ the average compute time is $l^*/2$ and therefore randomization of type (2) reduces the compute time by $2m+1$ times. In the worst situation $l^*/l \to 0$, the ratio is 1 and therefore no reduction is obtained. In both cases, the compute time keeps to be linear in l^*. We observe that, randomization of type (2) exploits the existence of paths that are not excessively longer than the shortest path.

Now, we focus on non–blind randomization of type (2), providing a negative result.

Lemma 4. *Any algorithm that finds the terminal vertex of an l–step–long path with probability p and able to position non–blindly in every vertex of the path, requires either an exponential number (*res*) of restarts or an exponential* cutoff *as l grows in length exponentially.*

Proof. Suppose to have an oracle that, given a *cutoff* and an almost complementary solution, is able to say whether or not such a solution is farther than *cutoff* from the terminal vertex. If the solution is farther, then it is discarded, otherwise the algorithm follows the path from the given solution and the terminal vertex. The probability to find a randomly generated solution that is not farther than *cutoff* from the equilibrium by *res* random restarts is: $1-(1-\mathit{cutoff}/l)^{res}$. By posing: $1-(1-\mathit{cutoff}/l)^{res} = p$, we obtain $res = \log(1-p)/\log(1-\mathit{cutoff}/l)$. When $\lim_{l\to+\infty} \mathit{cutoff}/l = 0$, we can write $res = -l\log(1-p)/\mathit{cutoff}$. Therefore, if l grows in length exponentially then either *res* or *cutoff* grow in length exponentially. When $\lim_{l\to+\infty} \mathit{cutoff}/l > 0$, *cutoff* grows in length exponentially as l does. □

From the above lemma, it can be easily observed that when all the LH paths grow in length exponentially non–blind randomization is useless: even dropping the completeness and accepting that the NE can be found with a probability p, in the worst case the compute time is $O(l^*)$. Thus, rrLH is asymptotically optimal among algorithms randomizing over LH paths.

3.2 Randomizing over Lemke (L) paths

As discussed in the previous section, randomization over paths exhibits a compute time that depends on the length of shortest path. The main drawback of LH is that the number of available paths is small. In this section, we resort to the Lemke algorithm adaptation as prescribed by [24], which we will call L. This algorithm allows for an arbitrary initial solution, each corresponding to a different path, and therefore it allows for an infinite number of paths. Define the polyhedron P as follows.

$$P = \left\{ [z_0 \;\; \mathbf{z}_1 \;\; \mathbf{z}_2] \;\middle|\; \begin{array}{cc} M_{1,1}\mathbf{z}_1 + M_{1,2}\mathbf{z}_2 + \mathbf{d}_1 z_0 & = \mathbf{q}_1 \\ M_{2,1}\mathbf{z}_1 + M_{2,2}\mathbf{z}_2 + \mathbf{d}_2 z_0 + \mathbf{q}_2 & \geq 0 \\ z_0, \mathbf{z}_2 & \geq 0 \end{array} \right\}$$

with $\mathbf{d}_1 = \mathbf{1}, \mathbf{d}_2 = [-(U_1\overline{\mathbf{x}}_2)^T, -(U_2\overline{\mathbf{x}}_1)^T]^T$ where $\overline{\mathbf{x}}_1$ and $\overline{\mathbf{x}}_2$ are parameters (therefore P is parametric), $\mathbf{q}_1 = -\mathbf{1}$, $\mathbf{q}_2 = \mathbf{0}$ and

$$\mathbf{z}_1 = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \quad M_{1,1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad M_{1,2} = \begin{bmatrix} \mathbf{1}^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{1}^T \end{bmatrix}$$

$$\mathbf{z}_2 = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \quad M_{2,1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \quad M_{2,2} = \begin{bmatrix} 0 & -U_1 \\ -U_2 & 0 \end{bmatrix}$$

Let V be the set of vertices of P. The space Θ of solutions traversed by the Lemke algorithm is a subset of V. Call $\mathbf{w}$ the slack variables of $M_{2,1}\mathbf{z}_1 + M_{2,2}\mathbf{z}_2 + \mathbf{d}_2 z_0 + \mathbf{q}_2 - \mathbf{w} = 0$ and consider the associated tableau. Call z_j, with $j \neq 0$, the j–th element of $\mathbf{z} = [\mathbf{z}_1^T, \mathbf{z}_2^T]$. Variables z_j and w_j are called complementary. A solution is completely complementary if the basis contains one complementary variable between z_j and w_j for every j (therefore z_0 is out the basis), while a solution is almost complementary if both variables z_j and w_j for a single j are not in the basis but z_0 is.

The algorithm moves along almost complementary solutions, each corresponding to a pair of strategies $(\mathbf{x}_1 + z_0\overline{\mathbf{x}}_1, \mathbf{x}_2 + z_0\overline{\mathbf{x}}_2)$. The initial solution is such that: (a) z_0 is in the basis and it is equal to 1, (b) all the variables $x_{i,a}$ such that a is a best response to $\overline{\mathbf{x}}_{-i}$ are in the basis except one, and (c) for all the non–best–response actions a the complementary variable w_j of $x_{i,a}$ is in the basis. Given the initial solution, the algorithm follows a path of almost complementary solutions by repeatedly applying complementary pivoting (the entering variable is the complementary variable of the leaving variable at the previous step, while the leaving variable is determined by the lexico minimum ratio test). At the first step of L, the entering variable is $x_{i,a}$ such that a is the only best response not yet in the basis. (L terminates when z_0 is the leaving variable, finding as NE.)

LH has a finite number of possible paths $(m_1 + m_2)$, while L has an infinite number of them, thus we try to design a non–blind randomization policy among paths and we fixed a limit to the restarts, otherwise in the worst case the compute time would be infinite. We design the version of L with random restarts (rrL) like rrLH except that: (a) the initial solution is determined by randomly generating $\overline{\mathbf{x}}_1$ and $\overline{\mathbf{x}}_2$; since there are infinitely many possible initial solutions and many of them could lead to potentially long paths, (b) we use quality metrics to accept or discard an initial solution θ (we accept θ if $g(\theta) > th$, where g can be defined in different ways, e.g., ϵ, ϵ_{WS}, and r, and th is a threshold), (c) we use a cutoff as a function of how the different metrics ϵ, ϵ_{WS}, r, and z_0 decrease

along the path; and (d) we disable the cutoff after a given number of restarts to guarantee completeness, the potential restarts would be infinite otherwise.

The advantage of rrL is that the initial solution can be any and thus much more paths than rrLH can be followed. The potential drawbacks are that the algorithm is forced to move along fixed paths and that the pivoting step requires about twice as much compute time as LH due to the tableau size.

4 Local search on best response vertices

We cast NE finding as a local search based optimization problem (Θ, f, N) where Θ is the solution space, f is a function $f : \Theta \to \mathbb{R}$ to minimize, and N is the *neighborhood function* that specifies, for each solution $\theta \in \Theta$, a set $N(\theta) \subseteq \Theta$ of solutions that can be directly reached from θ [16].

The solution space Θ is the set of vertices V_i of the polyhedron $P_i = \{\tilde{\mathbf{x}}_i \in \mathbb{R}^{m_i} | U_{-i}\tilde{\mathbf{x}}_i \leq \mathbf{1}, \tilde{\mathbf{x}}_i \geq \mathbf{0}\}$ that constitutes the best response polyhedron of agent i, where agent i is the agent with the minimum number of actions. This choice allows one to reduce the solution space as shown below (however, our algorithm can be applied with any i). Every vertex $\tilde{\mathbf{x}}_i$ of V_i, except for $\tilde{\mathbf{x}}_i = \mathbf{0}$, is equivalent to a strategy $\mathbf{x}_i = \frac{1}{\mathbf{1}^T\tilde{\mathbf{x}}_i}\tilde{\mathbf{x}}_i$.

The function f to minimize associates each strategy $\mathbf{x}_i = \overline{\mathbf{x}}_i$ with the ϵ value of the best ϵ–Nash equilibrium when agent i plays $\overline{\mathbf{x}}_i$. This value can be computed as:

$$\min_{\epsilon, \mathbf{x}_{-i}} \quad \epsilon \tag{5}$$

$$\text{s.t.} \quad \overline{\mathbf{x}}_i^T U_i \mathbf{x}_{-i} + \epsilon - \mathbf{e}_k^T U_i \mathbf{x}_{-i} \geq 0 \qquad \forall k \in A_i \tag{6}$$

$$\overline{\mathbf{x}}_i^T U_{-i} \mathbf{x}_{-i} + \epsilon - \overline{\mathbf{x}}_i^T U_{-i} \mathbf{e}_k \geq 0 \qquad \forall k \in A_{-i} \tag{7}$$

$$\mathbf{1}^T \mathbf{x}_{-i} = 1 \tag{8}$$

$$\mathbf{x}_{-i} \geq \mathbf{0} \tag{9}$$

$$\epsilon \geq 0 \tag{10}$$

The constraints ensure that $\mathbf{x}_{-i} \in \Delta_{-i}$ and ϵ is non–negative. Note that the solution ϵ^* of the above linear program is 0 if and only if there is some $\mathbf{x}_{-i} = \overline{\mathbf{x}}_{-i}$ such that $(\overline{\mathbf{x}}_i, \overline{\mathbf{x}}_{-i})$ is an NE. Therefore, we recognize whether or not a local minimum is a global minimum.

Given a vertex $\tilde{\mathbf{x}}_i$ of V_i, the neighbors are all the adjacent vertices of $\tilde{\mathbf{x}}_i$. These vertices can be found by exploiting pivoting. More precisely, consider the tableau $U_{-i}\tilde{\mathbf{x}}_i + \mathbf{s} = \mathbf{1}$ where $\mathbf{s}$ are slack variables. A basic solution of the tableau is composed by m_{-i} variables belonging to $\tilde{\mathbf{x}}_i$ and $\mathbf{s}$ (when only variables $\mathbf{s}$ are in the basis we have the artificial solution $\tilde{\mathbf{x}}_i = 0$). A vertex $\tilde{\mathbf{x}}_i$ of V_i corresponds to a basic solution of the tableau. We can change the basis by making a non–basic variable enter the basis (the leaving variable is uniquely determined by the minimum ratio test). The number of entering variables is m_i; thus each vertex has exactly m_i neighbors. In contrast, in the local search over supports, each solution has $m_1 \cdot m_2$ neighbors.

The local search algorithm for solving (Θ, f, N), called LS–v, is provided in Algorithm 2. The initial solution (Step 1) is generated starting from the artificial solution $\tilde{\mathbf{x}}_i = \mathbf{0}$ and applying a random number of random pivoting steps (as described above). In principle, any possible vertex of P_i may be an initial solution. We use a tabu list to keep track of previously generated initial solutions in order to avoid repetitions. Given solution s, the algorithm searches in the neighborhood $N(s)$ for a solution s' such that $f(s') < f(s)$ (Step 3). This search is driven by a heuristic. We use three main heuristics: *best improvement* (BI) where all neighbors are generated and the best neighbor better than the current is chosen, *first improvement* (FI) where the neighbors are searched in a given order and the first neighbor better than the current is chosen, *first improvement with random generation* (FIR) where the neighbors are searched randomly and the first neighbor better than the current solution is chosen. (With FIR we disable the tabu list for the initial solution because randomization makes the algorithm follow different paths at every execution, and we generate at most *max–n* neighbors.) If there is no neighbor better than the current solution or if the path length is longer than *cutoff* and there exists an unvisited initial solution, then a restart is conducted (Steps 4 and 5).

Algorithm 2 Local search on best response vertices (LS–v)

1: randomly choose an unvisited initial solution s from $\tilde{\mathbf{x}}_i = \mathbf{0}$ by pivoting
2: **repeat**
3: choose neighbor $\theta' \in N(\theta)$ with $f(\theta') < f(\theta)$ and assign $\theta = \theta'$
4: **if** there is no θ' or the path is longer than *cutoff* **then**
5: go to Step 1
6: **until** $f(\theta) > 0$
7: return current solution

The advantages of rrL are that it can traverse solutions that are not in LH/L paths and that the number of neighbors is low (w.r.t. local search on supports). The drawback is that the algorithm is incomplete.

5 Anytime iterative perturbation over paths

Given a two–player game with utilities (U_1, U_2) and a game with (U_1', U_2') where every U_i' is obtained by perturbing each payoff of U_i with an arbitrary probability distribution over $[-\delta, \delta]$, it has been shown that an NE for (U_1', U_2') is an ϵ–NE for (U_1, U_2) with $\epsilon \leq 4\delta$ [5]. We argue that the bound is tighter:

Theorem 1. *Given a two–player game with (U_1, U_2) and a game with (U_1', U_2') where every U_i' is obtained by perturbing each payoff of U_i with an arbitrary probability distribution over $[-\delta, \delta]$, an NE $(\hat{x}_1, \hat{x}_2)$ for*

(U_1', U_2') *is an* ϵ*–NE for* (U_1, U_2) *with* $\epsilon \leq 2\delta$.

Proof Call x_1^* the best response of agent 1 to $\hat{x}_2$ of agent 2 for the game (U_1, U_2). By definition, $||U_i - U_i'||_\infty \leq \delta$. We can compute the upper bound over the expected utility loss of agent 1: $\epsilon = x_1^* U_1 \hat{x}_2 - \hat{x}_1 U_1 \hat{x}_2 \leq x_1^* U_1 \hat{x}_2 - x_1^* U_1' \hat{x}_2 + \hat{x}_1 U_1' \hat{x}_2 - \hat{x}_1 U_1 \hat{x}_2 \leq |x_1^* U_1 \hat{x}_2 - x_1^* U_1' \hat{x}_2| + |\hat{x}_1 U_1' \hat{x}_2 - \hat{x}_1 U_1 \hat{x}_2| \leq x_1^* ||U_1 - U_1'||_\infty \hat{x}_2 + \hat{x}_1 ||U_1' - U_1||_\infty \hat{x}_2 \leq ||U_1 - U_1'||_\infty + ||U_1' - U_1||_\infty \leq 2\delta$. The same reasoning can be applied to agent 2, obtaining the same upper bound. Hence, the theorem is proved. □

We exploit this result to produce a simple anytime algorithm (Algorithm 3) based on the idea that an hard game could become easy after perturbed. The algorithm iteratively applies LH starting from a perturbed game obtained with a large perturbation (i.e., $\delta = 0.125$) and then exponentially reducing the perturbation at each step where LH has found an NE.

Algorithm 3 Anytime iterative perturbation over paths (ip-LH)

```
1: set k = 3
2: while deadline has not been reached do
3:     set δ = 1/2^k
4:     apply δ–uniform perturbation on (U_1, U_2)
5:     apply LH to perturbed (U_1, U_2)
6:     set k = k + 1
7: return current solution
```

6 Experimental evaluation

6.1 Experimental setting

We implemented LH, L, rrLH, rrL and ip–LH in C in two different versions: one with floating–point and one with arbitrary–precision integer arithmetic by means of the GMP library[2]. We implemented LS–v, PNS and LS–PNS in C calling CPLEX via AMPL to solve LPs, while MIP Nash directly in AMPL[3] with CPLEX[4]. All the algorithms that use pivoting techniques are optimized with the revised technique [15] and those using GMP with integer pivoting [23] to save compute time. (Revised technique allows us to store and perform pivoting on a reduced tableau, obtaining a relevant improvement in the performance of the algorithms.) We conducted the experiments on a UNIX computer with 2.33GHz CPU, 16GB RAM and kernel 2.6.24.

Using GAMUT [17], we generated 500 instances of CovariantGame–Rand, GraphicalGame–RG and PolymatrixGame–RG games (being the hardest classes w.r.t. PNS, LH and MIP Nash) and 10 instances of all the other classes with $m_1 = m_2 = m \in \{5, \ldots, 150\}$ and a step of 5. We also generated an instance of SGC games with $m \in \{3, \ldots, 99\}$ and a step of 4 as prescribed in [19]. In SGC games, all equilibria have a medium number of pure strategies in their supports. Finally, we generated one instance of HtSG (its generation not being random) with $m \in \{2, \ldots, 18\}$ and a step of 2 as prescribed in [20]. In HtSG, all the LH paths are exponentially long.[5]

6.2 Numerical instability with floating point precision

We evaluated numerical instability of LH and L with floating–point precision for different game classes, comparing their results to those obtained with arbitrary precision. With GAMUT game classes, L presents numerical instability (the algorithm goes in ray termination) even with small games ($m = 30$) for all the game classes, while LH presents instability with a limited number of game classes (not recognizing equilibria or finding non–NE solutions). With long double precision, LH performs a wrong path on 96.1% of 100x100 TravelersDilemma instances and on 22.75% of 100x100 WarOfAttrition instances. More available tricks could be used to diminish the numerical instability than the ones used in our implementation, but the result would be the same: LH and L are intrinsically unstable. Using other tricks we can increase the size of the games at which instability appears. However, the relative comparison of the game classes in terms of arbitrary precision stands. On HtSG, arbitrary precision arithmetic is required even during the game instance generation. We generated game instances using both the trigonometric and the non–trigonometric moment curves as described in the appendix of [20]. We counted, using the *lrs* algorithm [1], the number of vertices of the polyhedron associated with the game generated with floating point precision and compared it to that obtained with arbitrary precision. When at least one vertex is lost, the polyhedron loses its hardness characteristics and at least a short path appears. Tab. 1 shows that arbitrary precision is required for instances with ≥ 14 actions per agent.

Table 1: Percentage of lost vertices in HtSG due to floating–point precision with different moment curves.

Actions per agent					
6	8	10	12	14	16
non–trigonometric					
0%	0%	0%	0%	1%	29%
trigonometric					
0%	64%	72%	95%	92%	82%

6.3 Path distribution

We ran LH along every possible path with all the 100x100 (99x99 with SGC) instances of our experimental setting (except HtSG), measuring the length of

[2] http://gmplib.org/.

[3] http://www.ampl.com.

[4] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[5] Although HtSG square instances can be easily solved by support enumeration algorithms, the equilibrium being unique and fully mixed, non–square games can be built such that support enumeration algorithms require exponential time.

the paths and deriving their distribution for each game class. Almost all the distributions are *fat–tailed* [10]. These distributions present a lot of data points in the tail, showing that the performance of an algorithm may vary dramatically from run to run. Formally, a distribution is fat–tailed if its *kurtosis* (μ_4/μ_2^2, where μ_j is the j–th moment) is larger than 3 (i.e., the kurtosis of a standard normal). All the game classes have a kurtosis larger then 3, except for DispersionGame, MinimunEffortGame (the kurtosis is not defined since $\mu_2 = 0$ and $\mu_4 = 0$, all the paths having the same length), SGC (1.0), TravelersDilemma (1.8), and WarOfAttrition (2.39). As suggested in [10], with fat–tailed distributions, random restarts may drastically improve performance. Our experiments, discussed below, confirm this.

Although almost all the classes present fat–tailed distributions, the performance of LH varies greatly across them. We ran LH with instances of different sizes and bucketed the game classes into five groups:

1. (DispersionGame, MinimumEffortGame) The length of all the paths is constant (= 2) with game size.
2. (BidirectionalLEG–CG/RG/SG, CovariantGame–Pos, LocationGame, UniformLEG–CG/RG/SG, WarOfAttrition) The average and maximum path length tends to a constant value (< 10) as game size increases.
3. (SGC, TravelersDilemma) The average path length increases linearly in game size.
4. (BertrandOligopoly, CovariantGame–Rand/Zero, GraphicalGame–RG/Road/SG/SW, PolymatrixGame–CG/RG/Road/SW, RandomGame) These games have an exponential growth of the average number of steps with the game size, but there can be some paths with polynomial length. Fig. 1 reports, for two classes, the average number of steps and the pertinent *box–plot* diagram: *median*, 1–st and 3–rd *quartiles* (dashed), *max* and *min* (dotted).
5. (HtSG). All the paths are exponentially long.

We produced the same analysis for L by randomly generating m initial solutions. All the classes present the same behavior they have with LH — except DispersionGame, which, when solved with L, belongs to Group 2. HtSG preserves, with L, the same characteristic exhibited with LH: the length of the shortest paths grows exponentially.

6.4 Finding an NE by rrLH

Groups 1–3 are easy even without resorting to random restarts (these games are easy also with other algorithms, e.g. PNS). Thus, we just briefly summarize the main results omitting details. Instances of

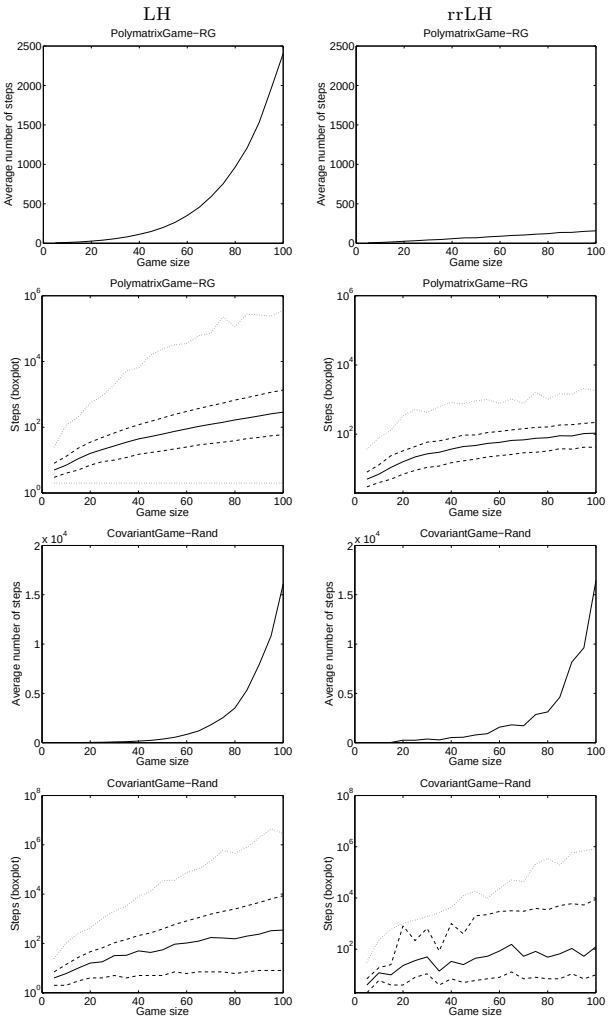


Figure 1: Comparison between LH (left) and rrLH (right) in terms of average number of steps and box–plot (median, 1–st/3–rd quartiles with dotted lines, min/max with dashed lines) of PolymatrixGame–RG and CovariantGame–Rand.

Groups 1 and 2 are solved by LH with a very small number of steps even in the worst cases (the compute times are less than one second). With Group 3, LH requires a linear compute time. With SCG it has already been experimentally demonstrated that LH outperforms MIP Nash and PNS [19]. With TravelersDilemma, LH (with arbitrary precision) is essentially the unique applicable algorithm — the commercial LP solvers do not support arbitrary precision, while a preliminary evaluation of GLPK with arbitrary precision led to compute times two orders of magnitude longer.

Random restarts play a crucial role with Group 4. These games are hard (in average) when solved with LH (and the other known algorithms: PNS, MIP Nash, LS–PNS). The application of rrLH with *cutoff* = 20 makes all the classes except for CovariantGame–Rand easy: all the executions of rrLH find an NE by the deadline and the average compute time is lin-

Table 2: Termination percentage and compute times.

rr–LH		MIP Nash		PNS	
BetrandOligopoly					
(100%)	0.27 s	(100%)	2.25 s	**(100%)**	**0.15 s**
CovariantGame–Rand					
(80%)	**176.61 s**	(10%)	254.02 s	(20%)	18.13 s
CovariantGame–Zero					
(100%)	**0.15 s**	(80%)	155.79 s	(90%)	3.75 s
GraphicalGame–RG					
(100%)	**0.16 s**	(10%)	124.42 s	(90%)	3.64 s
GraphicalGame–Road					
(100%)	**0.13 s**	(60%)	98.32 s	(90%)	4.47 s
GraphicalGame–SG					
(100%)	**0.15 s**	(40%)	180.61 s	(60%)	2.30 s
GraphicalGame–SW					
(100%)	**0.14 s**	(30%)	155.28 s	(90%)	3.42 s
PolymatrixGame–CG					
(100%)	**0.14 s**	(30%)	213.77 s	(60%)	3.27 s
PolymatrixGame–RG					
(100%)	**0.14 s**	(60%)	160.80 s	(90%)	1.14 s
PolymatrixGame–Road					
(100%)	**0.15 s**	(70%)	208.48 s	(90%)	2.51 s
PolymatrixGame–SW					
(100%)	**0.13 s**	(20%)	73.73 s	(90%)	2.43 s
RandomGame					
(100%)	**0.13 s**	(50%)	160.14 s	(60%)	4.80 s

ear the game size (the same for max and median), as shown in Fig. 1 for PolymatrixGame–RG. With CovariantGame–Rand, we observed that there is at least an instance whose shortest path grows in length exponentially (therefore, as shown in Section 3, no *cutoff*, even with iterative deepening, can lead to non–exponential compute times). As a result, the average number of steps is not smaller than that of LH.

We compared rrLH with floating–point precision (arbitrary precision is averagely 56 times slower) to MIP Nash and PNS with 150x150 instances of Group 4 (LS–PNS is not useful for non–hard instances [3]). Tab. 2 shows the percentage of instances solved within 600 s and the compute time in seconds. rrLH dramatically outperforms the other two algorithms. More precisely, PNS terminates very quickly if there is an NE with very small support and takes exponential time otherwise. Call $\underline{s}$ the smallest support size of all the NEs of a game. PNS scans $O(n^{2\underline{s}})$ supports before finding an NE, instead we observed that rrLH makes $O(2n\underline{s})$ pivoting steps. Thus, as n increases, PNS is inefficient even with small $\underline{s}$, instead rrLH scales well. MIP Nash performs radically worse than the others.

With HtSG, obviously, rrLH is not effective.

6.5 Finding an NE by rrL

The performance of rrL with blind random restarts are similar to rrLH, requiring for some classes (e.g. CovariantGame–Rand) a smaller number of steps but requiring more time per step. Given that all the other classes can be easily solved by applying rrLH except for CovariantGame–Rand and HtSG, we focused only on these two hard classes exploiting the main peculiarity of rrL: adopting a non–blind approach, trying to characterize good initial solutions and discarding those that are not promising.

We initially studied correlation between the values of ϵ, ϵ_{WS}, and r of the initial solutions and the length of the paths with L, and we observed that no statistical correlation is present. Thus, such parameters cannot be used to identify short paths and discard those potentially long. Instead, we found statistical correlation for CovariantGame–Rand (and PolymatrixGame–RG) between the length of the L paths and the distance (d_∞) in $||\cdot||_\infty$ between the initial solution and the equilibrium (the steps reduce as d_∞ increases), as shown in Fig. 2. Because the equilibrium is unknown, we can only use the vertices of the simplices as initial solutions (these are by definition the farthest points). However, with these initial solutions, L behaves like LH. This justifies why, although L allows potentially infinite paths, LH performs like L.

Given the difficulty of characterizing good initial solutions, we tried to characterize good paths by analyzing how some metrics (ϵ, ϵ_{WS}, r, z_0) vary during the paths. We designed a measure defined as $decr(z_0, h) = \frac{1}{z_{0_1}} \sum_{k=1}^{h} |z_{0_k} - z_{0_{k+1}}|$ that is equal to 1 when z_0 decreases monotonically and greater than 1 otherwise. Fig. 2 shows that with CovariantGame–Rand, $decr(z_0, h)$ is high on long paths. However, experimentally, this strategy resulted ineffective even fixing a small *cutoff*. Furthermore, there is no statistical correlation when ϵ or ϵ_{WS} or r are used in place of z_0.

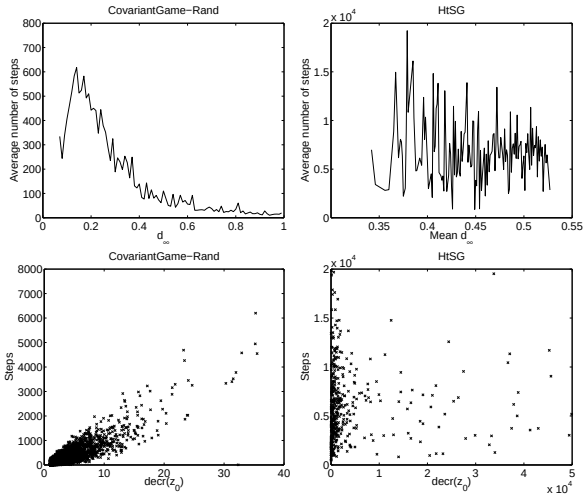


Figure 2: Relation between the measures d_∞, $decr(z_0, m)$ and number of steps with 50x50 CovariantGame–Rand and 16x16 HtSG.

6.6 Finding an NE by LS–v

We mainly focused the evaluation of LS–v on CovariantGame–Rand, the other classes being easy and LS–v being not applicable to HtSG due to lack of arbitrary precision. We tuned the parameters

cutoff and *max–n* of FIR in LS–v as follows. We isolated 15 100x100 hard CovariantGame–Rand instances that cannot be solved by 600 s by PNS, MIP Nash or rrLH. We randomly chose 5 instances and tuned LS–v with $cutoff \in \{m, 2m, m^2, 2m^2\}$ and $max\text{–}n \in \{m/2, m, 2m, m^2/2\}$. The best configuration was $cutoff = m^2/2$ and $max\text{–}n = 2m^2$. With such configuration, the number of random restarts is very small. In practice, every solution has a better neighbor with high probability. Then we compared the performance of the three heuristics BI, FI and FIR (with its best configuration) with the other 10 hard instances (non–used for the tuning). The best heuristic was FIR with $\epsilon = 3.64 \cdot 10^{-4}$ by 600 s, while $\epsilon > 3 \cdot 10^{-3}$ with the other two. However, LS–v never found an NE with these 10 hard instances. In addition, we evaluated LS–v with generic (not necessarily hard) instances. Tab. 3 reports success probability and ϵ of the best solution found by 600 s. LS–v is outperformed by rrLH and rrL (that found an NE with GraphicalGame–RG and PolymatrixGame–RG with a probability of 100% and of 30% with 100x100 CovariantGame–Rand).

Table 3: Percentage of solved instances and average best ϵ–NE found within 600 s with LS–v.

Actions per agent		
60	80	100
CovariantGame–Rand		
(56%)$9.34 \cdot 10^{-5}$	(63%)$1.82 \cdot 10^{-4}$	(13%)$1.80 \cdot 10^{-4}$
GraphicalGame–RG		
(56%)$1.32 \cdot 10^{-4}$	(38%)$1.72 \cdot 10^{-4}$	(36%)$7.83 \cdot 10^{-4}$
PolymatrixGame–RG		
(60%)$3.91 \cdot 10^{-5}$	(40%)$3.95 \cdot 10^{-4}$	(30%)$2.57 \cdot 10^{-4}$

6.7 Approximating an NE

We compared LS–v to the anytime versions of the other algorithms, obtained by keeping track of the best ϵ–NE during the pivoting, and ip–LH. Tab. 4 shows that LS–v is the best anytime algorithm (including PNS–anyT) for the 15 hard CovariantGame–Rand instances by an order of magnitude. It can be observed that ip–LH does not perform well with CovariantGame–Rand. This result shows that CovariantGame–Rand instances are stable, keeping to be hard even when perturbed.

We compared the anytime performance of LS–v and LH–anyT, these algorithms working on a similar solution space. Fig. 3 shows that LS–v found very good approximate solutions within a small number of steps and it outperforms LH–anyT for all setting of run time (the other heuristics provide similar results: LS–v finds very quickly a good approximate equilibrium).

We studied the ϵ–NEs with 16x16 HtSG. These games were easy for ip–LH (with arbitrary precision), requiring less than 10 steps even with a perturbation of 10^{-10}, returning $\epsilon \approx 10^{-12}$ (we cannot apply a perturbation smaller than 10^{-10} due to limits of GMP library). This shows that HtSG instances are highly unstable when perturbed. Therefore, there must exist another game class that is hard for LH and that is stable unless PPAD $\subseteq$ RP. Finally, LH–anyT provided best performance, returning $\epsilon \approx 10^{-34}$ within 600 s.

Table 4: Average best ϵ–NE found within 600 s on 100x100 hard CovariantGame–Rand.

LH–anyT	L–anyT	PNS–anyT	MILP–anyT
$6 \cdot 10^{-3}$	$2 \cdot 10^{-2}$	$8 \cdot 10^{-2}$	$3 \cdot 10^{-3}$

ip–LH	LS–v	LS–PNS
$1 \cdot 10^{-2}$	$\mathbf{3 \cdot 10^{-4}}$	$2 \cdot 10^{-3}$

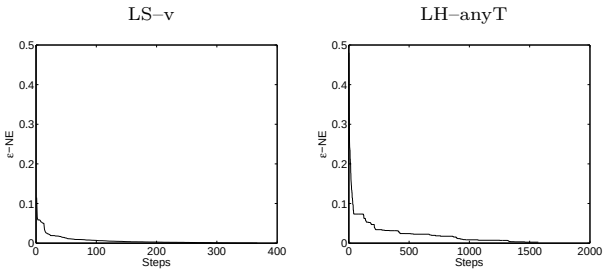


Figure 3: How the best ϵ–NE found so far changes during two sample executions of LS–v and LH–anyT.

7 Conclusions and future research

The computation of an NE is a challenging task even with two agents. In this paper we present by and large the fastest algorithms for the problem to date. We also present other results about the problem. For many situations, arbitrary–precision arithmetic is necessary even with LH. Complementary pivoting with blind random restarts over paths is the best heuristic, linearizing the average compute time in game size and outperforming all our and state–of–the–art algorithms for all the benchmark game classes except CovariantGame–Rand and HtSG. We provide theoretical results that allow us to say that rrLH is asymptotically optimal among algorithms that randomize over LH paths. We did not find any metric to characterize good L paths and therefore to have non–blind random restarts. Local search guided by the minimization of ϵ exhibits worse performance in finding exact equilibria, but it is the best in approximating hard instances, except for HtSG. HtSG games are unstable in the sense that they can be easily approximated by introducing a very small perturbation; therefore there must exist an alternative hard instance for LH that is stable (CovariantGame–Rand is a possible candidate, it being stable). In the future, we will extend our algorithms to games with more than two agents and isolate hard stable instances for each algorithm.

Acknowledgements

Tuomas Sandholm was funded under NSF grants IIS-0964579, CCF-1101668, and IIS-0905390.

References

[1] D. Avis, G. D. Rosenberg, R. Savani, and B. Von Stengel. Enumeration of Nash equilibria for two–player games. *ECON THEOR*, 42(1):9–37, 2010.

[2] H. Bosse, J. Byrka, and E. Markakis. New algorithms for approximate Nash equilibria in bimatrix games. *THEOR COMPUT SCI*, 411(1):164–173, 2010.

[3] S. Ceppi, N. Gatti, G. Patrini, and M. Rocco. Local search methods for finding a Nash equilibrium in two-player games. In *IAT*, pages 335–342, 2010.

[4] S. Ceppi, N. Gatti, G. Patrini, and M. Rocco. Local search techniques for computing equilibria in two–player general–sum strategic–form games. In *AAMAS*, pages 1469–1470, 2010.

[5] X. Chen, X. Deng, and S. H. Teng. Computing Nash equilibria: approximation and smoothed complexity. In *FOCS*, pages 603–612, 2006.

[6] X. Chen, X. Deng, and S. H. Teng. Settling the complexity of computing two–player Nash equilibria. *J ACM*, 56(3):14:1–14:57, 2009.

[7] B. Codenotti, S. De Rossi, and M. Pagan. An experimental analysis of lemke-howson algorithm. *CoRR*, abs/0811.3247, 2008.

[8] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. In *STOC*, pages 71–78, 2006.

[9] D. Fudenberg and J. Tirole. *Game Theory*. The MIT Press, 1991.

[10] C. P. Gomes. *Constraint and Integer Programming: Toward a Unified Methodology*, chapter Randomized backtrack search, pages 233–283. Kluwer Academic Publisher, 2003.

[11] S. C. Kontogiannis and P. G. Spirakis. Polynomial algorithms for approximating Nash equilibria of bimatrix games. *THEOR COMPUT SCI*, 410(17):1599–1606, 2009.

[12] P. La Mura. Game networks. In *UAI*, pages 335–342, 2000.

[13] P. La Mura and M. R. Pearson. Simulated annealing of game equilibria: A simple adaptive procedure leading to Nash equilibrium. In *WLSDA*, 2000.

[14] C. E. Lemke and J. J. T. Howson. Equilibrium points of bimatrix games. *SIAM J APPL MATH*, 12(2):413–423, 1964.

[15] D. G. Luenberger and Y. Ye. *Linear and Nonlinear programming*, chapter The simplex method, pages 33–59. Stanford University, 2008.

[16] W. Michiels, E. Aarts, and J. Korst. *Theoretical Aspects of Local Search*. Springer, 2007.

[17] E. Nudelman, J. Wortman, K. Leyton-Brown, and Y. Shoham. Run the GAMUT: A comprehensive approach to evaluating game–theoretic algorithms. In *AAMAS*, pages 880–887, 2004.

[18] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. In *AAAI*, pages 664–669, 2004.

[19] T. Sandholm, A. Gilpin, and V. Conitzer. Mixed–integer programming methods for finding Nash equilibria. In *AAAI*, pages 495–501, 2005.

[20] R. Savani and B. von Stengel. Hard–to–solve bimatrix games. *ECONOMETRICA*, 74(2):397–429, 2006.

[21] D. A. Spielman and S. H. Teng. Smoothed analysis of algorithms and heuristics: progress and open questions. In *Foundations of Computational Mathematics*, pages 274–342, 2005.

[22] H. Tsaknakis and P. G. Spirakis. An optimization approach for approximate Nash equilibria. In *WINE*, pages 42–56, 2007.

[23] B. von Stengel. *Algorithmic Game Theory*, chapter Equilibrium computation for two–player games in strategic and extensive form, pages 53–78. Cambridge University Press, 2007.

[24] B. von Stengel, A. van den Elzen, and D. Talman. Computing normal form perfect equilibria for extensive two–person games. *ECONOMETRICA*, 70(2):693–715, 2002.

Generalized Belief Propagation on Tree Robust Structured Region Graphs

Andrew E. Gelfand
Dept. of Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA

Max Welling
Dept. of Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA

Abstract

This paper provides some new guidance in the construction of region graphs for Generalized Belief Propagation (GBP). We connect the problem of choosing the outer regions of a Loop-Structured Region Graph (SRG) to that of finding a fundamental cycle basis of the corresponding Markov network. We also define a new class of *tree-robust* Loop-SRG for which GBP on any induced (spanning) tree of the Markov network, obtained by setting to zero the off-tree interactions, is exact. This class of SRG is then mapped to an equivalent class of *tree-robust* cycle bases on the Markov network. We show that a tree-robust cycle basis can be identified by proving that for every subset of cycles, the graph obtained from the edges that participate in a single cycle only, is multiply connected. Using this we identify two classes of tree-robust cycle bases: planar cycle bases and "star" cycle bases. In experiments we show that tree-robustness can be successfully exploited as a design principle to improve the accuracy and convergence of GBP.

1 Introduction

Loopy belief propagation (BP) on Markov networks (MNs) [16, 29, 5, 22, 1, 17] has remained an active area of research for more than a decade. Much progress has been made in characterizing its fixed points [26, 33, 9], improving and understanding its convergence properties [26, 4, 11, 12], designing convergent alternatives [32, 35, 9, 7, 8, 19], developing new message passing algorithms based on alternative convex objectives [27, 28, 10, 25], and extending BP to GBP based on larger clusters (also known as the Kikuchi approximation or Cluster Variation Method) [14, 20, 34].

Despite all this progress, very little attention has been paid to the question of how to actually choose the outer regions (clusters, cliques) for any of the GBP algorithms. This is surprising given that GBP has the potential to improve the accuracy of BP by orders of magnitude without necessarily adding all that much computational burden. Moreover, GBP is very sensitive to the choice of outer regions; a bad choice can lead to poor convergence behavior and little improvement in accuracy, while a good choice can result in fast convergence and an accurate approximation.

Some guidance on choosing regions has appeared in the literature. In [30] a (greedy) region pursuit algorithm was proposed that ranks candidate clusters by sending a limited set of messages through the network. This approach is computationally expensive because all candidate clusters need to be evaluated. Moreover, the algorithm has no way of deciding when to stop adding regions. A similarly expensive "wrapper" approach based on the edge deletion method was published in [2]. A scheme that utilizes mini-bucket elimination to choose regions in a two layer region graph (called a join graph) is presented in [3]. In this region graph, inner regions represent the separators and connections between outer and inner regions are chosen so that the running intersection property holds.

Two desirable properties of region graphs on partially ordered sets (posets) - k-connectedness and k-balancedness - were proposed in [18, 23]. Those authors also show that the Cluster Variation Method (CVM) [14, 20, 34] yields a region graph that is totally connected and balanced and that the join-graph construction is only 1-connected and 1-balanced. Structured Region Graphs (SRGs) introduced in [31] also inherit the total connectedness and balancedness properties. Another desirable property of region graph constructions discussed in [34] is that the sum of all counting numbers should equal 1. This property ensures exact results for infinitely strong interactions. A condition called maxent-normality, which ensures that the free energy is maximal in the limit of zero interactions, was also introduced in [34]. Maxent-uniqueness was strengthened in [31] to guarantee that uniform beliefs are the unique fixed points of GBP.

In [31] it was shown that Loop-SRGs - a specific class of

SRG with loop outer regions - satisfy all the desirable conditions when the loop regions form a linearly independent set of size #edges - #nodes + 1 - i.e. a cycle basis. Our contribution can be seen as a refinement of the criteria proposed in [31]. In particular, we show that to satisfy the desirable conditions (e.g. non-singularity and sum of counting numbers equal 1), the set of loop regions must form a fundamental cycle basis. We also propose a condition called "tree-robustness" that further restricts the choice of loop regions. The idea of tree-robustness is to require GBP on a SRG to be *exact on every possible tree embedded in the Markov Network*, where an embedded tree is obtained by zeroing out the interactions on the off-tree edges. For loop SRGs, we show that this idea can be translated to an equivalent problem in graph theory, namely that of finding tree-robust cycle bases. We then characterize the class of tree robust cycle bases and identify two families of tree robust cycle bases. Finally, we demonstrate the performance of GBP on Loop-SRGs constructed to satisfy tree-robustness.

2 Generalized Belief Propagation and Structured Region Graphs

Belief Propagation (BP) [24] is an algorithm for computing marginal probabilities in distributions taking the following form:

$$p(\mathbf{x}) = \frac{1}{Z}\prod_a f_a(\mathbf{x}_a) \tag{1}$$

where a indexes the factors in the model and $f(\mathbf{x}_a)$ is a function on $\mathbf{x}_a$, which is a subset of $\{x_1, ..., x_n\}$, the n discrete-valued random variables in the distribution. In this paper we assume $p(\mathbf{x})$ to be comprised of only pairwise factors but the extension to factor graphs is straightforward.

BP is a message passing algorithm that computes marginals by iteratively sending messages from one node to its neighboring nodes. It is most easily described in terms of messages passed along the edges of a *factor graph* [15] (factors are pairwise interactions in this paper), which is a bipartite graph comprised of *factor* nodes and *variable* nodes. The edges in a factor graph connect each factor node a to the variable nodes i for which $x_i \in \mathbf{x}_a$. The belief $b_i(x_i)$ at variable node i is an approximation to the exact marginal $p(x_i)$ and is defined as:

$$b_i(x_i) \propto \prod_{a \in N(i)} m_{a \to i}(x_i) \tag{2}$$

where $m_{a\to i}(x_i)$ is a message from factor node a to variable node i and $N(i)$ is the set of factor nodes neighboring variable node i. The belief $b_a(\mathbf{x}_a)$ over the variables $\mathbf{x}_a$ is defined in an analogous fashion, as a product of messages from variable nodes to factor node a. These beliefs will be exact if the factor graph contains no cycles.

An important limitation of BP is that messages are defined on a single variable. This means that interactions between variables may be lost during message passing. Generalized Belief Propagation (GBP) [34] is a class of algorithms that address this limitation. In GBP, messages over one or more variable are passed among sets of nodes or *regions*. A *region* graph is a structure, analogous to the factor graph, that helps organize the computation of GBP messages.

DEFINITION **1.** *A Region R is a set of variable nodes and factor nodes such that if factor node a is in region R, all variable nodes i where $x_i \in \mathbf{x}_a$ are also in R.*

Let $\mathcal{R}$ denote the set of all regions. Let $\mathbf{x}_R$ denote the set of variables in region R and $f_R(\mathbf{x}_R) = \prod_{a\in R} f_a(\mathbf{x}_a)$ be the factors in region R. Every variable and factor must belong to some region.

DEFINITION **2.** *A Region Graph (RG) is a directed graph $G(V, E)$, where each vertex $v \in V$ is associated with a region $R \in \mathcal{R}$ and the directed edges $e \in E$ are from some vertex v_p to vertex v_c if $\mathbf{x}_{R_c} \subset \mathbf{x}_{R_p}$, where $\mathbf{x}_{R_i}$ is the set of variables in region R_i. In such cases, vertex (region) p is the parent of vertex c.*

Let $pa(R)$ denote the parents, $an(R)$ denote the ancestors and $de(R)$ denote the descendants of region R. Regions with no parents will be referred to as *outer* regions; all other regions are *inner* regions. The Bethe RG is a RG with outer regions for each factor and inner regions for each variable.

Each region R is associated with a counting number κ_r used to define the region-based free energy of a RG. Let $R(i) = \{R \in \mathcal{R} | x_i \in \mathbf{x}_R\}$ denote the set of regions containing variable i. A RG is considered *1-balanced* if:

$$\sum_{R\in R(i)} \kappa_R = 1 \; \forall \, i \tag{3}$$

1-balancedness is satisfied if κ_R is defined recursively as:

$$\kappa_R = 1 - \sum_{A\in an(R)} \kappa_A \tag{4}$$

A RG is *1-Connected* if the subgraph consisting of the regions $R(i)$ is connected for all i. These conditions can be strengthened by considering the set of regions containing a larger set of variables - i.e. $R(s) = \{R \in \mathcal{R} | \mathbf{x}_s \in \mathbf{x}_R\}$ for $\mathbf{x}_s \subseteq \{x_1, ..., x_n\}$. A RG is called *totally* connected and balanced if it is connected and balanced for all sets of variables that are subsets of an outer region - i.e. for any $\mathbf{x}_s \subseteq \mathbf{x}_R$ for outer region R [23]. Junction graphs [18] and Join Graphs [3] are two-layer RG constructions satisfying 1-balancedness and 1-connectednesss, while CVM [14, 21] ensures total connectivity and balancedness.

Many different GBP algorithms can be defined on RGs. The parent-to-child (or canonical) GBP algorithm is one such algorithm. It is defined in terms of messages sent from a parent region to a child region. As in BP, the belief at a particular region R is computed as a product of messages

into R. However, the messages do not come from the regions neighboring R, but rather from regions external to R. The belief $b_R(\mathbf{x}_R)$ at region R is given by:

$$b_R(\mathbf{x}_R) \propto f_R(\mathbf{x}_R) \prod_{P \in pa(R)} m_{P \to R}(\mathbf{x}_R) \cdot \quad (5)$$

$$\prod_{D \in de(R)} \prod_{P' \in P'(R)} m_{P' \to D}(\mathbf{x}_D) \quad (6)$$

where $P'(R) = \{pa(D) \setminus \{R \cup de(R)\}\}$. Figure 1 illustrates how beliefs are computed on a 3-by-3 grid.

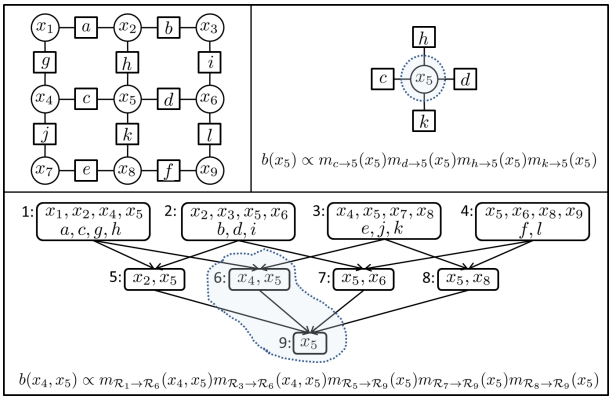


Figure 1: Illustration of message passing on a factor graph (top right) and region graph (bottom) for the 3-by-3 grid in the top-left. In BP, the belief $b_5(x_5)$ is a product of messages into variable node 5. The belief at region 6, $b_{R_6}(x_4, x_5)$, is a product of messages into region 6 and its descendants (i.e. region 9).

The RG framework was extended in [31] to structured message passing algorithms.

DEFINITION **3.** *A Structured Region Graph (SRG) is a region graph in which each region R is associated with a set of cliques $\mathcal{C}(R)$. Every variable in $\mathbf{x}_R$ must appear in some clique or factor. The set of factors and cliques associated with a region define a structure $\mathcal{G}(R)$, which is an undirected graph with vertices for each variable in $\mathbf{x}_R$ and edges connecting any pair of variables appearing in the same factor or clique.*

In the rest of the paper, we restrict our attention to Loop-SRGs, which are a particular type of SRG.

DEFINITION **4.** *A Loop-SRG is a 3-level SRG consisting of loop outer regions and edge and node inner regions, where a loop outer region has a structure $\mathcal{G}(R)$ that forms an (elementary) cycle. Loop outer regions are connected to the set of edge inner regions comprising the loop and edge inner regions are connected to the two node regions comprising the edge.*

A Loop-SRG on the grid in Figure 1 can be formed by taking the faces of the grid as the loop regions, the edges of the grid as edge regions and all vertices as node regions. Importantly, Loop-SRGs are not limited to planar graphs. In fact, a Loop-SRG can be constructed on any graph structure containing cycles.

3 Properties of SRGs

The balancedness and connectedness conditions can be satisfied by many different SRGs for some distribution $p(\mathbf{x})$. However, some SRGs will give better quality approximations than others. Two properties of SRGs that are useful in identifying "good" SRGs were identified in [34]. The first property is that $\sum_R \kappa_R = 1$. This property ensures that the region-based entropy is correct if the variables in $p(\mathbf{x})$ are perfectly correlated. This property will be referred to as *counting number unity*. The second property is *maxent-normality*, which requires the region-based entropy of a connected SRG to achieve its maximum when the region beliefs $b_R(\mathbf{x}_R)$ are uniform. The maxent-normality property was strengthened in [31] to require that message passing with uniform factors have a *unique* fixed point at which $b_R(\mathbf{x}_R)$ are uniform. This stronger property is referred to as *non-singularity*. An SRG that does not satisfy this condition is *singular*.

In [31] it was shown that an acyclic SRG is non-singular. Proving this required a set of reduction operators that modify an SRG's structure while preserving the fixed points of the region-based free energy. The reduction operators will be used in this paper, but are not presented for space reasons (see [31, 30]).

The following qualities of Loop-SRGs come from [31]:

THEOREM **1.** *A Loop-SRG has $\sum_R \kappa_R = |L| - |E| + |V|$, where $|L|$ is the number of loop regions, $|E|$ the number of edge regions and $|V|$ the number of node regions.*

THEOREM **2.** *A Loop-SRG is singular if $\sum_R \kappa_R > 1$.*

THEOREM **3.** *A Loop-SRG is singular iff there is a subset of loop regions and constituent edge regions such that all of the edge regions have 2 or more parents.*

These theorems are illustrated on the grid in Figure 2. In this Loop-SRG, every edge region has exactly 2 loop regions as parents and is thus singular by Theorem 3. There are also $|L| = 3$ loop regions, $|E| = 7$ edge regions and $|V| = 6$ node regions, so that $\sum_R \kappa_R = 2$. Thus, the Loop-SRG is also singular by Theorem 2.

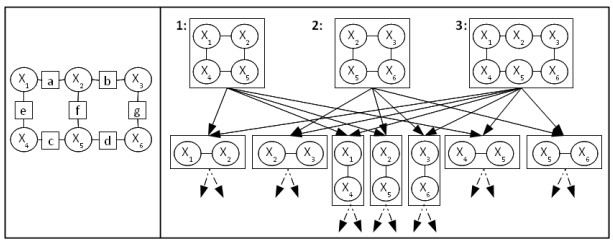


Figure 2: A Loop-SRG (right) for the 2-by-3 grid (left). There are 3 loop outer regions and 7 edge inner regions. The node regions have been dropped for clarity.

Theorems 2 and 3 describe conditions that are undesirable in Loop-SRGs, but offer no guidance on how to identify

"good" Loop-SRGs. The following section establishes a connection between the set of loop regions in a Loop-SRG and cycle bases of a graph that will prove useful in constructing well-behaved Loop-SRGs.

4 Loop Regions and Cycle Bases

In [31] it was noted that counting number unity in a Loop-SRG requires $|L| = |E| - |V| + 1$ loops. This number of loops is exactly the dimension of the cycle space of the undirected graph G describing the MN $p(\mathbf{x})$. The connection between loop regions and cycle bases is formalized in this section. We begin with some background on cycle bases taken from [13].

Let $G = (V, E)$ be a 2-connected graph. A simple cycle C in G is a connected Eulerian subgraph in which every vertex has degree 2. The cycle space $\mathfrak{C}(G)$ of a graph G consists of all simple and non-simple cycles of G including the empty cycle $\emptyset$. The dimension of a cycle space for a graph with 1 connected component is $\mu \equiv \mu(G) = |E| - |V| + 1$.

DEFINITION **5.** *A Cycle Basis of $\mathfrak{C}(G)$ is a set of simple cycles $\mathcal{B} = \{C_1, ..., C_\mu\}$ such that for every cycle C of G, there exists a unique subset $\mathcal{B}_C \subseteq \mathcal{B}$ such that the set of edges appearing an odd number of times in $\mathcal{B}_C$ comprise the cycle C.*

DEFINITION **6.** *A cycle basis $\mathcal{B}$ is a Fundamental Cycle Basis (FCB) if there exists a permutation π of the cycles in $\mathcal{B}$ such that $C_{\pi(i)} \setminus \{C_{\pi(1)} \cup \cdots \cup C_{\pi(i-1)}\} \neq \emptyset$ for $i = 2...\mu$. In other words, the cycles can be ordered so that cycle $C_{\pi(i)}$ has some edge that does not appear in any cycle preceding it in the ordering.*

Consider mapping each loop outer region R to a cycle C in the basis $\mathcal{B}$ of the undirected graph of $p(\mathbf{x})$.[1] Using this mapping, we can claim the following.

THEOREM **4.** *A Loop-SRG is Non-Singular and satisfies Counting Number Unity if its loop outer regions are a Fundamental Cycle Basis (FCB) of G.*

The proof[2] uses reduction operators to show that a loop outer region with a unique edge (i.e. an edge not shared with any other loop region) can be *reduced* to the set of edges comprising that loop. Since the set of loops form a FCB, we are guaranteed to find a loop region with a unique edge if we reduce loops along π - i.e. beginning with the loop corresponding to $C_{\pi(\mu)}$ and ending at loop $C_{\pi(1)}$.

This result implies that the loop regions in an SRG should form a FCB. This greatly reduces the set of loops considered when constructing a Loop-SRG. However, a graph may have many fundamental bases, so one may ask if Loop-SRGs formed from certain FCBs are better than others. The next section defines a class of Loop-SRG with loop regions corresponding to a specific type of FCB.

[1] More specifically, the structure $\mathcal{G}(R)$ of each outer region R is chosen to be a unique cycle $C \in \mathcal{B}$

[2] Formal proofs of all theorems in this paper are provided as supplementary material.

5 Tree Robust Cycle Bases

Non-singularity and counting number unity ensure that GBP behaves sensibly in two opposite and extreme situations. Non-singularity requires GBP to be exact in the presence of uniform factors, while counting number unity ensures that GBP is exact in the presence of perfectly correlated variables.

Tree-Robustness is a condition which ensures that GBP behaves sensibly in an orthogonal way. The idea of *Tree-Robustness* is as follows. Imagine removing some set of factors from $p(\mathbf{x})$ to produce a modified MN $p'(\mathbf{x})$ that is acyclic. It would be nice if GBP run on the original Loop-SRG of $p(\mathbf{x})$ but retaining only the factors of $p'(\mathbf{x})$, were equivalent to BP run directly on the Bethe RG for $p'(\mathbf{x})$ (since BP will be exact in that case). A Tree Robust SRG is a Loop-SRG that is exact w.r.t. all acyclic MNs produced by retaining some of the factors in $p(\mathbf{x})$.

More formally, let T be some spanning tree of the undirected graph G describing $p(\mathbf{x})$ and let $\bar{T}$ denote the off-tree edges. A SRG is Tree Exact w.r.t. to T if by removing only factors on edges $\bar{T}$, the SRG can be *reduced* so that inference on T is exact. In other words, a sequence of reduction operations can be applied to the SRG so that it can be reduced to a Bethe RG with edge regions for each edge in T and node regions for each variable. Since T is acyclic, running GBP on this resulting RG will be exact. Finally, we say that a SRG is Tree Robust (TR) if it is Tree Exact w.r.t. all spanning trees of G.

The previous section established that a Loop-SRG is non-singular if its loops form a FCB. In this section, we first define a TR cycle basis and then show that a Loop-SRG is TR if its loops form a TR cycle basis.

DEFINITION **7.** *Let T be some spanning tree of G. A cycle basis $\mathcal{B}$ is Tree Exact w.r.t. T if there exists an ordering π of the cycles in $\mathcal{B}$ such that $\{C_{\pi(i)} \setminus \{C_{\pi(1)} \cup \cdots \cup C_{\pi(i-1)}\}\} \setminus T \neq \emptyset$ for $i = 2...\mu$. In other words, the cycles can be ordered so that cycle $C_{\pi(i)}$ has some edge that: 1) does not appear in any cycle preceding it in the ordering; and 2) does not appear in the spanning tree T.*

DEFINITION **8.** *A cycle basis $\mathcal{B}$ is Tree Robust (TR) if it is Tree Exact w.r.t. all spanning trees of G.*

By mapping each loop outer region to a cycle in basis $\mathcal{B}$ it can be shown that[3]:

[3] See supplementary material for formal proof.

THEOREM **5.** *A Loop-SRG is Tree Robust if its loop outer regions are a Tree Robust cycle basis of G.*

In the remainder of this section, we introduce two theorems that characterize TR cycle bases. These results prove useful in showing that, for example, the faces of a planar graph constitute a TR cycle basis. The ensuing theorems require the following definition.

DEFINITION **9.** *The Unique Edge Graph of a set of cycles $\mathcal{C} = \{C_1, ..., C_k\}$ is a graph comprised of the set of edges that are in exactly one cycle in $\mathcal{C}$. We will use $I(\mathcal{C})$ to denote the unique edge graph for cycle set $\mathcal{C}$. $I(\mathcal{C})$ is* cyclic *if it contains at least one cycle.*

Using this definition we can now state the following equivalent characterization of TR cycle bases.

THEOREM **6.** *Let $\mathfrak{B}^{|k|}$ denote all size k subsets of cycles in $\mathcal{B}$. A FCB $\mathcal{B}$ is Tree Robust iff $I(\mathcal{B}_k)$ is cyclic and not empty for all $\mathcal{B}_k \in \mathfrak{B}^{|k|}$ for $1 \leq k \leq \mu$. In other words, the unique edge graph must be cyclic for all pairs of cycles, and all triples of cycles,..., and all of the μ cycles.*

COROLLARY **1.** *An FCB is TR iff $\mathcal{B}_k$ is TR for all $\mathcal{B}_k \in \mathfrak{B}^{|k|}$ for $1 \leq k \leq \mu$.*

We sketch the main idea of the proof here. Sufficiency follows because whatever ordering π we choose to remove the cycles in, there is never a tree T that can block all of the edges of the unique edge graph under consideration (since it is always cyclic). Necessity follows because if there were a subset of cycles for which the exposed edge graph is acyclic then there exists no ordering π that can avoid that subset (or a larger subset with an acyclic unique edge graph). Hence, by choosing the tree T to block all edges of the acyclic unique edge graph under consideration we prove that the bases is not tree robust.

6 Planar and Complete Graphs

Using the theorems from the previous section, we now identify TR cycle bases of planar and complete graphs. In the case of planar graphs, a TR basis can be constructed from the cycles forming the faces of the planar graph. This supports the observation of previous authors that GBP run on the faces of planar graphs gives accurate results.

THEOREM **7.** *Consider a planar graph G. The cycle basis $\mathcal{B}$ comprised of the faces of G is TR.*

Proof. We use theorem 6. Consider the graph formed by any subset of k faces of the planar graph. Consider any of its connected components. The path that traces the circumference of that component is a loop and also consists of unique edges. □

In the case of complete graphs, a TR basis can be constructed by choosing some vertex as a root, creating a spanning tree with edges emanating from that root and constructing loops of length 3 using each edge not in the spanning tree. This is exactly the 'star' construction of [31], which was shown empirically to be superior to other Loop-SRGs on complete graphs.

THEOREM **8.** *Consider a complete graph G on n vertices (i.e. K_n). Construct a cycle basis $\mathcal{B}$ as follows. Choose some vertex v as the root. Create a 'star' spanning tree rooted at v (i.e. with all edges v-u). Now construct cycles of the form v-i-j from each off-tree edge i-j. The basis $\mathcal{B}$ constructed in this way is TR.*

Proof. We use again theorem 6. Consider the graph constructed from any subset of k triangles. The edges not on the spanning tree (i.e. not connecting to the root) are all unique and will either form a loop (in which case we are done) or a tree. In case of a tree, consider a path connecting two leaf nodes, which are both also connected to the root, thus forming a loop. Because the two edges connecting to the root are also unique (since they correspond to leaf nodes) we have proven the existence of a cycle in the unique edge graph. □

This result can be extended to partially complete graphs where there exists some vertex v that is connected to all the other vertices in G.

7 TR SRGs in General Graphs

The previous section identified TR cycle bases for two specific classes of graphs. The prescription for how to construct TR SRGs for MNs on these types of graphs is clear: simply find a TR basis $\mathcal{B}$ of the underlying graph and make the cycles in $\mathcal{B}$ the loop regions. This prescription can be generalized in some cases to graphs containing many TR components. More formally,

THEOREM **9.** *Consider a graph G comprised of components (subgraphs) $H_1, ..., H_k$. Let the components $H_1, ..., H_k$ be mutually singly connected if for any two components H_i and H_j there exist vertices $v_i \in H_i$ and $v_j \in H_j$ that are singly connected (i.e. connected through a single path). Let $\mathcal{B}_{H_1}, ..., \mathcal{B}_{H_k}$ denote the TR cycle bases for each component. Then a TR cycle basis of G is $\mathcal{B}_G = \{\mathcal{B}_{H_1} \cup \cdots \cup \mathcal{B}_{H_k}\}$.*

Proof. First note that by singly connecting the components $H_1, ..., H_k$ we do not create any new cycles. Thus $\mathcal{B}_G$ is a cycle basis of G. Every subset of cycles of $\mathcal{B}_G$ is the union of some subset of cycles from the component cycle bases $\{\mathcal{B}_{H_i}\}$. Moreover, for any of these subsets the unique edge graph must be cyclic (by theorem 6). Since the component cycle bases do not overlap it thus follows that the unique edge graph for every subset of cycles of $\mathcal{B}_G$ must also be cyclic. □

For MNs defined over more general graphs, the picture of how to construct a TR SRG is less clear. Since verifying

that a basis is TR requires inspecting all subsets of cycles in that basis, searching for a TR basis in a general graph seems difficult. Moreover, not every graph will admit a TR basis. In this section we describe a method for constructing Loop-SRGs that are partially TR. In other words, we sacrifice finding a TR basis of G to find a basis that is Tree Exact for many (just not all) spanning trees of G.

The method for finding a partially TR basis works as follows. We first find the largest complete or planar subgraph H of G and construct a TR basis $\mathcal{B}(\mathcal{H})$ for H as described in the previous section. Since the TR core H is a subgraph of G, the $\mu(H)$ cycles in $\mathcal{B}(\mathcal{H})$ will not form a complete basis of G. We choose the remaining $\mu(G) - \mu(H)$ cycles so that the basis of G is fundamental. We do so by finding a sequence of *ears* (simple paths or cycles) in G, such that each new *ear* has some edge not occurring in some previous ear. This process is described in Algorithm *Construct Basis* and is illustrated on a 2×3 grid in Figure 3.

Algorithm: Construct Basis

Input: MN $p(\mathbf{x})$ described by undirected graph G
Output: Cycle Basis $\mathcal{B}$

Find TR subgraph H of G with TR basis $\mathcal{B}(\mathcal{H})$
if G contains no TR subgraph (i.e. $H = \emptyset$) **then**
  Let H be some simple cycle in G
end if
Add cycles $\mathcal{B}(\mathcal{H})$ to $\mathcal{B}$
Mark all edges in H as *used*
Mark all vertices in H as *visited*
while $\exists$ *unused* edge $e = (s, t)$ from a *visited* vertex s **do**
  If t is *visited*, then set $p_1 = e$
  Else, find an ear p_1 from s through edge $e = (s, t)$ to some *visited* vertex u.
  Find shortest path p_2 from s to u on *used* edges
  Add cycle C consisting of $p_1 \cup p_2$ to the bases $\mathcal{B}$
  Mark all edges (vertices) on C as *used* (*visited*)
end while

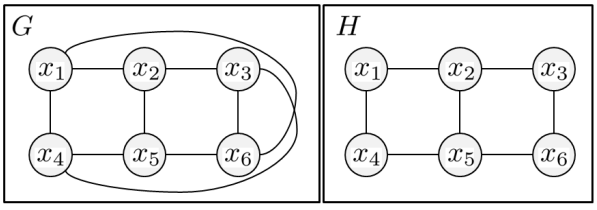


Figure 3: A graph G (left) and its TR core H (right). Add the faces $(1, 2, 5, 4)$ and $(2, 3, 6, 5)$ of H to $\mathcal{B}$. Mark all edges (vertices) in H as used (visited). Edge $(1, 6)$ is an ear (unused edge). The path $6, 3, 2, 1$ connects the start and end vertices of this ear along used edges, so cycle $(1, 6, 3, 2)$ is added to $\mathcal{B}$. Similarly, cycle $(3, 4, 5, 6)$ might be added for ear $(3, 4)$, giving us the partially TR basis $\mathcal{B} = \{(1, 2, 5, 4), (2, 3, 6, 5), (1, 6, 3, 2), (3, 4, 5, 6)\}$.

8 Experiments

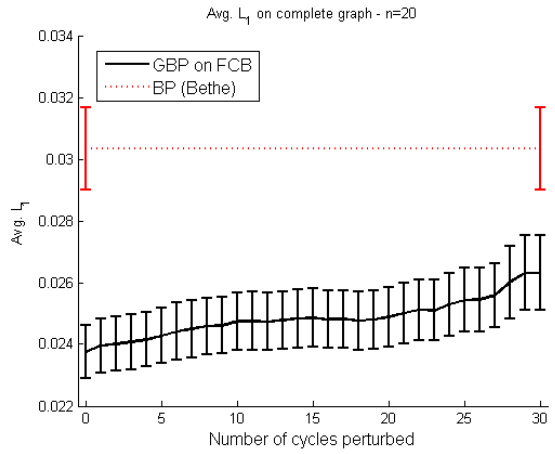
We conducted a series of experiments to validate the recommendations for constructing Loop-SRGs made in this paper. In each of the experiments that follow, we generated a set of M random MNs with binary variables. Each MN instance has unary potentials of the form $f_i(x_i) = [\exp(h_i); \exp(-h_i)]$ and pairwise potentials of the form $f_{ij}(x_i, x_j) = [\exp(w_{ij}) \exp(-w_{ij}); \exp(-w_{ij}) \exp(w_{ij})]$. The values of h_i and w_{ij} were drawn from Normal distributions $\mathcal{N}(0, \sigma^2_{h_i})$ and $\mathcal{N}(0, \sigma^2_{w_{ij}})$, respectively. Two different error measures are reported. $Error_Z = |\log Z - \log \tilde{Z}|$ is the absolute error in the exact (Z) and approximate ($\tilde{Z}$) values of the log partition function. $Error_{L_1}$ measures the error in the marginal probabilities and is computed as $Error_{L_1} = \frac{1}{n} \sum_{i=1}^{n} |b(x_i) - p(x_i)|$ where n is number of variables in a MN instance. Averages of $Error_Z$ and $Error_{L_1}$ were computed across the M MN instances. Error bars indicate standard error. To aid convergence, GBP was run with a damping factor of 0.5 for at most 1000 iterations.

8.1 TR Bases vs non-TR Bases

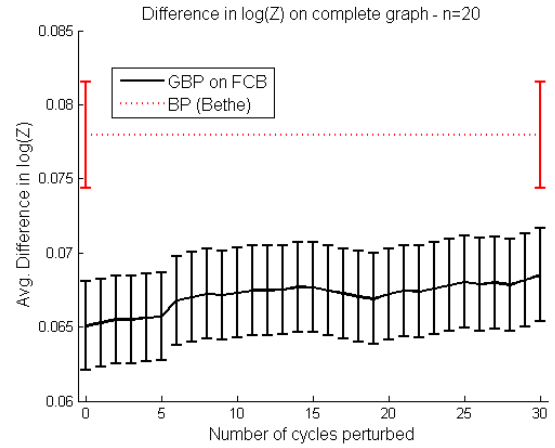
We first ran a set of experiments to show that tree robustness is a desirable property. To test this hypothesis, we generated 31 different Loop-SRGs for a MN defined on a complete graph with 20 binary variables (i.e. on K_{20}). We first constructed a TR basis $\mathcal{B}$ comprised of all cycles of length 3 passing through vertex 1 (e.g. using the *star* construction from Section 6 with vertex 1 as root). From this TR basis, a sequence of 30 fundamental bases were created as follows: for $i = 1...30$, we choose a cycle C_i of the form $C_i = (1, u, v)$ from $\mathcal{B}$ and modify it by swapping vertex 1 with some vertex w ($w \neq u \neq v \neq 1$) so that $C_i = (w, u, v)$. At every iteration we choose cycles that have not been modified and reject modifications that make the basis non-fundamental. In this way, as i increases the basis is made less TR but always remains fundamental.

Figure 4 shows $Error_{L_1}$ and $Error_Z$ as the Loop-SRG is made less TR. In this figure, we generated $M = 500$ random MN instances with $\sigma_{h_i} = 1$ and $\sigma_{w_{ij}} = 1/\sqrt{20 - 1}$. Note that by keeping the cycle length at 3 and ensuring that each basis is fundamental, the increase in error can only be explained by the change in the TR core of the basis. Since error increases as the basis is made less TR, these results support the hypothesis that tree robustness is a desirable property. It was also observed that GBP took an increasing number of iterations to converge as the basis was made less TR.

We also ran a set of experiments to characterize the convergence properties of TR SRGs on Ising grid models. Though not reported, these experiments confirm the finding in [34] that running GBP on an RG where the outer regions are



(a) $Error_{L_1}$ on increasingly non-TR Loop SRGs.



(b) $Error_Z$ on increasingly non-TR Loop SRGs.

Figure 4: Performance of GBP run on a sequence of increasingly non-TR Loop-SRGs.

taken as the faces of the grid model is more accurate than ordinary BP. We also found the TR SRG construction to be more accurate than GBP run on Loop-SRGs constructed from fundamental, but non-TR cycle bases on Ising grids.

8.2 Partially TR SRGs

The previous experiment considered a class of MNs for which a TR basis is known. This section considers more general MNs.

The algorithm in Section 7 seeks an initial TR subgraph H of the graph G. The previous experiments showed that GBP yields accurate approximations when $H = G$. When a graph contains no TR core (i.e. $H = \emptyset$) the algorithm *Construct Basis* method simply finds a fundamental basis of G. We wish to study the performance of GBP between these two extremes - i.e. on partially TR SRGs. We conducted experiments on two types of MNs, where the size of H (relative to G) can be controlled.

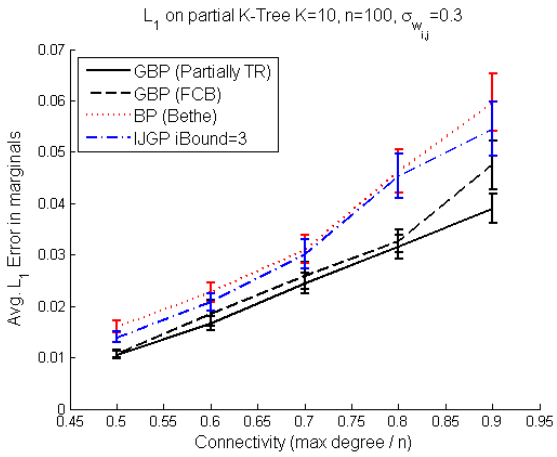
The following experiments include a comparison to the Iterative Join Graph Propagation (IJGP) method [3]. This method forms a join graph (i.e. a 1-connected and 1-balanced RG) using a heuristic, "mini-bucket" clustering strategy. In IJGP, the number of variables appearing in an outer region is restricted to be less than or equal to an *iBound* parameter. The *iBound* controls the computational complexity of message computations in IJGP because in the join graph construction each outer region forms a clique over all variables in that region. Importantly, since Loop-SRGs assume a loop structure in the outer regions, message computations on Loop-SRGs are equivalent to IJGP with $iBound = 3$.

8.2.1 Partial K-Trees

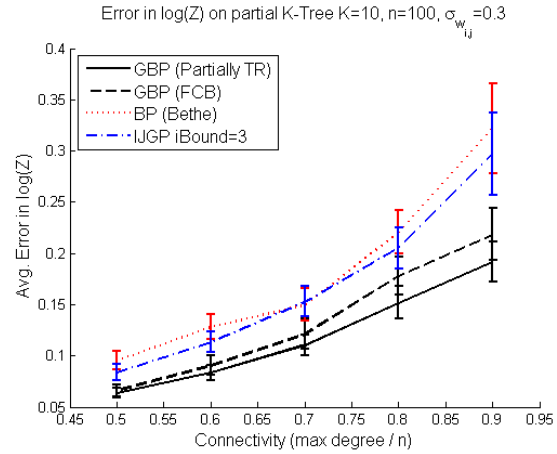
In these experiments we construct a set of partial K-tree instances via the following procedure[4]. We first build a random K-tree on n vertices using the process described in [6]. The number of neighbors (degree) of the vertices in K-trees constructed by this procedure follow a power law. This means there will exist a few vertices that are adjacent to most of the vertices in G. As a result, the TR core will comprise a large portion of G. To reduce the size of the TR core, we iteratively remove edges from the K-tree as follows: choose a vertex v with probability proportional to the current degree of that vertex. Modify G by removing an edge from v to one of its neighbors, so long as removing that edge does not disconnect G. This process is repeated until the ratio of the maximum degree in G to n falls below some threshold. We refer to this ratio as the connectivity of the graph. A random MN is formed over each partial K-tree structure by assigning random unary and pairwise potentials to the vertices and edges.

Figure 5 shows the performance of GBP as a function of connectivity. In these figures, we generated $M = 100$ random MN instances at each level of connectivity (with $K = 10$, $n = 100$, $\sigma_{h_i} = 1$ and $\sigma_{w_{ij}} = 0.3$). The partially TR SRGs are found by choosing the TR core to be the subgraph H found using the max degree vertex as the root of the *star* construction described in Section 6. Cycles are added to this TR core as described in algorithm *Construct Basis*. The partially TR SRGs are compared to Loop-SRGs formed by finding a fundamental cycle basis (FCB) of each partial K-tree (constructed using algorithm "*Construct Basis*" with $H = \emptyset$). Importantly, these FCBs do not build upon the TR core. Figure 5 shows that the benefit of the partially TR SRG diminishes as connectivity is decreased. This behavior confirms our belief that the benefit of finding a TR core decreases as the TR core comprises a smaller proportion of cycles in the fundamental basis. Even so, it is important to note that choosing a Loop-SRG with outer regions forming a FCB yields more accurate approximations than both IJGP and BP.

[4] K-trees are chordal graphs w/ size K maximal cliques. Partial K-trees are non-chordal graphs w/ size K maximal cliques

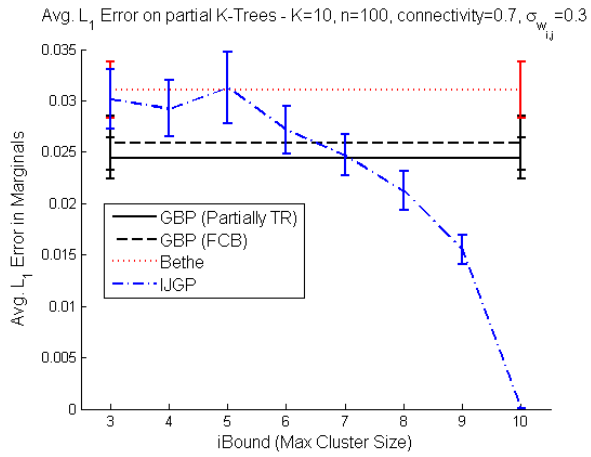


(a) $Error_{L_1}$ as a function of *connectivity*.

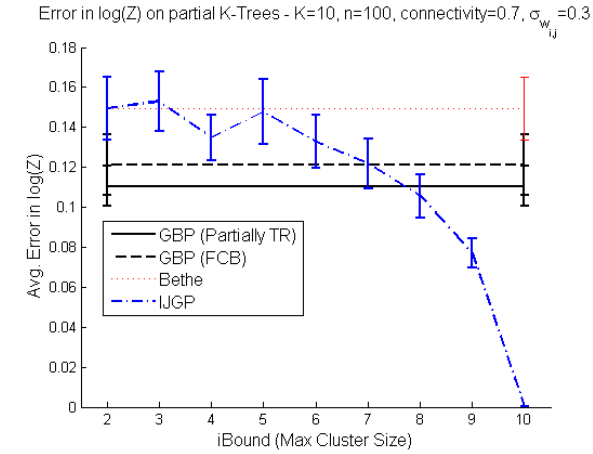


(b) $Error_Z$ as a function of *connectivity*.

Figure 5: Performance of GBP run on the partial TR construction, the FCB construction and IJGP with $iBound = 3$ as a function of *connectivity* (see text for discussion).



(a) $Error_{L_1}$ as a function of $iBound$.



(b) $Error_Z$ as a function of $iBound$.

Figure 6: Performance of the SRG constructions as a function of $iBound$ with fixed *connectivity* of 0.7.

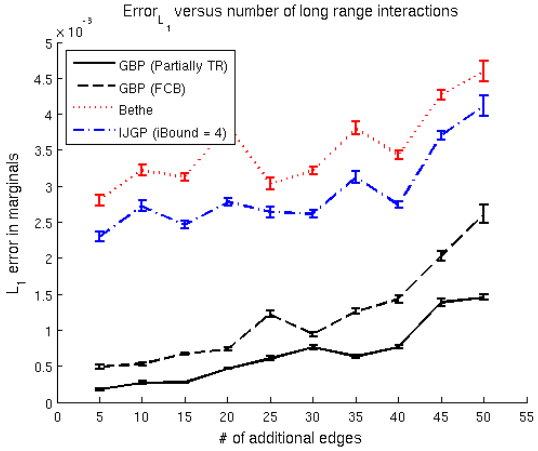
Figure 6 shows the performance of GBP as a function of $iBound$ for a fixed connectivity level. $iBound$ is increased from 2 (which is equivalent to BP) to 10 (which is exact). These plots show that GBP run on the partial TR SRG is roughly equivalent to IJGP with $iBound = 7$, while GBP run on the FCB is equivalent to IJGP with $iBound = 6$. The fact that GBP run on both SRGs performs better than IJGP with $iBound = 6$ is quite remarkable considering that message passing on Loop-SRGs is computationally equivalent to IJGP with $iBound = 3$.

8.2.2 Grids with long range interactions

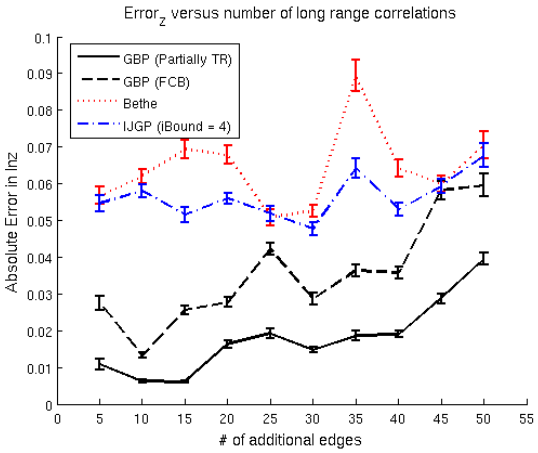
In addition to the partial K-tree instances, we also considered grid instances with an increasing number of long range interactions. The additional interactions were added via the following procedure. We begin with a 10×10 grid. Let G_0 denote this initial graph. Two vertices u and v are randomly chosen from the grid. If edge (u, v) exists in graph G_{i-1}, new vertices u and v are chosen randomly; if edge (u, v) is not in G_{i-1}, then graph G_i is created by adding edge (u, v) to G_{i-1}. This process is repeated until a specified number of edges have been added to G_0.

Figure 7 compares the $Error_{L_1}$ and $Error_Z$ of GBP run on different loop SRG constructions. $M = 25$ instances were generated with 5,10,.., and 50 additional edges. In all 250 of these MN instances the unary and pairwise terms were drawn with $\sigma_{h_i} = 1$ and $\sigma_{w_{ij}} = 0.5$. For the partially TR SRG construction, we take the TR core H to be G_0 and fill out the cycle basis using algorithm *Construct Basis* (as illustrated in Figure 3). As in the partial K-tree experiments, for the FCB construction we choose a fundamental basis that does not build upon the TR core by using algorithm "*Construct Basis*" with $H = \emptyset$. In Figure 7, we see that both the partially TR and FCB construction outperform IJGP with an equivalent $iBound$. Interestingly, when adding 50 additional edges we do not see the $Error_{L_1}$ of the partially TR and the FCB constructions coalesce. This may be explained by the fact that even with 50 additional edges more than 60% of the loops in the SRG are TR (131

cycles in the basis, 81 of which come from the TR core).



(a) $Error_{L_1}$ on grids with long range interactions



(b) $Error_Z$ on grids with long range interactions

Figure 7: Performance of the different SRG constructions as an increasing number of long range interactions are added to a 10×10 grid.

9 Conclusion

This paper provides some new guidance in the construction of region graphs. In particular, we connected the problem of choosing the loop outer regions of a Loop-SRG to that of finding a fundamental cycle basis of the undirected graph describing a MN. We proposed tree robustness as a refinement to the criterion that the loop regions form a fundamental basis and offered a characterization of the class of TR cycle bases. We identified TR cycle bases for planar and complete graphs. This characterization helps explain the success of GBP on the "star" construction of [31] for complete graphs and the "all faces" construction on planar graphs. We also proposed a practical Loop-SRG construction that first identifies a TR core and then expands to a full cycle basis using an ear construction which makes sure the final basis is still fundamental (and partially TR).

The experiments in this paper confirm that GBP can yield very accurate approximations when the loop regions of a Loop-SRG form a fundamental basis and that these approximations can be further improved by choosing a fundamental basis that is at least partially tree robust. The criteria proposed in this paper also lead to approximations that are comparable to IJGP run with a much higher $iBound$ (and therefore much higher space and computational complexity).

These findings open the door for much future work. Rather than simply finding a TR subgraph, as the method in Section 7 does, it would be preferable to have an algorithm that searches for TR bases in a graph or at the very least identifies when a graph does not admit a TR basis. The recommendations in this paper are also purely structural in nature. A natural extension would be to incorporate interaction strengths in the search for suitable loop regions.

The current paper considered pairwise interactions only. A natural extension is to consider factor graphs or more generally region graphs. We argue that we should be looking for a maximal collection of subsets $\{C_i\}$ of outer regions (or factors) that have the property that there exists an ordering π for which $C_{\pi(i)} \setminus \{C_{\pi(1)} \cup \cdots \cup C_{\pi(i-1)}\} \neq \emptyset$. In other words, the subsets can be ordered so that subset $C_{\pi(i)}$ has some element that does not appear in any subset preceding it in the ordering. This definition is identical to the one for a fundamental cycle basis (see definition 6) and will guarantee through the reduction rules of [31] that the region graph (factor graph) can be decomposed into independent variable nodes, establishing non-singularity. A next step would then be to define tree-robustness as a collection of region-subsets that can still be decomposed along some ordering if we do not allow certain elements that correspond to the regions of any embedded junction tree to become unique. Again this is very similar to definitions 7 and 8 for TR cycle bases. Whether these generalizations can be captured with mathematical structures as elegant as the theory of cycle spaces (or more generally matroids) remains to be seen and will be left for future research.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0914783, 0928427, 1018433.

References

[1] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding: turbo codes. *IEEE Transactions on Communications*, 44:1261–1271, 1996.

[2] A. Choi and A. Darwiche. An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 1107–1114, 2006.

[3] R. Dechter, K. Kask, and R. Mateescu. Iterative join-graph propagation. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.

[4] G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings, UAI*, July 2006.

[5] B.J. Frey. *Graphical models for machine learning and digital communication*. MIT Press, 1998.

[6] Y. Gao. The degree distribution of random k-trees. *Theor. Comput. Sci.*, 410:688–695, 2009.

[7] A. Globerson and T. Jaakkola. Convergent propagation algorithms via oriented trees. In *Uncertainty in Artificial Intelligence (UAI)*, 2007.

[8] T. Hazan and A. Shashua. Convergent message-passing algorithms for inference over general graphs with convex free energy. In *The 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

[9] T. Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy. In *Neural Information Processing Systems*, volume 15, Vancouver, CA, 2003.

[10] T. Heskes. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Machine Learning Research*, 26(153-190), 2006.

[11] Alexander T. Ihler, John W. Fisher, and Alan S. Willsky. Message errors in belief propagation. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Neural Information Processing Systems 17*, pages 609–616. MIT Press, Cambridge, MA, 2005.

[12] Mooij J, M and H.J. Kappen. Sufficient conditions for convergence of loopy belief propagation. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[13] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, and K.A. Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199–243, 2009.

[14] R. Kikuchi. A theory of cooperative phenomena. *Physical Review*, 81(6):988–1003, 1951.

[15] F.R. Kschischang, B. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[16] D.J.C. MacKay and R.M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 32:1645–1646, 1996.

[17] R. McEliece, D. MacKay, and J. Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE J. Selected Areas in Communication, 1997*, 16:140–152, 1998.

[18] R. McEliece and M. Yildirim. Belief propagation on partially ordered sets. In eds. D. Gilliam and J. Rosenthal, editors, *Mathematical Systems Theory in Biology, Communications, Computation, and Finance*, 1998.

[19] T. Meltzer, A. Globerson, and Y. Weiss. Convergent message passing algorithms - a unifying view. In *Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 393–401, 2009.

[20] T. Morita. Consistent relations in the method of reducibility in the cluster variation method. *Journal Statistical Physics*, 34:319–328, 1984.

[21] T. Morita. Formal structure of the cluster variation method. *Progress of Theoretical Physics supplement*, 115:27–39, 1994.

[22] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, 1999.

[23] Payam Pakzad and Venkat Anantharam. Estimation and marginalization using kikuchi approximation methods. *Neural Computation*, 17:1836–1873, 2005.

[24] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, California, 1988.

[25] D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *Neural Information Processing Systems*, 2007.

[26] M.J. Wainwright, T. Jaakkola, and A.S. Willsky. Tree-based reparameterization for approximate estimation on loopy graphs. In *Advances Neural Information Processing Systems*, volume 14, vancouver, Canada, 2001.

[27] M.J. Wainwright, T. Jaakkola, and A.S. Willsky. A new class of upper bounds on the log partition function. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, Edmonton, CA, 2002.

[28] M.J. Wainwright and M.I. Jordan. Semidefinite relaxations for approximate inference on graphs with cycles. In *Neural Information Processing Systems*, 2004. Rep. No. UCB/CSD-3-1226.

[29] Y. Weiss. Interpreting images by propagation Bayesian beliefs. In *Neural Information Processing Systems*, pages 908–914, 1996.

[30] M. Welling. On the choice of regions for generalized belief propagation. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 585–592, 2004.

[31] M. Welling, Tom Minka, and Yee Whye Teh. Structured region graphs: Morphing EP into GBP. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 607–614, 2005.

[32] M. Welling and Y.W. Teh. Belief optimization for binary networks: a stable alternative to loopy belief propagation. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 554–561, 2001.

[33] J.S. Yedidia, W. Freeman, and Y. Weiss. Bethe free energy, kikuchi approximations, and belief propagation algorithms. Technical report, MERL, 2001. Technical Report TR-2001-16.

[34] J.S. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical report, MERL, 2002. Technical Report TR-2002-35.

[35] A.L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, 2002.

Exploiting compositionality to explore a large space of model structures

Roger B. Grosse
Comp. Sci. & AI Lab
MIT
Cambridge, MA 02139

Ruslan Salakhutdinov
Dept. of Statistics
University of Toronto
Toronto, Ontario, Canada

William T. Freeman
Comp. Sci. & AI Lab
MIT
Cambridge, MA 02139

Joshua B. Tenenbaum
Brain and Cognitive Sciences
MIT
Cambridge, MA 02193

Abstract

The recent proliferation of richly structured probabilistic models raises the question of how to automatically determine an appropriate model for a dataset. We investigate this question for a space of matrix decomposition models which can express a variety of widely used models from unsupervised learning. To enable model selection, we organize these models into a context-free grammar which generates a wide variety of structures through the compositional application of a few simple rules. We use our grammar to generically and efficiently infer latent components and estimate predictive likelihood for nearly 2500 structures using a small toolbox of reusable algorithms. Using a greedy search over our grammar, we automatically choose the decomposition structure from raw data by evaluating only a small fraction of all models. The proposed method typically finds the correct structure for synthetic data and backs off gracefully to simpler models under heavy noise. It learns sensible structures for datasets as diverse as image patches, motion capture, 20 Questions, and U.S. Senate votes, all using exactly the same code.

1 Introduction

There has been much interest recently in learning hierarchical models, which extend simpler models by introducing additional dependencies between the parameters. While there have been many advances in modeling particular kinds of structure, as the desired structure becomes higher level and more abstract, the correct model becomes less obvious a priori. We aim to determine an appropriate model structure automatically from the data, in order to make hierarchical modeling usable by non-experts and to explore a wider variety of structures than would be possible by manual engineering.

There has been much work on structure learning in particular model classes, such as undirected (Lee et al., 2006) and directed (Teyssier and Koller, 2005) graphical models. Most such work focuses on determining the particular factorization and/or conditional independence structure within a fixed model class. Our concern, however, is with identifying the overall form of the model. For instance, suppose we are interested in modeling the voting patterns of U.S. Senators. We can imagine several plausible modeling assumptions for this domain: e.g., that political views can be summarized by a small number of dimensions, that Senators cluster into voting blocks, or that votes can be described in terms of binary attributes. Choosing the correct assumptions is crucial for uncovering meaningful structure. Right now, the choice of modeling assumptions is heavily dependent on the intuition of the human researcher; we are interested in determining appropriate modeling assumptions automatically.

Many common modeling assumptions, or combinations thereof, can be expressed by a class of probabilistic models called matrix decompositions. In a matrix decomposition model, component matrices are first sampled independently from a small set of priors, and then combined using simple algebraic operations. This expressive model class can represent a variety of widely used models, including clustering, co-clustering (Kemp et al., 2006), binary latent factors (Griffiths and Ghahramani, 2005), and sparse coding (Olshausen and Field, 1996). Nevertheless, the space of models is *compositional*: each model is described recursively in terms of simpler matrix decomposition models and the operations used to combine them. We propose to exploit this compositional structure to efficiently and generically evaluate and perform inference in matrix decomposition models, and to automatically search through the space of structures to find one appropriate for a dataset.

A common heuristic researchers use for designing hierarchical models is to fit an existing model, look for additional dependencies in the learned representation, and extend the model to capture those dependencies. We formalize this process in terms of a context-free grammar. In particular,

we present a notation for describing matrix decomposition models as algebraic expressions, and organize the space of models into a context-free grammar which generates such expressions. The starting symbol corresponds to a structureless model where the entries of the input matrix are modeled as *i.i.d.* Gaussians. Each production rule corresponds to a simple unsupervised learning model, such as clustering or dimensionality reduction. These production rules lie at the heart of our approach: we fit and evaluate a wide variety of models using a small toolbox of algorithms corresponding to the production rules, and the production rules guide our search through the space of structures.

The main contributions of this paper are threefold. First, we present a unifying framework for matrix decompositions based on a context-free grammar which generates a wide variety of structures through the compositional application of a few simple production rules. Second, we exploit our grammar to infer the latent components and estimate predictive likelihood in all of these structures generically and efficiently using a small toolbox of reusable algorithms corresponding to different component matrix priors and production rules. Finally, by performing greedy search over our grammar using predictive likelihood as the criterion, we can (in practice) typically choose the correct structure from the data while evaluating only a small fraction of all possible structures.

Section 3 defines our matrix decomposition formalism, and Sections 4 and 5 present our generic algorithms for inferring component matrices and evaluating individual structures, respectively. Section 6 describes how we search our space of structures. Finally, in Section 7, we evaluate the structure learning procedure on synthetic data and on real-world datasets as diverse as image patches, motion capture, 20 Questions, and Senate voting records, all using *exactly the same code*. Our procedure learns correct and/or plausible model structures for a wide variety of synthetic and real datasets, and gracefully falls back to simpler structures in high-noise conditions.

2 Related work

There is a long history of attempts to infer model structures automatically. The field of algorithmic information theory (Li and Vitanyi, 1997) studies how to represent data in terms of a short program/input pair which could have generated it. One prominent example, Solomonoff induction, can learn any computable generative model, but is itself uncomputable. Minimum message length (Wallace, 2005), minimum description length (Barron et al., 1998), and Bayesian model comparison (MacKay, 1992) are frameworks which can, in principle, be used to compare very different generative models. In practice, they have primarily been used for controlling complexity within a given model class. By contrast, our aim is to choose from a very large space of model classes by exploiting shared structure between the models.

Other work has focused on searching within more restricted spaces of models, such as undirected (Lee et al., 2006) and directed (Teyssier and Koller, 2005) graphical models, and graph embeddings (Kemp and Tenenbaum, 2008). Kemp and Tenenbaum (2008) model human "domain structure" learning as selecting between a fixed set of graph structures. Similarly to this paper, their structures are generated from a few simple rules; however, whereas their set of structures is small enough to exhaustively evaluate each one, we search over a much larger set of structures in a way that explicitly exploits the recursive nature of the space. Furthermore, our space of matrix decomposition structures is especially broad, including many bread-and-butter models from unsupervised learning, as well as the building blocks of many hierarchical Bayesian models.

We note that several other researchers have proposed unifying frameworks for unsupervised learning which overlap substantially with our own. Roweis and Ghahramani (1999)'s "generative model for generative models" presents a lattice showing relationships between different models. Srebro (2004) and Singh and Gordon (2008) each interpreted a variety of unsupervised learning algorithms as factorizing an input matrix into a product of two factors. Exponential family PCA (Collins et al., 2002; Mohamed et al., 2008) generalizes low-rank factorizations to other observation models in the exponential family. Our work differs from these in that our matrix decomposition formalism is specifically designed to support efficient generic inference and structure learning. We defer discussion of particular matrix decomposition models to Section 3.1, after we have introduced our formalism.

Our work has several parallels in the field of equation discovery. Langley et al. (1984) built a knowledge discovery system called BACON which reproduced classical scientific discoveries. BACON followed a greedy search procedure similar to our own: it repeatedly fit statistical models and looked for additional structure in the learned parameters. Our work is similar in spirit, but uses matrices rather than scalars as the building blocks, allowing us to capture rich structure in high-dimensional spaces. Todorovski and Dzeroski (1997) used a context-free grammar to define spaces of candidate equations. Our approach differs in that we explicitly use the grammar to structure posterior inference and search over model structures.

3 A grammar for matrix decompositions

We first present a notation for describing matrix decomposition models as algebraic expressions, such as $MG + G$. Each letter corresponds to a particular distribution over matrices. When the dimensions of all the component matrices are specified, the algebraic expression defines a generative

model: first sample all of the component matrices independently from their corresponding priors, and then evaluate the expression. The component priors are as follows:

1. **Gaussian (G).** Entries are independent Gaussians:

$$u_{ij} \sim \text{Gaussian}(0, \lambda_i^{-1}\lambda_j^{-1}).$$

   This is our most generic component prior, and gives a way of deferring or ignoring structure.[1]

2. **Multinomial (M).** Rows are independent multinomials, with one 1 and the rest 0's:

$$\pi \sim \text{Dirichlet}(\alpha) \qquad u_i \sim \text{Multinomial}(\pi).$$

   This is useful for clustering models, where u_i determines the cluster assignment for the i^{th} row.

3. **Bernoulli (B).** Entries are independent Bernoullis:

$$\pi_j \sim \text{Beta}(a, b) \qquad u_{ij} \sim \text{Bernoulli}(\pi_j).$$

   This is useful for binary latent feature models.

4. **Integration matrix (C).** Entries below the diagonal are deterministically 1:

$$u_{ij} = \mathbf{1}_{i \geq j}.$$

   This is useful for modeling temporal structure, as multiplying by this matrix has the effect of cumulatively summing the rows. (Mnemonic: C for "cumulative.")

We allow expressions consisting of addition, matrix multiplication, matrix transpose, elementwise multiplication ($\circ$), and elementwise exponentiation ($\exp$). Some of the dimensions of the component matrices are determined by the size of the input matrix; the rest (the latent dimensions) are determined automatically using the techniques of Section 4.

We observe that this notation for matrix decompositions is recursive: each sub-expression (such as $GM^T + G$ in the above example) is itself a matrix decomposition model. Furthermore, the semantics is compositional: the value of each expression depends only on the values of its sub-expressions and the operations used to combine them. These observations motivate our decision to define a space of models using a context-free grammar, a formalism which is widely used for representing recursive and compositional structures such as languages.

[1]The precision parameters λ_i and λ_j are drawn from the distribution $\text{Gamma}(a, b)$. If i indexes a data dimension (i.e. rows correspond to rows of the input matrix), the λ_is are tied. This allows the variance parameters to generalize to additional rows. If i indexes a latent dimension, the λ_is are all independent draws. This allows the variances of latent dimensions to be estimated. The same holds for the λ_js.

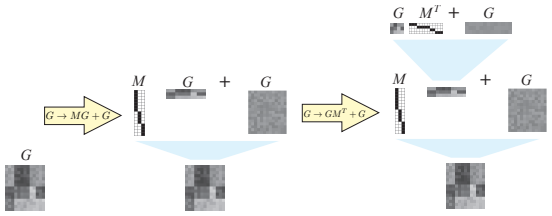


Figure 1: A synthetic example showing how an input matrix with block structure can be co-clustered by fitting the matrix decomposition structure $M(GM^T + G) + G$. Rows and columns are sorted for visualization purposes.

The starting symbol in our grammar is G, a structureless model where the entries are assumed to be independent Gaussians. Other models (expressions) are generated by repeatedly applying one of the following production rules:

low-rank approximation	$G \to GG + G$	(1)
clustering	$G \to MG + G \mid GM^T + G$	(2)
	$M \to MG + G$	(3)
linear dynamics	$G \to CG + G \mid GC^T + G$	(4)
sparsity	$G \to \exp(G) \circ G$	(5)
binary factors	$G \to BG + G \mid GB^T + G$	(6)
	$B \to BG + G$	(7)
	$M \to B$	(8)

For instance, any occurrence of G in a model may be replaced by $GG + G$ or $MG + G$. Repeated application of these production rules allows us to build hierarchical models by capturing additional dependencies between variables which were previously modeled as independent.

3.1 Examples

We now turn to several examples in which our simple components and production rules give rise to a rich variety of models from unsupervised learning. While the model space is symmetric with respect to rows and columns, for purposes of exposition, we will adopt the convention that the rows of the input matrix correspond to data points and columns corresponds to observed attributes.

We always begin with the model G, which assumes the entries of the matrix are *i.i.d.* Gaussian. Applying productions in our grammar allows us to capture additional structure. For instance, starting with Rule 2(a) gives the model $MG + G$, which clusters the rows (data points). In more detail, the M represents the cluster assignments, the first G represents the cluster centers, and the second G represents within-cluster variation. These three matrices are sampled independently, the assignment matrix is multiplied by the center matrix, and the within-cluster variation is added to the result. By applying Rule 2(b), the clustering model can be extended to co-clustering (Kemp et al., 2006), where the columns (attributes) form clusters as well. In our framework, this can be represented as $M(GM^T + G) + G$. We

need not stop here: for instance, there may be coherent covariation even within individual clusters. One can capture this variation by applying Rule 3 to get the Bayesian Clustered Tensor Factorization (BCTF) (Sutskever et al., 2009) model $(MG+G)(GM^T+G)+G$. This process is shown in cartoon form in Figure 1.

For an example from vision, consider a matrix X, where each row is a small (e.g. 12×12) patch sampled from an image and vectorized. Image patches can be viewed as lying near a low-dimensional subspace spanned by the lowest frequency Fourier coefficients (Bossomaier and Snyder, 1986). This can be captured by the low-rank model $GG+G$. In a landmark paper, Olshausen and Field (1996) found that image patches are better modeled as a linear combination of a small number of components drawn from a larger dictionary. In other words, X is approximated as the product WA, where each row of A is a basis function, and W is a sparse matrix giving the linear reconstruction coefficients for each patch. By fitting this "sparse coding" model, they obtained a dictionary of oriented edges similar to the receptive fields of neurons in the primary visual cortex. If we apply Rule (5), we obtain a Bayesian version of sparse coding, $(\exp(G) \circ G)G + G$, similar to the model proposed by Berkes et al. (2008). Intuitively, the latent Gaussian coefficients are multiplied elementwise by "scale" variables to give a heavy-tailed distribution. Many researchers have designed models to capture the dependencies between these scale variables, and such "Gaussian scale mixture" models represent the state-of-the art for low-level vision tasks such as denoising (Portilla et al., 2003) and texture synthesis (Portilla and Simoncelli, 2000). One such GSM model is that of Karklin and Lewicki (2008), who fit a low-rank model to the scale variables. By applying Rule (1) to the sparse coding structure, we can represent their model in our framework as $(\exp(GG+G)\circ G)G+G$. This model has been successful at capturing higher-level textural properties of a scene and has properties similar to complex cells in the primary visual cortex.

Figure 2 gives several additional examples of matrix decomposition models and highlights the relationships between them. We emphasize that our goal is not to reproduce existing models exactly, but to develop a formalism powerful enough to express a wide variety of statistical assumptions about the latent factors underlying the data.

We note that many of the above models are not typically viewed as matrix decomposition structures. Describing them as such results in a compact notation for defining them and makes clearer the relationships between the different models. The above examples have in common that complex models can be derived by incrementally adding structure to a sequence of simpler models (in a way that parallels the path researchers took to discover them). This observation motivates our proposed procedures for inference and structure learning.

4 Posterior inference of component matrices

Searching over matrix decomposition structures requires a generic and unified approach for posterior sampling of the latent matrices. Unfortunately, for most of the structures we consider, this posterior is complicated and multimodal, and escaping from local modes requires carefully chosen special-purpose sampling operators. Engineering such operators for thousands of different models would be undesirable.

Fortunately, the compositional nature of our model space allows us to focus the engineering effort on the relatively small number of production rules. In particular, observe that in a realization of the generative process, the value of an expression depends only on the values of its subexpressions. This suggests the following initialization procedure: when applying a production rule P to a matrix S, sample from the posterior for P's generative model conditioned on it evaluating (exactly) to S. Many of our production rules correspond to simple machine learning models for which researchers have already expended much time developing efficient inference algorithms:

1. **Low rank.** To apply the rule $G \to GG+G$, we fit the probabilistic matrix factorization (Salakhutdinov and Mnih, 2008) model using block Gibbs sampling over the two factors. While PMF assumes a fixed latent dimension, we choose the dimension automatically by placing a Poisson prior on the dimension and moving between states of differing dimension using reversible jump MCMC (Green, 1995).

2. **Clustering.** To apply the clustering rule to rows: $G \to MG + G$, or to columns: $G \to GM^T + G$, we perform collapsed Gibbs sampling over the cluster assignments in a Dirichlet process mixture model.

3. **Binary factors.** To apply the rule $G \to BG + G$ or $G \to GB^T + G$, we perform accelerated collapsed Gibbs sampling (Doshi-Velez and Ghahramani, 2009) over the binary variables in a linear-Gaussian Indian Buffet Process (Griffiths and Ghahramani, 2005) model, using split-merge proposals (Meeds et al., 2006) to escape local modes.

4. **Markov chains.** The rule $G \to CG + G$ is equivalent to estimating the state of a random walk given noisy observations, which is done using Rauch-Tung-Striebel (RTS) smoothing.

The remaining production rules begin with a random decomposition of S. While some of these algorithms involve fitting Bayesian nonparametric models, once the dimensionality is chosen, the model is converted to a finite model of fixed dimensionality (as defined in section 3). The

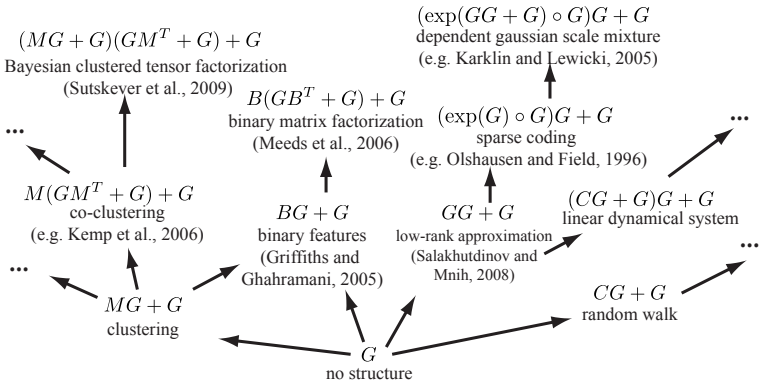


Figure 2: Examples of existing machine learning models which fall under our framework. Arrows represent models reachable using a single production rule. Only a small fraction of the 2496 models reachable within 3 steps are shown, and not all possible arrows are shown.

smart initialization step is followed by generic Gibbs sampling over the entire model. We note that our initialization procedure generalizes "tricks of the trade" whereby complex models are initialized from simpler ones (Kemp et al., 2006; Miller et al., 2009).

In addition to simplifying the engineering, this procedure allows us to reuse computations between different structures. Most of the computation time is in the initialization steps. Each of these steps only needs to be run once on the full matrix, specifically when the first production rule is applied. Subsequent initialization steps are performed on the component matrices, which are considerably smaller. This allows a large number of high level structures to be fit for a fraction of the cost of fitting them from scratch.

5 Scoring candidate structures

Performing model selection requires a criterion for scoring individual structures which is informative yet tractable. To motivate our method, we first discuss two popular choices: marginal likelihood of the input matrix and entrywise mean squared error (MSE). Marginal likelihood, the probability of the data with all the latent variables integrated out, is widely used in Bayesian model selection. Unfortunately, this requires integrating out all of the latent component matrices, whose posterior distribution is highly complex and multimodal. While elegant solutions exist for particular models, estimating the data marginal likelihood generically is still extremely difficult. At the other extreme, one can hold out a subset of the entries of the matrix and compute the mean squared error for predicting these entries. MSE is easier to implement, but we found that it was unable to distinguish many of the the more complex structures in our grammar.

As a compromise between these two approaches, we chose to evaluate predictive likelihood of held-out rows and columns. That is, for each row (or column) x of the matrix, we evaluate $p(x|X_O)$, where X_O denotes an "observed" sub-matrix. Like marginal likelihood, this tests the model's ability to predict entire rows or columns. However, it can be efficiently approximated in our class of models using a small but carefully chosen toolbox corresponding to the component matrix priors in our grammar. We discuss the case of held-out rows; columns are handled analogously.

First, by expanding out the products in the expression, we can write the decomposition uniquely in the form

$$X = U_1 V_1 + \cdots + U_n V_n + E, \tag{1}$$

where E is an *i.i.d.* Gaussian "noise" matrix and the U_i's are any of the following: (1) a component matrix G, M, or B, (2) some number of Cs followed by G, (3) a Gaussian scale mixture. The held-out row x can therefore be represented as:

$$x = V_1^T u_1 + \cdots + V_n^T u_n + e. \tag{2}$$

The predictive likelihood is given by:

$$p(x|X_O) = \int p(U_O, V|X_O) p(u|U_O) p(x|u,V)\, dU_O\, du\, dV \tag{3}$$

where U_O is shorthand for $(U_{O1}, \ldots, U_{On})$ and u is shorthand for $(u_1, \ldots, u_n)$.

In order to evaluate this integral, we generate samples from the posterior $p(U_O, V|X)$ using the techniques described in Section 4, and compute the sample average of

$$p_{pred}(x) \triangleq \int p(u|U_O) p(x|u,V)\, du \tag{4}$$

If the term U_i is a Markov chain, the predictive distribution $p(u_i|U_O)$ can be computed using Rauch-Tung-Striebel smoothing; in the other cases, u and U_O are related only

through the hyperparameters of the component prior. Either way, each term $p(u_i|U_O)$ can be summarized as a Gaussian, multinomial, Bernoulli, or Gaussian scale mixture distribution.

It remains to marginalize out the latent representation u of the held-out row. While this can be done exactly in some simple models, it is intractable in general (for instance, if u is Bernoulli or a Gaussian scale mixture). It is important that the approximation to the integral be a lower bound, because otherwise an overly optimistic model could be chosen even when it is completely inappropriate.

Our approach is a hybrid of variational and sampling techniques. We first lower bound the integral (4) in an approximate model $\tilde{p}$ where the Gaussian scale mixture components are approximated as Gaussians. This is done using using the variational Bayes bound

$$\log \tilde{p}_{pred}(x) \geq E_q[\log \tilde{p}_{pred}(x, u)] + \mathcal{H}(q).$$

The approximating distribution $q(u)$ is such that all of the discrete components are independent, while the Gaussian components are marginalized out. The ratio $p_{pred}(x)/\tilde{p}_{pred}(x)$ is then estimated using annealed importance sampling (AIS) (Neal, 2001). Because AIS is an unbiased estimator which always takes positive values, by Markov's inequality we can regard it as a stochastic lower bound. Therefore, this small toolbox of techniques allows us to (stochastically) lower bound the predictive likelihood across a wide variety of matrix decomposition models.

6 Search over structures

We aim to find a matrix decomposition structure which is a good match to a dataset, as measured by the predictive likelihood criterion of Section 5. Since the space of models is large and inference in many of the models is expensive, we wish to avoid exhaustively evaluating every model. Instead, we adopt a greedy search procedure inspired by the process of scientific discovery. In particular, consider a common heuristic researchers use to build probabilistic models: we begin with a model which has already been applied to a problem, look for additional dependencies not captured by the model, and refine the model to account for those dependencies.

In our approach, refining a model corresponds to applying one of the productions. This suggests the following greedy search procedure, which iteratively "expands" the best-scoring unexpanded models by applying all possible production rules and scoring the resulting models. In particular we first expand the structureless model G. Then, in each step, we expand the K best-performing models from the previous step by applying all possible productions. We then score all the resulting models. The procedure stops when no model achieves sufficient improvement over the best model from the previous step. We refer to the models reached in i productions as the Level i models; for instance, $GG+G$ is a Level 1 model and $(MG+G)G+G$ is a Level 2 model.

The effectiveness of this search procedure depends whether the score of a simple structure is a strong indicator of the best score which can be obtained from the structures derived from it. In our experiments, the scores of the simpler structures turned out to be a powerful heuristic: while our experiments used $K = 3$, in most all cases, the correct (or best-scoring) structure would have been found with a purely greedy search ($K = 1$). This results in enormous savings because of the compositional nature of our search space: while the number of possible structures (up to a given level) grows quickly in the number of production rules, the number of structures evaluated by this search procedure is merely linear.

The search procedure returns a high-scoring structure for each level in our grammar. There remains a question of when to stop. Choosing between structures of differing complexity imposes a tradeoff between goodness of fit and other factors such as interpretability and tractability of inference, and inevitably the choice is somewhat subjective. In practice, a user may wish to run our procedure up to a fixed level and analyze the sequence of models chosen, as well as the predictive likelihood improvement at each level. However, for the purposes of evaluating our system, we need it to return a single answer. In all of our experiments, we adopt the following arbitrary but consistent criterion: prefer the higher level structure if its predictive log-likelihood score improves on the previous level by at least one nat per row and column.[2]

7 Experiments

7.1 Synthetic data

We first validated our structure learning procedure on synthetic data where the correct model was known. We generated matrices of size 200×200 from all of the models in Figure 2, with 10 latent dimensions. The noise variance σ^2 was varied from 0.1 to 10, while the signal variance was fixed at 1.[3] The structures selected by our procedure are shown in Table 1.

[2] More precisely, if $\frac{S_i - S_{i-1}}{N+D} > 1$, where S_i is the total predictive log-likelihood for the level i model summed over all rows and columns, and N and D are the numbers of rows and columns, respectively. We chose to normalize by $N+D$ because the predictive likelihood improvements between more abstract models tend to grow with the number of rows and columns in the input matrix, rather than the number of entries.

[3] Our grammar generates expressions of the form $\cdots + G$. We consider this final G term to be the "noise" and the rest to be the "signal," even though the models and algorithms do not distinguish the two.

	— Increasing noise ⟶			
	$\sigma^2 = 0.1$	$\sigma^2 = 1$	$\sigma^2 = 3$	$\sigma^2 = 10$
low-rank	$GG + G$	$GG + G$	$GG + G$	①G
clustering	$MG + G$	$MG + G$	$MG + G$	$MG + G$
binary latent features	①$(BG + G)G + G$	$BG + G$	$BG + G$	$BG + G$
co-clustering	$M(GM^T + G) + G$	$M(GM^T + G) + G$	$M(GM^T + G) + G$	①$GM^T + G$
binary matrix factorization	①$(BG + G)(GB^T + G) + G$	$(BG + G)B^T + G$	②$GG + G$	②$GG + G$
BCTF	$(MG + G)(GM^T + G) + G$	$(MG + G)(GM^T + G) + G$	②$GM^T + G$	③G
sparse coding	$(\exp(G) \circ G)G + G$	$(\exp(G) \circ G)G + G$	$(\exp(G) \circ G)G + G$	②G
dependent GSM	①$(\exp(G) \circ G)G + G$	①$(\exp(G) \circ G)G + G$	①$(\exp(G) \circ G)G + G$	③$BG + G$
random walk	$CG + G$	$CG + G$	$CG + G$	①G
linear dynamical system	$(CG + G)G + G$	$(CG + G)G + G$	$(CG + G)G + G$	②$BG + G$

Table 1: The structures learned from 200×200 matrices generated from various distributions, with signal variance 1 and noise variance σ^2. Incorrect structures are marked with a 1, 2, or 3, depending how many decisions would need to be changed to find the correct structure. We observe that our approach typically finds the correct answer in low noise settings and backs off to simpler models in high noise settings.

We observe that seven of the ten structures were identified perfectly in both trials where the noise variance was no larger than the data variance ($\sigma^2 \leq 1$). When $\sigma^2 = 0.1$, the system incorrectly chose $(BG+G)G+G$ for the binary latent feature data, rather than $BG + G$. Similarly, it chose $(BG + G)(GB^T + G) + G$ rather than $(BG + G)B^T + G$ for binary matrix factorization. In both cases, the sampler learned an incorrect set of binary features, and the additional flexibility of the more complex model compensated for this. This phenomenon, where more structured models compensate for algorithmic failures in simpler models, has also been noted in the context of deep learning (Salakhutdinov and Murray, 2008).

Our system also did not succeed in learning the dependent Gaussian scale mixture structure $(\exp(GG+G)\circ G)G+G$ from synthetic data, instead generally falling back to the simpler sparse coding model $(\exp(G) \circ G)G + G$. For $\sigma^2 = 0.1$ the correct structure was in fact the highest scoring structure, but did not cross our threshold of 1 nat improvement over the previous level. We note that in every case, there were nearly 2500 incorrect structures to choose from, so it is notable that the correct model structure can be recovered most of the time.

In general, when the noise variance was much larger than the signal variance, the system gracefully fell back to simpler models, such as $GM^T + G$ instead of the BCTF model $(MG + G)(GM^T + G) + G$ (see Section 3.1). At the extreme, in the maximum noise condition, it chose the structureless model G much of the time. Overall, our procedure reliably learned most of the model structures in low-noise settings (impressive considering the extremely large space of possible wrong answers) and gracefully fell back to simpler models when necessary.

7.2 Real-world data

Next, we evaluated our system on several real-world datasets. We first consider two domains, motion capture and image statistics, where the core statistical assumptions are widely agreed upon, and verify that our learned structures are consistent with these assumptions. We then turn to domains where the correct structure is more ambiguous and analyze the representations our system learns.

In general, we do not expect every real-world dataset to have a unique best structure. In cases where the predictive likelihood score differences between multiple top-scoring models were not statistically significant, we report the set of top-scoring models and analyze what they have in common.

Motion capture. We first consider a human motion capture dataset (Hsu et al., 2005; Taylor et al., 2007) consisting of a person walking in a variety of styles. Each row of the matrix gives the person's orientation and displacement in one frame, as well as various joint angles. We used 200 frames (6.7 seconds), and 45 state variables. In the first step, the system chose the Markov chain model $CG + G$, which assumes that the components of the state evolve continuously but independently over time. Since a person's different joint angles are clearly correlated, the system next captured these correlations with the model $C(GG+G)+G$. This is slightly different from the popular linear dynamical system model $(CG + G)G + G$, but it is more physically correct in the sense that the LDS assumes the deviations of the observations from the low-dimensional subspace must be independent in different time steps, while our learned structure captures the temporal continuity in the deviations.

Natural image patches. We tested the system on the Sparsenet dataset of Olshausen and Field (1996), which consists of 10 images of natural scenes which were blurred and whitened. The rows of the input matrix corresponded to 1,000 patches of size 12×12. In the first stage, the model learned the low-rank representation $GG + G$, and in the second stage, it sparsified the linear reconstruction coefficients to give the sparse coding model $(\exp(G) \circ G)G + G$. In the third round, it modeled the dependencies between the scale variables by recursively giving them a low-rank representation, giving a dependent Gaussian scale mixture (GSM) model $(\exp(GG + G) \circ G)G + G$ reminiscent of Karklin and Lewicki (2008). A closely related model, $(\exp(GB^T + G) \circ G)G + G$, also achieved a score not significantly lower. Both of these structures resulted in a

	Level 1	Level 2	Level 3
Motion capture	$CG + G$	$C(GG + G) + G$	—
Image patches	$GG + G$	$(\exp(G) \circ G)G + G$	$(\exp(GG + G) \circ G)G + G$
20 Questions	$MG + G$	$M(GG + G) + G$	—
Senate votes	$GM^T + G$	$(MG + G)M^T + G$	—

Table 2: The best performing models at each level of our grammar for real-world datasets. These correspond to plausible structures for the datasets, as discussed in the text.

rank-one factorization of the scale matrix, similar to the radial Gaussianization model of Lyu and Simoncelli (2009) for neighboring wavelet coefficients.

Dependent GSM models (see Section 3.1) are the state-of-the-art for a variety of image processing tasks, so it is interesting that this structure can be learned merely from the raw data. We note that a single pass through the grammar reproduces an analogous sequence of models to those discovered by the image statistics research community as discussed in Section 3.1.

20 Questions. We now consider a dataset collected by Pomerleau et al. (2009) of Mechanical Turk users' responses to 218 questions from the 20 Questions game about 1000 concrete nouns (e.g. animals, foods, tools). The system began by clustering the entities using the flat clustering model $MG + G$. In the second stage, it found low-rank structure in the matrix of cluster centers, resulting in the model $M(GG+G)+G$. No third-level structure achieved more than 1 nat improvement beyond this. The low-rank representation had 8 dimensions, where the largest variance dimension corresponded to living vs. nonliving and the second largest corresponded to large vs. small. The 39 clusters, the 20 largest of which are shown in Figure 3, correspond to semantically meaningful categories.

We note that two other models expressing similar assumptions, $M(GB^T + G) + G$ and $(MG + G)G + G$, achieved scores only slightly lower. What these models have in common is a clustering of entities (but not questions) coupled with low-rank structure between entities and questions. The learned clusters and dimensions are qualitatively similar in each case.

Senate voting records. Finally, we consider a dataset of roll call votes from the 111th United States Senate (2009-2010). Rows correspond to Senators, and the columns correspond to all 696 votes, most of which were on procedural motions and amendments to bills. Yea votes were mapped to 1, Nay and Present were mapped to -1, and absences were treated as unobserved. In the first two stages, our procedure clustered the votes and Senators, giving the clustering model $GM^T + G$ and the co-clustering model $(MG+G)M^T+G$, respectively. Senators clustered along party lines, as did most of the votes, according to the party of the proposer. The learned representations are all visualized in Figure 4.

In the third stage, one of the best performing models was Bayesian clustered tensor factorization (BCTF) (see section 3.1), where Senators and votes are each clustered inside a low-rank representation.[4] This low-rank representation was rank 5, with one dominant dimension corresponding to the liberal-conservative axis. The BCTF model makes it clearer that the clusters of Senators and votes are not independent, but can be seen as occupying different points in a low-dimensional representation. This model improved on the previous level by less than our 1 nat cutoff.[5] The models in this sequence increasingly highlight the polarization of the Senate.

8 Discussion

We have presented an effective and practical method for automatically determining the model structure in a particular space of models, matrix decompositions, by exploiting compositionality. However, we believe our approach can be extended beyond the particular space of models presented here. Most straightforwardly, additional components can be added to capture other motifs of probabilistic modeling, such as tree embeddings and low-dimensional embeddings. More generally, it should be fruitful to investigate other model classes with compositional structure, such as tensor decompositions.

In either case, exploiting the structure of the model space becomes increasingly essential. For instance, the number of models reachable in 3 steps is cubic in the number of production rules, whereas the complexity of the greedy search is linear. For tensors, the situation is even more overwhelming: even if we restrict our attention to analogues of $GG+G$, a wide variety of provably distinct generalizations have been identified, including the widely used Tucker3 and PARAFAC decompositions (Kolda and Bader, 2007).

[4] The other models whose scores were not significantly different were: $(MG+G)M^T+BG+G$, $(MG+G)M^T+GM^T+G$, $G(GM^T + G) + GM^T + G$, and $(BG + G)(GM^T + G) + G$. All of these models include the clustering structure but account for additional variability within clusters.

[5] BCTF results in a more compact representation than the co-clustering model, but our predictive likelihood criterion doesn't reward this except insofar as overfitting hurts a model's ability to generalize to new rows and columns. We speculate that a fully Bayesian approach using marginal likelihood may lead to more compact structures.

1. Miscellaneous. key, chain, powder, aspirin, umbrella, quarter, cord, sunglasses, toothbrush, brush
2. Clothing. coat, dress, pants, shirt, skirt, backpack, tshirt, quilt, carpet, pillow, clothing, slipper, uniform
3. Artificial foods. pizza, soup, meat, breakfast, stew, lunch, gum, bread, fries, coffee, meatballs, yoke
4. Machines. bell, telephone, watch, typewriter, lock, channel, tuba, phone, fan, ipod, flute, aquarium
5. Natural foods. carrot, celery, corn, lettuce, artichoke, pickle, walnut, mushroom, beet, acorn
6. Buildings. apartment, barn, church, house, chapel, store, library, camp, school, skyscraper
7. Printed things. card, notebook, ticket, note, napkin, money, journal, menu, letter, mail, bible
8. Body parts. arm, eye, foot, hand, leg, chin, shoulder, lip, teeth, toe, eyebrow, feet, hair, thigh
9. Containers. bottle, cup, glass, spoon, pipe, gallon, pan, straw, bin, clipboard, carton, fork
10. Outdoor places. trail, island, earth, yard, town, harbour, river, planet, pond, lawn, ocean
11. Tools. knife, chisel, hammer, pliers, saw, screwdriver, screw, dagger, spear, hoe, needle
12. Stuff. speck, gravel, soil, tear, bubble, slush, rust, fat, garbage, crumb, eyelash
13. Furniture. bed, chair, desk, dresser, table, sofa, seat, ladder, mattress, handrail, bench, locker
14. Liquids. wax, honey, pint, disinfectant, gas, drink, milk, water, cola, paste, lemonade, lotion
15. Structural features. bumper, cast, fence, billboard, guardrail, axle, deck, dumpster, windshield
16. Non-solid things. surf, fire, lightning, sky, steam, cloud, dance, wind, breeze, tornado, sunshine
17. Transportation. airplane, car, train, truck, jet, sedan, submarine, jeep, boat, tractor, rocket
18. Herbivores. cow, horse, lamb, camel, pig, hog, calf, elephant, cattle, giraffe, yak, goat
19. Internal organs. rib, lung, vein, stomach, heart, brain, smile, blood, lap, nerve, lips, wink
20. Carnivores. bear, walrus, shark, crocodile, dolphin, hippo, gorilla, hyena, rhinocerous

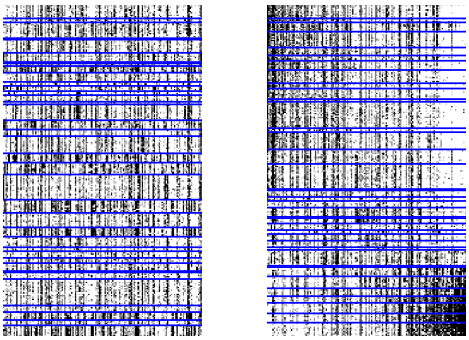
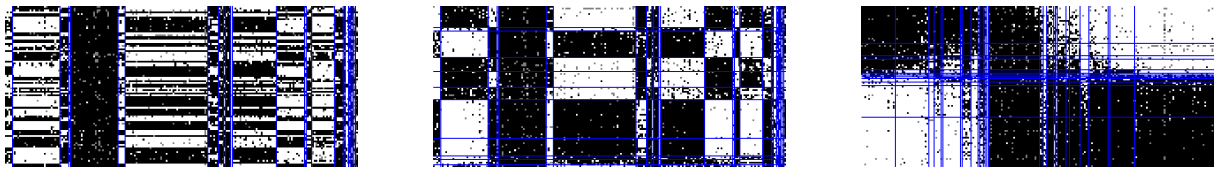


Figure 3: **(left)** The 20 largest clusters discovered by our Level 2 model $M(GG+G)+G$ for the 20 Questions dataset. Each line gives **our interpretation**, followed by random items from the cluster. **(right)** Visualizations of the Level 1 representation $MG+G$ and the Level 2 representation $M(GG+G)+G$. Rows = entities, columns = questions. 250 rows and 150 columns were selected at random from the original matrix. Rows and columns are sorted first by cluster, then by the highest variance dimension of the low-rank representation (if applicable). Clusters were sorted by the same dimension as well. Blue = cluster boundaries.



(a) Level 1: GM^T+G (b) Level 2: $(MG+G)M^T+G$ (c) Level 3: $(MG+G)(GM^T+G)+G$

Figure 4: Visualization of the representations learned from the Senate voting data. Rows = Senators, columns = votes. 200 columns were selected at random from the original matrix. Black = yes, white = no, gray = absence. Blue = cluster boundaries. Rows and columns are sorted first by cluster (if applicable), then by the highest variance dimension of the low-rank representation (if applicable). Clusters are sorted by the same dimension as well. The models in the sequence increasingly reflect the polarization of the Senate.

What is the significance of the grammar being context-free? While it imposes no restriction on the models themselves, it has the effect that the grammar "overgenerates" model structures. Our grammar licenses some nonsensical models: for instance, $G(MG+G)+G$, which attempts to cluster dimensions of a latent space which is defined only up to affine transformation. Reassuringly, we have never observed such models being selected by our search procedure — a useful sanity check on the output of the algorithm. The only drawback is that the system wastes some time evaluating meaningless models. Just as context-free grammars for English can be augmented with attributes to enforce contextual restrictions such as agreement, our grammar could be similarly extended to rule out unidentifiable models. Such extensions may become important if our approach is applied to a much larger space of models.

Our context-free grammar formalism unifies a wide variety of matrix decomposition models in terms of compositional application of a few production rules. We exploited this compositional structure to efficiently and generically sample from and evaluate a wide variety of latent variable models, both continuous and discrete, flat and hierarchical. Greedy search over our grammar allows us to select a model structure from raw data by evaluating only a small fraction of all models. This search procedure was effective at recovering the correct structure for synthetic data and sensible structures for real-world data. More generally, we believe this paper is a proof-of-concept for the practicality of selecting complex model structures in a compositional manner. Since many model spaces other than matrix factorizations are compositional in nature, we hope to spur additional research on automatically searching large, compositional spaces of models.

Acknowledgments

This work was partly funded by the ARO grant W911NF-08-1-0242 and by an NDSEG fellowship to RBG.

References

A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *Transactions on Information Theory*, 1998.

P. Berkes, R. Turner, and M. Sahani. On sparsity and overcompleteness in image models. In *Advances in Neural Information Processing Systems*, 2008.

T. Bossomaier and A. W. Snyder. Why spatial frequency processing in the visual cortex? *Vision Research*, 26(8):1307–1309, 1986.

M. Collins, S. Dasgupta, and R. Schapire. A generalization of principal component analysis to the exponential family. In *Neural Information Processing Systems*, 2002.

Finale Doshi-Velez and Zoubin Ghahramani. Accelerated sampling for the Indian buffet process. In *Int'l. Conf. on Machine Learning*, 2009.

P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 1995.

T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. Technical report, Gatsby Computational Neuroscience Unit, 2005.

E. Hsu, K. Pulli, and J. Popovic. Style translations for human motion. In *ACM Transactions on Graphics*, 2005.

Y. Karklin and M. S. Lewicki. Emergence of complex cell properties by learning to generalize in natural scenes. *Nature*, 457: 83–86, January 2008.

Charles Kemp and Joshua B. Tenenbaum. The discovery of structural form. *PNAS*, 2008.

Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, pages 381–388, 2006.

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 2007.

Pat Langley, Herbert A. Simon, and Gary L. Bradshaw. Heuristics for empirical discovery. In *Knowledge Based Learning Systems*. Springer-Verlag, London, UK, 1984.

S. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In *NIPS*, 2006.

M. Li and P. Vitanyi. *An introduction to Kolmogorov complexity and its applications*. Springer, 1997.

S. Lyu and E. P. Simoncelli. Nonlinear extraction of independent components of natural images using radial Gaussianization. *Neural Computation*, 21(6):1485–1519, 2009.

DJC MacKay. Bayesian interpolation. *Neural Computation*, 1992.

E. Meeds, Z. Ghahramani, R. Neal, and S. T. Roweis. Modelling dyadic data with binary latent factors. In *NIPS*, volume 20, pages 1002–1009, 2006.

K. T. Miller, T. L. Griffiths, and M. I. Jordan. Nonparametric latent feature models for link prediction. In *Advances in Neural Information Processing Systems*, 2009.

S. Mohamed, K. Heller, and Z. Ghahramani. Bayesian exponential family PCA. In *NIPS*, 2008.

R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, April 2001.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–9, June 1996.

D. Pomerleau, G. E. Hinton, M. Palatucci, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009.

J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–71, 2000.

J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Signal Processing*, 12(11): 1338–1351, 2003.

Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. In *Neural Computation*, volume 11, pages 305–345, 1999.

Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, 2008.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Int'l. Conf. on Machine Learning*, 2008.

Ajit P. Singh and Geoffrey J. Gordon. A unified view of matrix factorizations. In *European Conference on Machine Learning*, 2008.

N. Srebro. *Learning with matrix factorizations*. PhD thesis, MIT, 2004.

I. Sutskever, R. Salakhutdinov, and J. B. Tenenbaum. Modelling relational data using Bayesian clustered tensor factorization. In *NIPS*, pages 1821–1828. 2009.

Graham W. Taylor, Geoffrey E. Hinton, and Sam Roweis. Modeling human motion using binary latent variables. In *NIPS*, 2007.

M. Teyssier and D. Koller. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *UAI*, 2005.

Ljupco Todorovski and Saso Dzeroski. Declarative bias in equation discovery. In *Int'l. Conf. on Machine Learning*, 1997.

C.S. Wallace. *Statistical and inductive inference by minimum message length*. Springer, 2005.

A Slice Sampler for Restricted Hierarchical Beta Process with Applications to Shared Subspace Learning

Sunil Kumar Gupta **Dinh Phung** **Svetha Venkatesh**
Center for Pattern Recognition and Data Analytics (PRaDA)
Deakin University, Australia
{sunil.gupta, dinh.phung, svetha.venkatesh}@deakin.edu.au

Abstract

Hierarchical beta process has found interesting applications in recent years. In this paper we present a modified hierarchical beta process prior with applications to hierarchical modeling of multiple data sources. The novel use of the prior over a hierarchical factor model allows factors to be shared across different sources. We derive a slice sampler for this model, enabling tractable inference even when the likelihood and the prior over parameters are non-conjugate. This allows the application of the model in much wider contexts without restrictions. We present two different data generative models – a linear Gaussian-Gaussian model for real valued data and a linear Poisson-gamma model for count data. Encouraging transfer learning results are shown for two real world applications – text modeling and content based image retrieval.

1 Introduction

Hierarchical modeling is becoming increasingly popular in Bayesian statistics – it allows joint modeling of multiple data sources, permitting shared patterns. A successful example of hierarchy in text modeling is the hierarchical Dirichlet process (HDP) [18]. HDP is used as a prior over a mixture model allowing shared mixture components (topics) across multiple data sources. Each data point from a data source is modeled using 1-out-of-K topics, shared across different sources. In many applications, where it may not be easy to separate out data into K mutually exclusive groups, an appropriate approach is factor analysis, where each data point is modeled using P-out-of-K factors. The benefits of joint modeling can still be retained by sharing these factors across different sources.

Previous works on nonparametric factor analysis have used beta processes. One of the earliest work develops the Indian buffet process (IBP) and uses it as a nonparametric prior over infinite binary matrices [6]. Similar works differing in inference methods are proposed in [15, 14, 5]. A significant contribution by Thibaux and Jordan [19] uncovers that the IBP is a predictive process for an underlying beta process with concentration parameter equaling one. In addition, they propose a hierarchical beta process (HBP) model and demonstrate its use for document classification in a *single* data source. The use of HBP can be extended to modeling multiple data sources by utilizing it as a nonparametric prior for *factor* models, in a similar way to HDP being used as a nonparametric prior for *mixture* models. However, when combining HBP with the factor model, inference is often intractable. This is mainly due to the need to integrate out the parameters when invoking new factors according to the HBP prior. For efficient computation of this integration, conjugacy is required between the parameter priors and the likelihood. However, the requirement of the conjugacy restricts the applicability of this model. One approach to deal with the non-conjugacy issues is through explicit representation of all the samples drawn from beta process. This requires representation of an infinite set of atoms - a practically impossible task. To circumvent this problem, one possible solution is to truncate the beta process. However, a pre-specified truncation level is arbitrary. To avoid this arbitrary truncation for a IBP based factor model, Teh et al [17] use a slice sampling technique, which turns the infinite representation problem into a finite representation problem. A similar approach for Dirichlet processes was taken by Walker [20]. However, a slice sampler for HBP based factor model is *yet* to be developed.

An attempt on nonparametric, hierarchical factor analysis using HBP was made in [8]. However, inference for this model has many *approximation* steps (refer section 3.1.2 in [8]), which risk sampling from incorrect posterior distributions. Thus *more accurate inference* methods *need to be developed.*

In this paper, we take the HBP prior and replace the upper beta process layer by an IBP prior, i.e. the concentration parameter of the underlying beta process is fixed to one. We refer to this modified hierarchical prior as the

Restricted Hierarchical Beta Process (R-HBP). This modification allows us to use the stick breaking construction of IBP [17] and explicitly represent the samples of the underlying beta process. To handle the resultant infinite representation problem, *we present a novel slice sampler* for the proposed R-HBP, deriving an infinite limit, necessary for tractable posterior over stick-weights of IBP. At the data source level, we still retain beta processes with arbitrary concentration parameters, allowing arbitrary sharing between data sources.

For the hierarchical factor analysis (i.e. factor analysis using R-HBP prior), we propose two different data models – a linear Gaussian data model for real-valued data and a Poisson-gamma model for count data. For the linear Poisson-gamma model, we present novel inference using auxiliary variables. Through synthetic experiments, we illustrate the behavior of our model, demonstrating that it correctly discovers all parameters including the dimensionalities of the factor subspaces. We further demonstrate the proposed models for two real world tasks – text modeling and content based image retrieval. For text modeling, which is demonstrated on NIPS 0-12 dataset (constructed from 12 years of NIPS proceedings), we successfully show the benefits of transfer learning. In addition, we show that linear Poisson-gamma model achieves much better perplexity than other models such as linear Gaussian model or HDP mixture model. For content based image retrieval, using NUS-WIDE animal dataset, we demonstrate that our method outperforms recent state-of-the-art methods.

Our main contributions are:

- A slice sampler for the proposed R-HBP model – we derive an infinite limit, which is necessary for tractable posterior over stick-weights of IBP.
- A novel, auxiliary variable, Gibbs sampler for the linear Poisson-gamma model.
- Demonstration of the proposed models for text modeling and content based image retrieval.

The rest of this paper is organized as follows: Section 2 presents brief background on the related nonparametric priors. Section 3 describes the modeling distributions for nonparametric hierarchical factor analysis. Section 4 presents the inference covering the novel slice sampling along with Gibbs sampling. Section 5 demonstrates the experiments using synthetic and real-world text and image data. Section 6 concludes this paper.

2 Preliminaries

2.1 Hierarchical Modeling using Beta Process

Let Ω be the set of all outcomes and $\mathscr{F}$ be its sigma algebra. We denote a beta process [10] as $B \sim \mathrm{BP}(\gamma_0, B_0)$ where γ_0 is a positive concentration parameter and B_0 is a base measure on Ω. The beta process is a completely random measure [13], implying that if $C_1, \ldots, C_n$ are the disjoint sets in Ω, the measures $B(C_1), \ldots, B(C_n)$ are independent random variables and the draws from a beta process are discrete with probability one. Using this property, a draw B from the beta process $\mathrm{BP}(\gamma_0, B_0)$ can be written as $B = \sum_k \beta_k \delta_{\phi_k}$ where $\phi_k \in \Omega$ is an atom drawn from B_0 and β_k is a random weight assigned to ϕ_k according to the following

$$\beta_k \sim \mathrm{beta}\left(\gamma_0 B_0(\phi_k), \gamma_0\left(1 - B_0(\phi_k)\right)\right), k = 1, 2, \ldots \quad (1)$$

If $B_0 = \sum_k \xi_k \delta_{\phi_k}$ (i.e. discrete), $B_0(\phi_k) = \xi_k$. For a continuous B_0, $\{(\phi_k, \beta_k)\}$ are i.i.d. draws from a Poisson process on the product space $\Omega \times [0,1]$ with a Lévy measure [12].

Building a hierarchy over beta processes, Thibaux and Jordan [19] introduced a hierarchical beta process (HBP) prior that allows the sharing of the atoms (drawn from a beta process) across multiple data sources. A hierarchical beta process is constructed in the following way : $B \sim \mathrm{BP}(\gamma_0, B_0)$ is a draw from a beta process with base measure B_0 and there emanate J child beta processes using B as their base measure (drawn as $A_j \sim \mathrm{BP}(\alpha_j, B)$, $j = 1, \ldots, J$). Finally, each A_j is used to parametrize a Bernoulli process denoted as $\mathbf{Z}_j^{i,:} \mid A_j \sim \mathrm{BeP}(A_j)$ where $\mathbf{Z}_j^{i,:}$ denotes i-th row of a binary matrix $\mathbf{Z}_j$. The support of random measure B is the same as that of base measure B_0 and is passed on to each A_j (for each j) allowing sharing of atoms across J-sources.

2.2 Indian Buffet Process and Stick-Breaking Construction

Indian buffet process (IBP) [6] is a nonparametric predictive prior developed as a generative model for infinite binary matrices. A key property of IBP is that it is exchangeable, i.e. if $\mathbf{Z} = [z_1, \ldots, z_N]$ (where z_i is a binary-valued infinite vector) then $p(z_1, \ldots, z_N) = p\left(z_{\sigma(1)}, \ldots, z_{\sigma(N)}\right)$ where σ is a permutation over $\{1, \cdots, N\}$. Given this exchangeability property, it is interesting to find out the underlying de Finetti stochastic process [4]. Investigating this question, Thibaux and Jordan [19] discovered that this underlying stochastic process is a beta process with concentration parameter equal to one, (i.e. $\gamma_0 = 1$ in Eq (1)). In most of the models, to construct the binary matrix $\mathbf{Z}$ from IBP, a sequential process [6] can be followed. However, this keeps the underlying beta process hidden. There are many applications where it is required to represent the underlying beta process explicitly. In such applications, one needs to know how to sample from the beta process underlying IBP. A solution to this problem was proposed by Teh et al [17] who derived stick-breaking construction for IBP and related this scheme to a similar stick breaking construction for Dirichlet processes.

Let us denote the beta process underlying IBP by $B \sim \mathrm{BP}(1, B_0)$ and assume that $\tau_0 = B_0(\Omega)$. Using an infi-

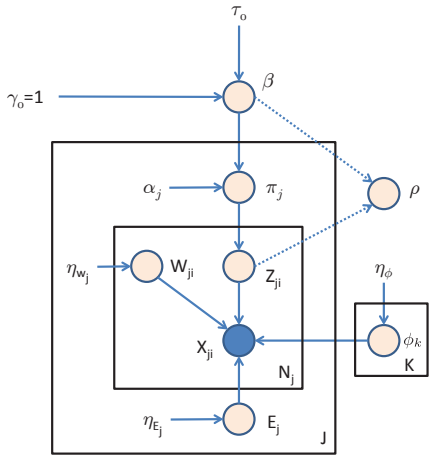


Figure 1: Directed graphical representation for the proposed nonparametric hierarchical factor model. The hyperparameters over $\mathbf{W}_j$, ϕ_k and $\mathbf{E}_j$ are denoted by η_{w_j},η_ϕ and η_{E_j} respectively.

nite set of atoms, we can write $B = \sum_l \beta_l \delta_{\phi_l}$. Now let $\{\beta_{(1)}, \beta_{(2)}, \ldots, \beta_{(K)}\}$ be the decreasing ordered representation of $\{\beta_1, \beta_2, \ldots, \beta_K\}$, where $\beta_l \sim \text{beta}\left(\frac{\tau_0}{K}, 1\right)$. Teh et al [17] showed that in the limit $K \to \infty$, the $\beta_{(k)}$ variables can be obtained from the following procedure

$$\nu_{(t)} \sim \text{beta}\,(\tau_0, 1)\,,\ \beta_{(k)} = \nu_{(k)}\beta_{(k-1)} = \Pi_{t=1}^{k}\nu_{(t)} \tag{2}$$

We shall denote the stick-breaking construction of Eq (2) as $\beta_{(k)} \sim \text{StickIBP}(\tau_0)$. A key property of this construction is that there exists a Markov relation [17] among $\{\beta_{(1)}, \beta_{(2)}, \ldots, \beta_{(K)}\}$, i.e. $\beta_{(k)}$ is conditionally independent of $\beta_{(1:k-2)}$ given $\beta_{(k-1)}$. Formally,

$$p\left(\beta_{(k)} \mid \beta_{(1:k-1)}\right) = \tau_0 \beta_{(k-1)}^{-\tau_0} \beta_{(k)}^{\tau_0 - 1} 1\left(0 \le \beta_{(k)} \le \beta_{(k-1)}\right) \tag{3}$$

where $1(Q)$ equals to one if Q is true and zero otherwise. Given top k variables $\{\beta_{(1)}, \beta_{(2)}, \ldots, \beta_{(k)}\}$, the other variables β_l have the following density function [17]

$$p\left(\beta_l \mid \beta_{(1:k)}\right) = \frac{\tau_0}{K} \beta_{(k)}^{-\left(\frac{\tau_0}{K}\right)} \beta_l^{\left(\frac{\tau_0}{K}\right)-1} 1\left(0 \le \beta_l \le \beta_{(k)}\right) \tag{4}$$

2.3 Restricted Hierarchical Beta Process

Restricted hierarchical beta process (R-HBP) is defined as a hierarchical beta process where the parent beta process has a concentration parameter equal to one. In other words, the parent beta process in a R-HBP is a beta process for which the predictive process is an Indian buffet process. Formally,

$$B \sim \text{BP}(1, B_0)\,, \quad A_j \sim \text{BP}(\alpha_j, B)\,, \quad \mathbf{Z}_j^{i,:} \mid A_j \sim \text{BeP}(A_j)$$

Alternatively, using Eqs (1-2), we can write the above as

$$\beta_{(k)} \sim \text{StickIBP}(\tau_0) \tag{5}$$

$$\pi_{jk} \sim \text{beta}\left(\alpha_j \beta_{(k)}, \alpha_j\left(1 - \beta_{(k)}\right)\right), \mathbf{Z}_j^{i,:} \sim \text{BeP}(\pi_j) \tag{6}$$

3 Hierarchical Factor Analysis using R-HBP

We now consider joint factor analysis [7, 8] and use the R-HBP prior to infer the dimensionalities of factor subspaces. Our goal is to model *multiple* data matrices $\mathbf{X}_1, ..., \mathbf{X}_J$ jointly using a factor matrix $\Phi = [\phi_1, ..., \phi_K]$ where $\phi_k \in \mathbb{R}^M$ denotes the k-th factor. Some of the factors in Φ may be *shared* amongst the various data sources whereas others factors are specific to *individual* sources. For each $j = 1, \ldots, J$, $\mathbf{X}_j \in \mathbb{R}^{M \times N_j}$ has a representation $\mathbf{H}_j \in \mathbb{R}^{N_j \times K}$ in the subspace spanned by Φ along with the factorization error $\mathbf{E}_j$. Formally,

$$\Pi : \begin{cases} \mathbf{X}_1 = & \Phi \mathbf{H}_1^{\mathsf{T}} + \mathbf{E}_1 \\ \cdots & \cdots \\ \mathbf{X}_J = & \Phi \mathbf{H}_J^{\mathsf{T}} + \mathbf{E}_J \end{cases} \tag{7}$$

We allow the number of factors (K) to grow infinitely when more data is observed. To infer the value of K, we represent the matrix $\mathbf{H}_j$ as an element-wise multiplication of two matrices $\mathbf{Z}_j$ and $\mathbf{W}_j$, i.e. $\mathbf{H}_j = \mathbf{Z}_j \odot \mathbf{W}_j$, where $\mathbf{Z}_j^{ik} = 1$ implies the presence of factor ϕ_k for i-th data point $\mathbf{X}_j^{:,i}$ from source j and $\mathbf{W}_j^{ik}$ represents the corresponding coefficient or weight of the factor ϕ_k. The collection of matrices $\mathbf{Z}_1, \ldots, \mathbf{Z}_J$ are now modeled using R-HBP prior. In particular, we model i-th row of $\mathbf{Z}_j$ as a draw from a Bernoulli process parametrized by a beta process A_j, i.e. $\mathbf{Z}_j^{i,:} \sim \text{BeP}(\pi_j)$ where $\pi_j \triangleq [\pi_{j1}, \ldots, \pi_{jK}]$ and π_{jk} is defined in Eq (6). We note that sampling π_{jk} according to Eq (6) allows sharing of the factors ϕ_k's across data sources. This is because $\beta_{(k)}$ (the stick-breaking weights of B, i.e. $\beta_{(k)} \sim \text{StickIBP}(\tau_0)$) are used for all data sources. For the factor analysis parameters, we propose two different models : a Gaussian-Gaussian model (GGM) for real-valued data and a Poisson-gamma model (PGM) for count data. The whole model can be summarized as

$$\text{PGM} : \begin{cases} \Phi^{ik} \sim \text{gamma}\left(a_\phi, b_\phi\right) \\ \mathbf{W}_j^{i,:} \sim \Pi_{k=1}^{K} \text{gamma}\left(a_{w_j}, b_{w_j}\right) \\ \mathbf{X}_j^{:,i} \mid \Phi, \mathbf{Z}_j^{i,:}, \mathbf{W}_j^{i,:} \\ \sim \text{Poisson}\left(\Phi\left(\mathbf{Z}_j^{i,:} \odot \mathbf{W}_j^{i,:}\right)^{\mathsf{T}} + \lambda_j\right) \end{cases} \tag{8}$$

$$\text{GGM} : \begin{cases} \Phi^{ik} \sim \mathcal{N}\left(0, \sigma_\phi^2\right), \mathbf{W}_j^{i,:} \sim \mathcal{N}\left(\mathbf{0}, \sigma_{w_j}^2 \mathbf{I}\right) \\ \mathbf{X}_j^{:,i} \mid \Phi, \mathbf{Z}_j^{i,:}, \mathbf{W}_j^{i,:} \sim \mathcal{N}\left(\Phi\left(\mathbf{Z}_j^{i,:} \odot \mathbf{W}_j^{i,:}\right)^{\mathsf{T}}, \sigma_{n_j}^2 \mathbf{I}\right) \end{cases} \tag{9}$$

For reference purposes, we call the above model as *Nonparametric Hierarchical Factor Analysis* (NHFA) and the two variants as NHFA-PGM and NHFA-GGM.

4 Model Inference

For inference, we use a combination of slice sampling and collapsed Gibbs sampling. The main variables of interest

for NHFA model are $\mathbf{Z}$, $\mathbf{W}$, Φ, π, β and α_j (see Figure 1). In addition, we introduce an auxiliary variable ρ. The purpose of this auxiliary variable is to turn the infinite representation of beta process into a finite representation. We integrate out π and sample $\mathbf{Z}$, $\mathbf{W}$, Φ and β. Notation-wise, we use $\beta = \{\beta_k : k = 1,\ldots,K\}$, $\pi = \{\pi_j : j = 1,\ldots,J\}$ and a superscript attached to a variable followed by a '$-$' sign (e.g. $\mathbf{m}^{-jk}$, $\mathbf{Z}_j^{i,-k}$ etc) means a set excluding the variable indexed by the superscript. In the following, we describe the inference steps. ***Further details on some of the derivations are provided in the supplementary material [9].***

The joint distribution of β and $\mathbf{Z}$ can be written as

$$p(\beta,\mathbf{Z},\rho) = p(\beta)\,p(\rho \mid \mathbf{Z},\beta)\int_{\pi} p(\mathbf{Z} \mid \pi)\,p(\pi \mid \beta)\,d\pi \tag{10}$$

Let us assume that the auxiliary variable ρ is uniformly distributed as

$$p(\rho \mid \mathbf{Z},\beta) = \frac{1}{\beta^*}1(0 \le \rho \le \beta^*) \tag{11}$$

where β^* (a function of $\beta_{(1:\infty)}$ and $\mathbf{Z}$) is equal to the stick weight of the smallest active factor, i.e.

$$\beta^* = \min_{k|\exists i,\, \mathbf{Z}_j^{ik}=1} \beta_{(k)} \tag{12}$$

Using Eq (11), the conditional of $\mathbf{Z}$ given ρ and β can be written as

$$p(\mathbf{Z} \mid \rho,\beta) \propto \frac{1}{\beta^*}1(0 \le \rho \le \beta^*)\int_{\pi} p(\mathbf{Z} \mid \pi)\,p(\pi \mid \beta)\,d\pi \tag{13}$$

We note that given ρ, $\forall i$, $\mathbf{Z}_j^{ik} = 0$ if $\beta_{(k)} < \rho$ and therefore, we only need to update $\mathbf{Z}_j^{ik}$ for those values of k such that $\beta_{(k)} \ge \rho$ [17]. We now proceed to derive the full conditional distributions required for Gibbs sampling.

4.1 Sampling ρ

Let $K^\dagger$ denote the index such that all active features have index $k < K^\dagger$. The index $K^\dagger$ is also represented although it is an inactive feature. We sample ρ according to Eq (11). If the new value of ρ is such that $\beta_{(K^\dagger)} \ge \rho$, then we extend the stick breaking representation until $\beta_{(K^\dagger)} < \rho$. For the extended representation, Φ and $\mathbf{W}$ can be sampled from their prior distributions while $\beta_{(k)}$ are sampled from the following conditional distribution

$$\begin{aligned} &p\left(\beta_{(k)} \mid \beta_{(k-1)}, \mathbf{Z}^{:,\ge k} = 0\right) \\ &\propto p\left(\beta_{(k)} \mid \beta_{(k-1)}\right) p\left(\mathbf{Z}^{:,\ge k} = 0 \mid \beta_{(k)}\right) \end{aligned} \tag{14}$$

where $\mathbf{Z}^{:,\ge k} \triangleq \left\{\mathbf{Z}_j^{ik'} \mid \forall j,i \text{ and } k' \ge k\right\}$. The likelihood term in the above expression can be computed as

$$\begin{aligned} &p\left(\mathbf{Z}^{:,\ge k} = 0 \mid \beta_{(k)}\right) \\ &= \Pi_{j=1}^{J} p\left(\mathbf{Z}_j^{:,k} = 0 \mid \beta_{(k)}\right) p\left(\mathbf{Z}_j^{:,>k} = 0 \mid \beta_{(k)}\right) \end{aligned} \tag{15}$$

In the above expression, the first term of the right hand side (R.H.S.) requires marginalizing out π_{jk} and is computed as

$$\begin{aligned} &p\left(\mathbf{Z}_j^{:,k} = 0 \mid \beta_{(k)}\right) \\ &= \int_0^1 p\left(\mathbf{Z}_j^{:,k} = 0 \mid \pi_{jk}\right) p\left(\pi_{jk} \mid \beta_{(k)}\right) d\pi_{jk} \\ &= \frac{\Gamma(\alpha_j)\,\Gamma\left(\alpha_j\bar{\beta}_{(k)} + N_j\right)}{\Gamma\left(\alpha_j\bar{\beta}_{(k)}\right)\Gamma(\alpha_j + N_j)} \end{aligned} \tag{16}$$

where $\bar{\beta}_{(k)} \triangleq 1 - \beta_{(k)}$. For each $k \ge 1$, let us use l_k to define the mapping $\beta_{l_k} = \bar{\beta}_{(k)}$and let $\mathbf{L}_k = \{1,\ldots,K\} \setminus \{l_1,\ldots,l_k\}$, then the second term in the right hand side (R.H.S.) of Eq (15) can be computed as the following

$$\begin{aligned} &p\left(\mathbf{Z}_j^{:,>k} = 0 \mid \beta_{(k)}\right) \\ &= \lim_{K\to\infty} \int p\left(\beta_{\mathbf{L}_k} \mid \beta_{(k)}\right) p\left(\mathbf{Z}_j^{:,\mathbf{L}_k} = 0 \mid \beta_{\mathbf{L}_k}\right) d\beta_{\mathbf{L}_k} \\ &= \lim_{K\to\infty} \int \Pi_{l\in\mathbf{L}_k} p\left(\beta_l \mid \beta_{(k)}\right) p\left(\mathbf{Z}_j^{:,l} = 0 \mid \beta_l\right) d\beta_{\mathbf{L}_k} \\ &= \lim_{K\to\infty} \left(\int_0^{\beta_{(k)}} p\left(\beta_l \mid \beta_{(k)}\right) p\left(\mathbf{Z}_j^{:,l} = 0 \mid \beta_l\right) d\beta_l\right)^{K-k} \end{aligned} \tag{17}$$

In the above limmand, we first simplify the integral to the following

$$\begin{aligned} &\int_0^{\beta_{(k)}} p\left(\beta_l \mid \beta_{(k)}\right) p\left(\mathbf{Z}_j^{:,l} = 0 \mid \beta_l\right) d\beta_l \\ &= \frac{\Gamma(\alpha_j)}{\Gamma(\alpha_j + N_j)} \sum_{u_{jk}=0}^{N_j} \begin{bmatrix} N_j \\ u_{jk} \end{bmatrix} \alpha_j^{u_{jk}} \\ &\times \frac{\Pi_{t"=1}^{u_{jk}} t"}{\Pi_{t'=1}^{u_{jk}}\left(\frac{\tau}{K} + t'\right)} \left[1 + \frac{\tau}{K}\sum_{p=1}^{u_{jk}} \frac{\Pi_{t=1}^{p-1}\left(\frac{\tau}{K} + t\right)}{p!}\bar{\beta}_{(k)}^{p}\right] \end{aligned} \tag{18}$$

where $\begin{bmatrix} n \\ m \end{bmatrix}$ denotes the unsigned Stirling numbers of the first kind. In the above expression, we note that the term $\frac{\Gamma(\alpha_j)}{\Gamma(\alpha_j+N_j)}\sum_{u_{jk}=0}^{N_j}\begin{bmatrix} N_j \\ u_{jk} \end{bmatrix}\alpha_j^{u_{jk}} \times (...)$ can be written as

$$\begin{aligned} &\frac{\Gamma(\alpha_j)}{\Gamma(\alpha_j + N_j)} \sum_{u_{jk}=0}^{N_j} \begin{bmatrix} N_j \\ u_{jk} \end{bmatrix} \alpha_j^{u_{jk}} \times \left(\text{ term involving } u_{jk}\right) \\ &= \sum_{u_{jk}=0}^{N_j} \frac{\begin{bmatrix} N_j \\ u_{jk} \end{bmatrix}\alpha_j^{u_{jk}}}{\sum_{u'_{jk}=0}^{N_j}\begin{bmatrix} N_j \\ u'_{jk} \end{bmatrix}\alpha_j^{u'_{jk}}} \times \left(\text{ term involving } u_{jk}\right) \\ &= \sum_{u_{jk}=0}^{N_j} \tilde{\omega}_{u_{jk}} \times \left(\text{ term involving } u_{jk}\right) \end{aligned}$$

where $\sum_{u_{jk}=0}^{N_j}\tilde{\omega}_{u_{jk}}$ is equal to 1. Continuing from Eq (18) and substituting it in Eq (17), we can write

$$\begin{aligned} &p\left(\mathbf{Z}_j^{:,>k} = 0 \mid \beta_{(k)}\right) \\ &= \lim_{K\to\infty} \left(\sum_{u_{jk}=0}^{N_j} \tilde{\omega}_{u_{jk}} \frac{\Pi_{t"=1}^{u_{jk}} t"}{\Pi_{t'=1}^{u_{jk}}\left(\frac{\tau}{K} + t'\right)}\left(1 + \frac{\tau}{K}R_{u_{jk}}\right)\right)^{K-k} \end{aligned}$$

which simplifies to (see *supplementary material [9]*)

$$
\begin{aligned}
& p\left(\mathbf{Z}_j^{:,>k}=0 \mid \beta_{(k)}\right) \\
&= \lim_{K\to\infty}\left(1+\sum_{u_{jk}=0}^{N_j}\tilde{\omega}_{u_{jk}}\frac{\tau\left(T_{u_{jk}}-H_{u_{jk}}\right)}{K+\tau H_{u_{jk}}}\right)^{K-k} \\
&= \exp\left(\tau\sum_{u_{jk}=0}^{N_j}\tilde{\omega}_{u_{jk}}\left(T_{u_{jk}}-H_{u_{jk}}\right)\right) \qquad (19)
\end{aligned}
$$

where we have defined $R_{u_{jk}} \triangleq \sum_{p=1}^{u_{jk}} \frac{\Pi_{t=1}^{p-1}\left(\frac{\tau}{K}+t\right)}{p!}\bar{\beta}_{(k)}^{p}$, $T_{u_{jk}} \triangleq \sum_{p=1}^{u_{jk}} \frac{\bar{\beta}_{(k)}^{p}}{p}$ and the harmonic number $H_{u_{jk}} = \sum_{h=1}^{u_{jk}} \frac{1}{h}$. Plugging the likelihood of (19) and (16) into Eq (15) and using Eq (14), we obtain

$$
\begin{aligned}
& p\left(\beta_{(k)} \mid \beta_{(k-1)}, \mathbf{Z}^{:,\geq k}=0\right) \\
& \propto \tau\beta_{(k-1)}^{-\tau}\beta_{(k)}^{\tau-1}1\left(0\leq\beta_{(k)}\leq\beta_{(k-1)}\right)\times \\
& \Pi_{j=1}^{J}\frac{\Gamma\left(\alpha_j\bar{\beta}_{(k)}+N_j\right)}{\Gamma\left(\alpha_j\bar{\beta}_{(k)}\right)}\exp\left(\tau\sum_{u_{jk}=0}^{N_j}\tilde{\omega}_{u_{jk}}\left(T_{u_{jk}}-H_{u_{jk}}\right)\right) \\
& \propto \beta_{(k)}^{\tau-1}1\left(0\leq\beta_{(k)}\leq\beta_{(k-1)}\right)\times \\
& \Pi_{j=1}^{J}\sum_{v_{jk}=0}^{N_j}\begin{bmatrix}N_j\\ v_{jk}\end{bmatrix}\left(\alpha_j\bar{\beta}_{(k)}\right)^{v_{jk}}\exp\left(\tau\sum_{u_{jk}=0}^{N_j}\tilde{\omega}_{u_{jk}}\sum_{p=1}^{u_{jk}}\frac{\bar{\beta}_{(k)}^{p}}{p}\right)
\end{aligned}
$$

Introducing the auxiliary variables $\mathbf{v}=\left(v_{jk}:\forall j,k\right)$, we get

$$
\begin{aligned}
& p\left(\beta_{(k)} \mid \beta_{(k-1)}, \mathbf{Z}^{:,\geq k}=0, \mathbf{v}\right) \\
& \propto \beta_{(k)}^{\tau-1}\left(\bar{\beta}_{(k)}\right)^{\sum_j v_{jk}}\exp\left(\tau\sum_{u_{jk}=0}^{N_j}\tilde{\omega}_{u_{jk}}\sum_{p=1}^{u_{jk}}\frac{\bar{\beta}_{(k)}^{p}}{p}\right) \\
& 1\left(0\leq\beta_{(k)}\leq\beta_{(k-1)}\right) \qquad (20)
\end{aligned}
$$

The auxiliary variable v_{jk} can be sampled as

$$
p\left(v_{jk} \mid \mathbf{v}^{-jk}\right) \propto \begin{bmatrix}N_j\\ v_{jk}\end{bmatrix}\left(\alpha_j\bar{\beta}_{(k)}\right)^{v_{jk}} \qquad (21)
$$

Sampling from Eq (20) can be obtained using Adaptive Rejection Sampling (ARS) procedure as the distribution is *log-concave* in $\beta_{(k)}$. Sampling of auxiliary variable v_{jk} is straight-forward as it is sampling from a discrete distribution with finite support.

4.2 Sampling $\mathbf{Z}_j$

Given other variables, $\mathbf{Z}_j^{ik}$ (for $k=1,\ldots,K^{\dagger}-1$) can be sampled from the following Gibbs conditional posterior distribution

$$
\begin{aligned}
& p\left(\mathbf{Z}_j^{ik}=1 \mid \mathbf{Z}_j^{i,-k}, \mathbf{Z}^{-ji}, \mathbf{W}_j^i, \beta, \Phi, \mathbf{X}_j^i, \rho\right) \\
& \propto p\left(\rho \mid \mathbf{Z}_j^{ik}=1, \mathbf{Z}_j^{i,-k}, \mathbf{Z}^{-ji}, \beta\right) \\
& \times p\left(\mathbf{Z}_j^{ik}=1 \mid \mathbf{Z}_j^{-i,k}, \beta_{(k)}\right) p\left(\mathbf{X}_j^i \mid \mathbf{Z}_j^{ik}=1, \mathbf{Z}_j^{i,-k}, \mathbf{W}_j^i, \Phi\right) \\
& = \frac{n_j^{-i,k}+\alpha_j\beta_{(k)}}{\beta^{*}} p\left(\mathbf{X}_j^i \mid \mathbf{Z}_j^{ik}=1, \mathbf{Z}_j^{i,-k}, \mathbf{W}_j^i, \Phi\right) \qquad (22)
\end{aligned}
$$

where $n_j^{-i,k}=\sum_{i'\neq i}\mathbf{Z}_j^{i'k}$. In the above posterior expression, we note the dependency of β^{*} on the sample of $\mathbf{Z}_j^{ik}$.

4.3 Sampling $\beta_{(k)}$

To sample from the posterior of each $\beta_{(k)}$, we use auxiliary variable sampling [3, 18]. We define $\mathbf{m}=\left(m_{jk}:\forall j,k\right), \mathbf{l}=\left(l_{jk}:\forall j,k\right)$ and $n_{jk}=\sum_i \mathbf{Z}_j^{i,k}$ where $m_{jk}\in\{0,1,\ldots,n_{jk}\}$, $l_{jk}\in\{0,1,\ldots,N_j-n_{jk}\}$, and sample $\beta_{(k)}$ (for $k=1,\ldots,K^{\dagger}$) and auxiliary variables $\mathbf{m},\mathbf{l}$ as below

$$
\begin{aligned}
& p\left(\beta_{(k)} \mid \beta^{-(k)}, \mathbf{m}, \mathbf{l}, \mathbf{Z}, \rho\right) \\
& \propto \beta_{(K^{\dagger})}^{\tau_0}\beta_{(k)}^{m_k-1}\left(1-\beta_{(k)}\right)^{l_k}1\left(\beta_{(k+1)}\leq\beta_{(k)}\leq\beta_{(k-1)}\right) \qquad (23)
\end{aligned}
$$

$$
p\left(m_{jk} \mid \mathbf{m}^{-jk}, \mathbf{l}, \beta, \mathbf{Z}, \rho\right) \propto \begin{bmatrix}n_{jk}\\ m_{jk}\end{bmatrix}\left(\alpha_j\beta_k\right)^{m_{jk}} \qquad (24)
$$

$$
p\left(l_{jk} \mid \mathbf{l}^{-jk}, \mathbf{m}, \beta, \mathbf{Z}, \rho\right) \propto \begin{bmatrix}\bar{n}_{jk}\\ l_{jk}\end{bmatrix}\left(\alpha_j\bar{\beta}_k\right)^{l_{jk}} \qquad (25)
$$

where $\mathbf{m}_k \triangleq \sum_j m_{jk}$, $\mathbf{l}_k \triangleq \sum_j l_{jk}$, $\bar{n}_{jk}=N_j-n_{jk}$. For detailed derivation, see *supplementary material [9]*.

4.4 Sampling Parameters Φ, $\mathbf{W}_j$ and λ_j for NHFA-PGM model

Under the model described in Eq (8), Gibbs conditional posterior of Φ can be written as

$$
\begin{aligned}
& p\left(\Phi^{i,:} \mid \mathbf{Z}_{1:J}, \mathbf{W}_{1:J}, \mathbf{X}_{1:J}, \lambda_{1:J}, a_\phi, b_\phi, \mathbf{s}\right) \\
& \propto \Pi_{k=1}^{K^{\dagger}}\left(\Phi^{ik}\right)^{a_\phi+\sum_{j,l}s_j^{ilk}-1}\exp\left\{-\left(b_\phi+\sum_{j,l}\mathbf{H}_j^{lk}\right)\Phi^{ik}\right\} \qquad (26)
\end{aligned}
$$

The auxiliary variables $\mathbf{s}=\left\{s_j^{ilk},\forall j,l\right\}_{k=1}^{K^{\dagger}+1}$ are drawn as

$$
\begin{aligned}
& p\left(s_j^{il1},\ldots s_j^{ilK^{\dagger}}, s_j^{il\left(K^{\dagger}+1\right)} \mid \text{rest}\right) \\
& \propto \frac{\mathbf{X}_j^{il}!}{s_j^{il1}!\ldots s_j^{ilK}!s_j^{il\left(K^{\dagger}+1\right)}!}\Pi_{k=1}^{K^{\dagger}}\left(\Phi^{ik}\mathbf{H}_j^{lk}\right)^{s_j^{ilk}}\lambda_j^{s_j^{il\left(K^{\dagger}+1\right)}} \\
& \qquad \text{s.t. } \sum_{p=1}^{K^{\dagger}+1}s_j^{ilp}=1 \qquad (27)
\end{aligned}
$$

Gibbs sampling update for $\mathbf{W}_{ji}$ conditioned on the remaining variables is given as

$$
\begin{aligned}
& p\left(\mathbf{W}_j^{l,:} \mid \mathbf{Z}_j, \Phi, \mathbf{X}_j, \lambda_j, a_{w_j}, b_{w_j}, \mathbf{t}\right) \\
& \propto \Pi_{k=1}^{K^{\dagger}}\left(\mathbf{W}_j^{lk}\right)^{a_{w_j}+\sum_i t_j^{ilk}-1} \\
& \times \exp\left\{-\left(b_{w_j}+\sum_i\left(\Phi\mathbf{D}_{\mathbf{Z}_j^{i,:}}\right)^{ik}\right)\mathbf{W}_j^{lk}\right\} \qquad (28)
\end{aligned}
$$

where $\mathbf{D}_{\mathbf{Z}_j^{l,:}}$ denotes a diagonal matrix constructed from $\mathbf{Z}_j^{l,:}$ and the auxiliary variables $\mathbf{t} = \left\{ t_j^{ilp}, \text{for each } i \right\}_{p=1}^{K^\dagger+1}$ are sampled as below

$$p\left(t_j^{il1}, \ldots t_j^{ilK^\dagger}, t_j^{il\left(K^\dagger+1\right)} \mid \text{rest}\right) \propto \frac{\mathbf{X}_j^{il}!}{t_j^{il1}! \ldots t_j^{ilK^\dagger}! t_j^{il\left(K^\dagger+1\right)}!} \Pi_{k=1}^{K^\dagger} \left(\Phi^{lk}\mathbf{H}_j^{lk}\right)^{t_j^{ilk}} \lambda_j^{t_j^{il\left(K^\dagger+1\right)}} \quad \text{s.t. } \sum_{p=1}^{K^\dagger+1} t_j^{ilp} = 1 \tag{29}$$

Gibbs sampling of λ_j remains similar to the variables Φ and $\mathbf{W}_j$. The required posterior distributions are given as

$$p\left(\lambda_j \mid \mathbf{Z}_{1:J}, \mathbf{W}_{1:J}, \Phi, \mathbf{X}_{1:J}, \mathbf{r}\right) \propto \lambda_j^{a_{\lambda_j} + \sum_{i=1}^{M}\sum_{l=1}^{N_j} r_j^{il} - 1} \exp\left\{-\left(b_{\lambda_j} + MN_j\right)\right\} \tag{30}$$

For each i, l, the auxiliary variables r_j^{il} can be sampled as

$$r_j^{il} \mid \text{rest} = \text{Binomial}\left(\mathbf{X}_j^{il}, \frac{\lambda_j}{\Phi^{i,:}\left(\mathbf{H}_j^{l,:}\right)^\top + \lambda_j}\right) \tag{31}$$

4.5 Sampling Parameters Φ, $\mathbf{W}_j$, σ_{w_j} and σ_{n_j} for NHFA-GGM model

Gibbs sampling updates for these parameters remains same as the updates described in [8].

4.6 Sampling α_j

Once again, Gibbs sampling updates for α_j remains same as the updates described in [8]. The hyperparameter α_j controls the variation of A_j (source-specific beta process) around B (parent beta process). As α_j vary from a low to high value, its concentration on the random distribution B increases. Since the random distribution B is shared across different data sources, this increases the probability of sharing more and more factors.

4.7 Predictive Likelihood

Let us denote the test data from the j-th source by matrix $\tilde{\mathbf{X}}_j$ and the corresponding matrix factorization as $\tilde{\mathbf{X}}_j = \Phi\left(\tilde{\mathbf{Z}}_j \odot \tilde{\mathbf{W}}_j\right) + \tilde{\mathbf{E}}_j$, then the Monte Carlo approximation of predictive likelihood can be written as

$$p\left(\tilde{\mathbf{X}}_j \mid \mathbf{X}_{1:J}\right) \approx \frac{1}{LR}\sum_{r=1}^{R}\sum_{l=1}^{L} p\left(\tilde{\mathbf{X}}_j \mid \Phi^{[l]}, \tilde{\mathbf{Z}}_j^{[r]}, \tilde{\mathbf{W}}_j^{[r]}\right) \tag{32}$$

where L is the number of training samples $\left\{\Phi^{[l]}\right\}$ and R is the number of test samples $\left\{\tilde{\mathbf{Z}}_j^{[r]}, \tilde{\mathbf{W}}_j^{[r]}\right\}$.

5 Experiments

We carry out a variety of experiments to demonstrate the effectiveness of the proposed NHFA model. To illustrate the behavior of our model, we first perform experiments with a synthetic dataset, for which the true dimensionality of subspaces and other parameters are known. Through these experiments, we show the correct recovery of these parameters. Next, we demonstrate the usefulness of our model for two real-world tasks – text modeling and content based image retrieval. For both synthetic and real-world tasks, the priors for hyperparameters are chosen as the following : $\tau_0 = 1$, $\alpha_j \sim \text{gamma}(1,1)$. For NHFA-GGM, both the shape and the scale parameters of gamma priors for σ_ϕ, σ_{nj} and σ_{wj} were set to 1. For NHFA-PGM, since we expect the results to be sparse we set the shape parameters of hyperparameters as $a_\phi = 1$, $a_{w_j} = 1$ whereas the scale parameters were sampled as : $b_\phi \sim \text{gamma}\left(1, 1/\mu_\phi\right)$, $b_{w_j} \sim \text{gamma}\left(1, 1/\mu_{w_j}\right)$, where $\mu_\phi \triangleq \frac{1}{MK}\sum_{i,k}\Phi^{ik}$ and $\mu_{w_j} \triangleq \frac{1}{N_j K}\sum_{l,k}\mathbf{W}_j^{lk}$. *Supplementary material* provides further details.

5.1 Experiments-I : Synthetic Data

We create a synthetic dataset similar to [8] so that we can show the benefits of our model vis-à-vis the model considered in [8]. For this dataset, we create twelve 100-D binary factors and distribute them across the two data sources termed as $\mathscr{D}_1$ and $\mathscr{D}_2$. The first four factors were used by the data points from $\mathscr{D}_1$, the next four factors were used by the data points from $\mathscr{D}_2$ while the last four factors were shared by data points across both $\mathscr{D}_1$ and $\mathscr{D}_2$. We generated the mixing configuration matrices $\mathbf{Z}_1$ and $\mathbf{Z}_2$ randomly with discrete support $\{0,1\}$. The weight/coefficient matrices of the two sources, i.e. $\mathbf{W}_1$ and $\mathbf{W}_2$ were sampled from gamma distribution with shape and scale parameters 1 and 2 respectively. The using these parameters along with noise (with rate parameter 0.1 for both sources), the data for both $\mathscr{D}_1$ and $\mathscr{D}_2$ were generated from Poisson distributions according to the generative model described in section 3.

To verify the correctness of the inference, we run the slice sampler along with Gibbs updates as detailed in section 4 starting with the value of $K^\dagger$ as one. We observe that the sampler converges to the true value of the number of factors (i.e. $K^\dagger = 12$) in less than 100 iterations. However, we run the sampler longer to verify that the mode of the posterior over the number of factors remains at this true value. It can be seen from Figure 2 that the model correctly learns all the factors and their scores automatically.

To compare the computational efficiency of our slice sampler with the approximate Gibbs sampler of [8], we see that both methods need to sample $\{\beta, \mathbf{Z}, \Phi, \mathbf{W}\}$. Sampling Φ and $\mathbf{W}$ remain identical in both cases. Only sampling of β and $\mathbf{Z}$ differ. In case of the approximate Gibbs sampler, β conditioned on $\{\mathbf{m}, \mathbf{l}, \mathbf{Z}\}$ is drawn from a beta distribution whereas in the slice sampler, β conditioned on $\{\rho, \mathbf{m}, \mathbf{l}, \mathbf{Z}\}$ uses adaptive rejection sampling (ARS) (see Eq (23)). Although not as fast as sampling from a beta distribution, ARS is quite efficient due to sampling in one dimensional space. When comparing sampling of $\mathbf{Z}$, the slice sampler is more efficient than the approximate Gibbs sampler. The main difference lies in sampling the number of new factors

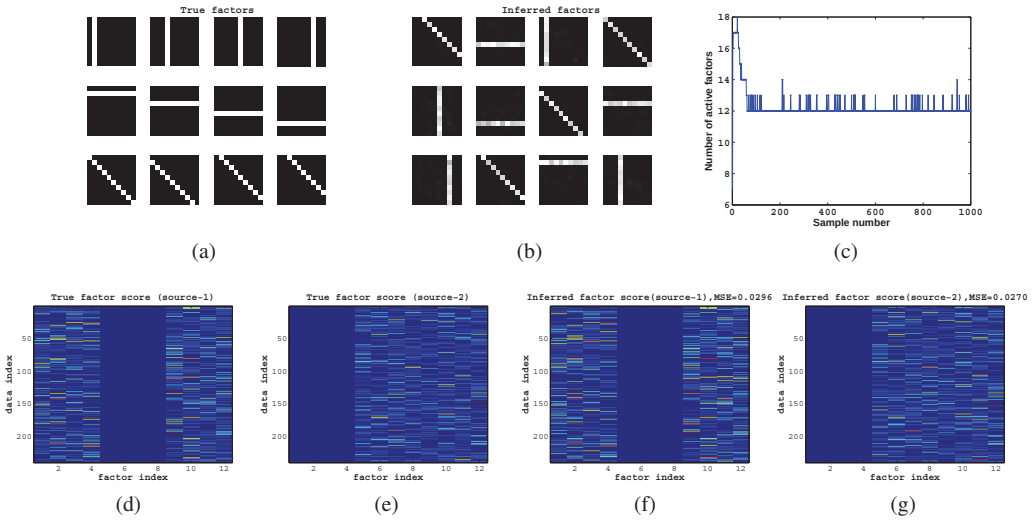


Figure 2: Synthetic data experiments (a) *true* factors (b) *inferred* factors (c) Number of active factors vs. Gibbs iterations (d) *true* factor scores for source-1 (e) *true* factor scores for source-2 (f) *inferred* factor scores for source-1 (g) *inferred* factor scores for source-2; for an easy comparison with (d) and (e), columns of (f) and (g) are permuted according to the mapping between (a) and (b).

for i-th data point from j-th source (say F_{ji}). In slice sampling, F_{ji} grows gradually based on the slice variable and requires sampling of F_{ji} new β_k variables using ARS. On the other hand, the approximate Gibbs sampler, to sample F_{ji}, approximates an intractable integral (Eq (3.22) in [8]) using Monte Carlo samples. For the synthetic experiments described above, slice sampler takes about *100 iterations* to converge and runs in **6.67** minutes. On the other hand, the approximate Gibbs sampler [8] requires about *1000 iterations* to converge to the true number of factors taking totally **52.3** minutes. This timing analysis is performed using a Windows PC with Intel i7@3.4 GHz and 8 GB RAM.

5.2 Experiments-II : Real Data

5.2.1 Results using NIPS 0-12 Dataset

Our first real-world dataset is the NIPS 0-12 dataset, which contains the articles from the proceedings of *Neural Information Processing Systems* (NIPS) conference published between 1988 and 1999. In this dataset[1], text articles are divided into nine different sections/tracks plus one miscellaneous section/track. We work with nine sections which are Cognitive Science (CS), Neuroscience (NS), Learning Theory (LT), Algorithms and Architecture (AA), Implementations (IM), Speech and Signal Processing (SSP), Visual Processing (VP), Applications (AP) and Control, Navigation and Planning (CNP). We treat each section as one data source and generate nine different term-document matrices. In doing so, we ensure that these matrices use a common dictionary for terms[2].

[1] A processed version of this dataset is made available by Y. W. Teh at http://www.gatsby.ucl.ac.uk/~ywteh/research/data.html.

[2] It is possible to make a common dictionary by merging the terms from all the sections.

Through this dataset, we intend to show the strength of our model for transfer learning. For this, we choose section VP as target data source while other sections (one at a time) as auxiliary data sources. We follow this scheme for *two* reasons. *First*, we believe that although there may be underlying sharing across different sections, each section has its own focus and differs in distribution. NHFA exploits the sharing across different sections while still retaining the focus of individual section by maintaining a hierarchy. *Second*, this allows us to compare our results with two baselines (i.e. Gupta et al [8] and Teh et al [18]) as they use the same dataset with similar settings.

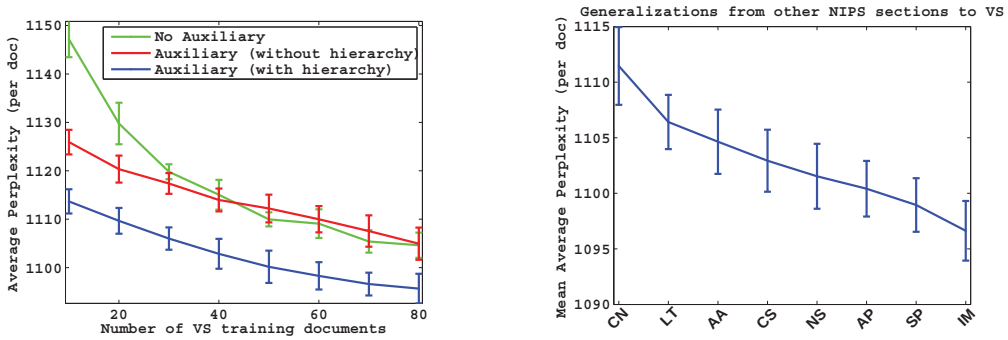
Out of 1564 articles in total across nine sections, we randomly select 80 articles from each section and use them for training. Similar to [8, 18], the test set is chosen from VP section (consisting of 44 articles) and kept fixed throughout the experimentation with NIPS 0-12 dataset. On average, the number of words per article are approximately 1000. We compute the perplexity on the test set and report the performance in terms of *perplexity per document*. Given the training data $\mathbf{X}_{1:J}$ and a test set $\tilde{\mathbf{X}}_j$ from j-th data source, perplexity per document (PPD) is defined as

$$\text{PPD}\left(\tilde{\mathbf{X}}_j\right) = \exp\left(-\frac{1}{\tilde{N}}\log\, p\left(\tilde{\mathbf{X}}_j \mid \mathbf{X}_{1:J}\right)\right) \qquad (33)$$

where $\tilde{N}$ denotes the number of documents in the test set.

Baseline Methods To compare the proposed model with other related works, we choose *three* baselines.

- Baseline-1a ["No auxiliary"]: This baseline is a linear Poisson-gamma based factor analysis model which totally relies on the target data and does not use any auxiliary data for training.



(a) (b)

Figure 3: Perplexity results using NIPS 0-12 dataset (a) perplexity for the test data from VP section vs. number of VP training documents (averaged over 10 runs); for Baseline-1a, Baseline-1b and the proposed NHFA-PGM (b) mean average perplexity for the test data from VP section (averaged over 10 runs) versus different NIPS auxiliary sections (except VP) using the proposed NHFA-PGM.

- Baseline-1b ["Auxiliary - without hierarchy"]: This baseline is a linear Poisson-gamma based factor analysis model similar to the first baseline but also uses auxiliary data simply by augmenting them with the target data and does not maintain any hierarchy.
- Baseline-2 [NHFA-GGM (approx.)]: This baseline is the hierarchical model recently proposed in Gupta et al [8]. This model attempts to learn both shared and individual subspace using a hierarchical model. The data distributions for this model are similar to the NHFA-GGM model proposed in section 2, however it uses a series of approximations for inference.
- Baseline-3 [HDP]: This baseline is the hierarchical Dirichlet process proposed in [18]. This model uses a hierarchy for sharing topics across different groups.

Experimental Results We run the proposed NHFA-PGM and the baseline models and compute the perplexity per doc (PPD) values over 10 independent runs. For each run, we select VP section as the target source while other sections as auxiliary (one at a time) and compute perplexity values for increasing number of target training articles varying from 10 to 80 at a step of 10. By doing this, our intention is to see at what point, the auxiliary data ceases to improve the generalization performance.

Figure 3 depicts the perplexity results for the proposed model and compares with baseline-1a and baseline-1b. It can be seen from Figure 3a that the proposed NHFA-PGM model achieves significantly lower perplexity compared to the two baselines irrespective of the number of training articles from VP section. When comparing the results between the baseline-1a and the baseline 1b, one can see that as long as the number of training documents from VP section are less than 50, the performance of baseline-1b is better than that of baseline 1a. This goes according to our intuition as the auxiliary source help learning some patterns. But, as soon as there are enough training articles from the VP section, simply combining auxiliary data does not help and the performance of baseline-1b goes lower than that of baseline-1a. This clearly shows that the baseline-1b is prone to negative transfer learning whereas the same is not true for the proposed NHFA-PGM model which retains low perplexity values compared to baseline-1a. Figure 3b focuses on the results of NHFA-PGM model and provides the mean average perplexity for each auxiliary section (a single figure measure computed by averaging the perplexity values along increasing number of training documents). It can be seen from the figure that the top three auxiliary sections that help maximum generalization are IM, SP and AP in decreasing order of performance.

Table 1 lists a comparison of the proposed NHFA-PGM and NHFA-GGM models with baseline-2 and baseline-3. One can clearly see that since the NIPS 0-12 data matrices are count data, both NHFA-PGM and the baseline-3 achieve significantly lower perplexity compared to NHFA-GGM and baseline-2. The proposed NHFA-PGM clearly outperforms the baseline-3, suggesting that Poisson-gamma model may be a better fit to the count data than topic models. When comparing the perplexity values of baseline-2 and NHFA-GGM, NHFA-GGM achieves better performance. This gain in performance is attributed to the exact inference made possible by slice sampling.

Method	*Average Log Perplexity (per doc)*
NHFA-GGM (approx.) [8]	6232.5 ± 164.0
Proposed NHFA-GGM	5945.3 ± 16.8
HDP [18]	2862.5 ± 268.3
Proposed NHFA-PGM	$\mathbf{1102.9 \pm 6.5}$

Table 1: Comparison of various hierarchical models in terms of average log perplexity per doc (PPD) using NIPS 0-12 dataset.

5.2.2 Results using NUS-WIDE Animal Dataset

Our second dataset is a subset of the NUS-WIDE [2] dataset. This subset comprises of 3411 images involving 13 animals. We use six different low-level features [2] namely 64-D color histogram, 144-D color correlogram, 73-D edge direction histogram, 128-D wavelet texture, 225-D block-wise color moments. We construct a real-valued feature matrix for each animal category and treat it as a separate data source under our hierarchical framework. This is done with a belief that features of different animal categories vary in their distributions. Out of 3411 images, 2054 images are used for training while the remaining images are held for testing. This is done for comparing with [21, 22, 1, 8] where an *identical* training and test settings were used.

Unlike NIPS 0-12 data, the feature matrices for NUS-WIDE images are real valued. Therefore, we use NHFA-GGM model (instead of NHFA-PGM model) to learn the factor matrix Φ and $\mathbf{H}_j$ (for $j = 1,\ldots,13$). Having learnt the matrix Φ, we construct the test matrix $\tilde{\mathbf{X}}$ from the test data and compute its subspace coefficient matrix $\tilde{\mathbf{H}} = \left[\tilde{\mathbf{h}}_1,\ldots,\tilde{\mathbf{h}}_{\tilde{N}}\right]$ such that $\tilde{\mathbf{X}} \approx \Phi\tilde{\mathbf{H}}$. To retrieve the similar images for the r-th test image, cosine similarity between its subspace coefficient vector $\tilde{\mathbf{h}}_r$ and the subspace coefficient matrices of the training data (i.e. $\mathbf{H}_j$ for each $j = 1,\ldots,13$) is computed. The retrieved images are ranked in decreasing order of their similarity values.

Evaluation Measures and Baselines To evaluate the performance of the proposed method, we use standard precision-scope curve[3] and mean average precision (MAP). We compare the result of the proposed NHFA-GGM with the recent state-of-the-art multi-view learning techniques [1, 21, 22] and "NHFA-GGM (approx.)" model [8] (the model used as baseline-2 for NIPS 0-12 experiments).

Experimental Results Table 2 compares the proposed NHFA-GGM with the recent works [1, 21, 22, 8] on image retrieval using NUS-WIDE animal dataset. This comparison is based on the mean average precision (MAP) values presented in [1, 8]. We note that the dataset used for generating these results (including the test set) is *identical*. For the works in [1, 21, 22], the MAP values are reported using 60 topics. The NHFA-GGM (approx.) model of [8] is a nonparametric model and reports the total number of shared and individual factors varying between 5 to 8. On the contrary, our proposed NHFA-GGM, which is exact version of NHFA-GGM (approx.), uses 30 to 40 factors typically. It can be clearly seen from the Table 2, that NHFA-GGM outperforms all the baselines. Focusing on the exact and approximate models, we report the precision scope curve which shows that NHFA-GGM model achieves clearly higher precision at initial scope levels which is often quite important in search applications as users are typically more interested in top few search results.

[3] Precision scope (P@N) curve reports precision values for top N retrieved items.

Method	*Mean Average Precision (MAP)*
DWH [21]	0.153
TWH [22]	0.158
MMH [1]	0.163
NHFA-GGM (approx.) [8]	0.1789 ± 0.0128
Proposed NHFA-GGM	$\mathbf{0.1945 \pm 0.0115}$

Table 2: Comparison of image retrieval results with recent state-of-the art multi-view learning and hierarchical modeling techniques using NUS-WIDE animal dataset.

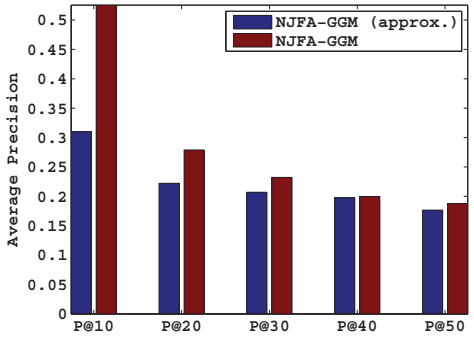


Figure 4: Comparison of the retrieval performances in terms of precision-scope (P@N) curve for NHFA-GGM and NHFA-GGM (approx.)[8].

6 Conclusion

We have presented a novel slice sampler for restricted hierarchical beta process (R-HBP). A key feature of the proposed sampler is that it keeps the inference tractable, even when the prior and the likelihood are non-conjugate – therefore, it can be applied for joint modeling of multiple data sources in more elaborate settings. Another key feature of this sampler is that it offers an exact inference and does not need a series of approximations used by standard Gibbs samplers [11, 16, 8]. We have combined this sampler with two hierarchical factor analysis data models – linear Poisson-gamma model and linear Gaussian-Gaussian model for modeling count data and real-valued data respectively. To show the utility of the proposed models, we applied them for learning shared and individual subspaces across multiple data sources where unlike parametric models, the subspace dimensionalities were learned automatically. Using two real world datasets (NIPS 0-12 and NUS-WIDE animal datasets), we have demonstrated that our method outperforms recent state-of-the-art methods for text modeling and content based image retrieval. Further, the slice sampling derived in this paper is general and can be utilized for other matrix factorizations.

References

[1] N. Chen, J. Zhu, and E.P. Xing. Predictive subspace learning for multi-view data: a large margin approach. *Advances in Neural Information Processing Systems*, pages 361–369, 2010.

[2] T.S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. NUS-WIDE: A real-world web image database from national university of singapore. *Proceeding of the ACM International Conference on Image and Video Retrieval*, pages 48:1–48:9, 2009.

[3] P. Damien, J. Wakefield, and S. Walker. Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(2):331–344, 1999.

[4] B. De Finetti. *Theory of probability: A critical introductory treatment. Vol. 2*. Wiley, 1990.

[5] D. Görür, F. Jäkel, and C.E. Rasmussen. A choice model with infinitely many latent features. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 361–368. ACM, 2006.

[6] T. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. *Advances in Neural Information Processing Systems*, 18:475, 2006.

[7] S.K. Gupta, D. Phung, B. Adams, and S. Venkatesh. A Bayesian framework for learning shared and individual subspaces from multiple data sources. In *Advances in Knowledge Discovery and Data Mining, 15th Pacific-Asia Conference*, pages 136–147, 2011.

[8] S.K. Gupta, D. Phung, and S. Venkatesh. A Bayesian nonparametric joint factor model for learning shared and individual subspaces from multiple data sources. In *Proceedings of 12th SIAM International Conference on Data Mining*, pages 200–211, 2012.

[9] S.K. Gupta, D. Phung, and S. Venkatesh. Supplementary material for UAI-12 publication "A slice sampler for restricted hierarchical beta process with applications to shared subspace learning". *https://sites.google.com/site/sunilsun133/uai12-sm.pdf*, 2012.

[10] N.L. Hjort. Nonparametric Bayes estimators based on beta processes in models for life history data. *The Annals of Statistics*, 18(3):1259–1294, 1990.

[11] H. Ishwaran and L.F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.

[12] Y. Kim. Nonparametric Bayesian estimators for counting processes. *The Annals of Statistics*, 27(2):562–588, 1999.

[13] J.F.C. Kingman. Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78, 1967.

[14] D. Knowles and Z. Ghahramani. Infinite sparse factor analysis and infinite independent components analysis. *Independent Component Analysis and Signal Separation*, pages 381–388, 2007.

[15] E. Meeds, Z. Ghahramani, R.M. Neal, and S.T. Roweis. Modeling dyadic data with binary latent factors. *Advances in Neural Information Processing Systems*, 19:977–984, 2007.

[16] J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 777–784, 2009.

[17] Y.W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. *Journal of Machine Learning Research - Proceedings Track*, 2:556–563, 2007.

[18] Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[19] R. Thibaux and M.I. Jordan. Hierarchical beta processes and the Indian buffet process. *Journal of Machine Learning Research - Proceedings Track*, 2:564–571, 2007.

[20] S.G. Walker. Sampling the Dirichlet mixture model with slices. *Communications in Statistics–Simulation and Computation*, 36(1):45–54, 2007.

[21] E. Xing, R. Yan, and A.G. Hauptmann. Mining associated text and images with dual-wing harmoniums. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 633–641, 2005.

[22] J. Yang, Y. Liu, E.X. Ping, and A.G. Hauptmann. Harmonium models for semantic video representation and classification. *Proceedings of 7th SIAM International Conference on Data Mining*, pages 1–12, 2007.

Semantic Understanding of Professional Soccer Commentaries

Hannaneh Hajishirzi, Mohammad Rastegari, Ali Farhadi, and Jessica K. Hodgins
{hannaneh.hajishirzi, mrastega, afarhadi, jkh}@cs.cmu.edu
Disney Research Pittsburgh and Carnegie Mellon University

Abstract

This paper presents a novel approach to the problem of semantic parsing via learning the correspondences between complex sentences and rich sets of events. Our main intuition is that correct correspondences tend to occur more frequently. Our model benefits from a discriminative notion of similarity to learn the correspondence between sentence and an event and a ranking machinery that scores the popularity of each correspondence. Our method can discover a group of events (called *macro-events*) that best describes a sentence. We evaluate our method on our novel dataset of professional soccer commentaries. The empirical results show that our method significantly outperforms the state-of-the-art.

1 Introduction

This paper addresses the problem of understanding professional commentaries of soccer games. Computational understanding of such domains has crucial impact in automatic generation of commentaries and also in game analysis and strategic planning. To this end, one needs to infer the semantics of natural language text; this is an extremely challenging problem. Understanding professional soccer commentaries further introduces interesting and challenging issues. For example, commentators do not typically talk about all the events of the game, selecting what is important. Also, they use a variety of phrases to report similar events. For example, a simple event of "A passes to B" can be commentated in several different ways: "A feeds B", "A and B in a nice combination", "A, what a beautiful way to B". Further, in some cases, commentators create a group of events and only mention a *macro-event*. For example, instead of saying "A passes to B, B passes to C, and C passes to D", the commentators report this whole sequence of events as "Team X is coming forward" or "nice attack by X". Also, professional commentators report several statistics and related information about the league, players, stadium, and weather during less interesting segments of a play.

A general solution to understanding such a complex phenomenon requires infering about game-related events, reasoning in terms of very complex paraphrases, and also forming high-level understandings of game events. Most recent work in semantic parsing of natural language translates individual sentences into the underlying meaning representations. Meaning representations are usually logical forms represented with events or relations among entities. The problem of semantic parsing can be formulated as learning to map between sentences and meaning representation in a supervised fashion [Zettlemoyer and Collins, 2005]. One can decrease the amount of supervision in specific controlled domains, such as RoboCup soccer [Chen *et al.*, 2010; Hajishirzi *et al.*, 2011] and Windows help instructions [Branavan *et al.*, 2009]. Recently, [Liang *et al.*, 2009] introduce a general semantic parsing technique that is not restricted to a specific domain, but is not scalable to large datasets due to the complexity of the model. In this paper, we introduce an algorithm that does not require domain-specific knowledge and is scalable to larger datasets.

We formulate the problem of understanding soccer commentaries as learning to align sentences in commentaries to a list of events in the corresponding soccer game. Our approach does not need expensive supervision in terms of correspondences between sentences and events. Similar to previous work [Liang *et al.*, 2009; Chen *et al.*, 2010], we use loose temporal alignments between sentences in commentaries and events of games. We pair sentences with several events that occur in the rough temporal vicinities of the sentences. Each pair consists of a sentence and a corresponding event. We then try to distinguish between correct and incorrect pairs. We rank pairs based on how consistently they appear in other places. We use a discriminative notion of similarity to reason about repetitions of pairs of sentences and events. The core intuition is that,

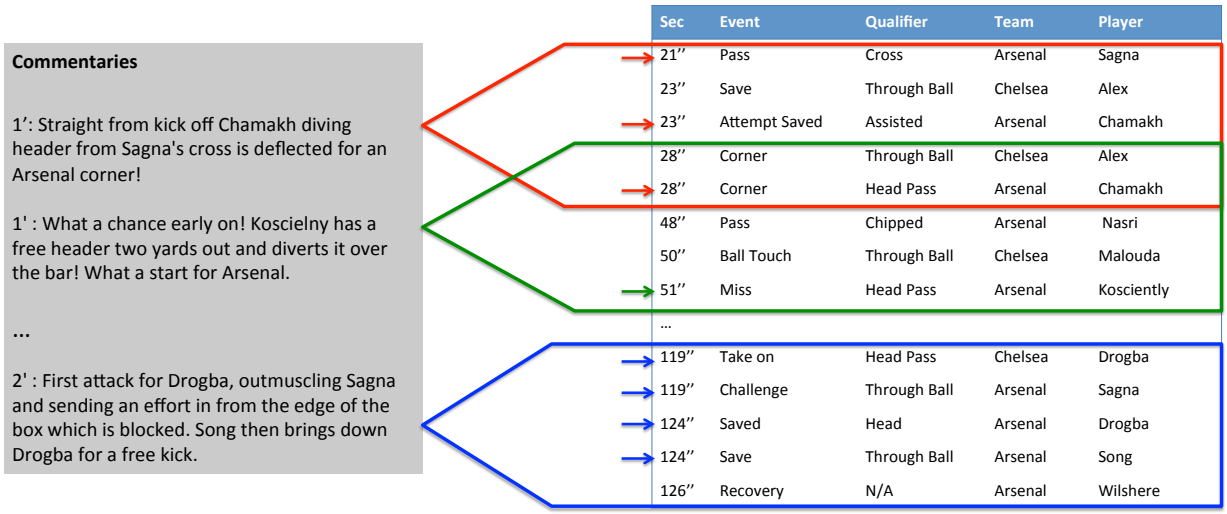


Figure 1: Examples of sentences in the commentaries with corresponding buckets of events. The correct correspondences in each bucket are marked with arrows.

under appropriate similarity metrics, correct pairs of sentences and events appear more often across several games. Our model is capable of forming macro-events and matching a group of events to a sentence. Experimental evaluations show that our method significantly outperforms the state of the art in our dataset and a benchmark of RoboCup soccer dataset. Our qualitative results demonstrate how our method can form macro-events and reason about complex paraphrases. We also introduce a dataset of eight English Primier League games with their commentaries and the accurate log of the game events. For evaluation purposes, we manually aligned sentences with events. Our dataset is publicly available.

1.1 Related Works

There are several approaches that learn to map natural language texts to meaning representations. Some researchers use full supervision in general and controlled domains [Zettlemoyer and Collins, 2005; Ge and Mooney, 2006; Snyder and Barzilay, 2007] . More recent work reduces the amount of required supervision in mapping sentences to meaning representations in controlled domains. [Kate and Mooney, 2007] introduce an Expectation Maximization (EM) approach with weak supervision in the GeoQuery and Child corpora. [Branavan *et al.*, 2009; Vogel and Jurafsky, 2010] use a reinforcement learning approach to map Windows help or navigational instructions to real events in the world. The supervision is weakly provided through interactions with a physical environment. [Chen *et al.*, 2010; Hajishirzi *et al.*, 2011] introduce EM-like approaches to map RoboCup soccer commentaries to real events of the game. They use weak supervision through the temporal alignments between sentences and events or the domain knowledge about soccer events. [Poon and Domingos, 2009] introduce an unsupervised method that finds is-a relationships among the entities in the domain by taking advantage of clustering structures of sentences and arguments. Most previous work take advantage of the special properties of the underlying domains and cannot be applied to the commentaries of real soccer games due to the complexity of the domain.

The closest work to ours are [Liang *et al.*, 2009; Bordes *et al.*, 2010] that introduce an interesting generative approach and a ranking system to map texts to meaning representations, respectively. The generative approach of [Liang *et al.*, 2009] is general, but it is not scalable as we demonstrate in our experiments. The system of [Bordes *et al.*, 2010] uses different models and objectives for ranking functions. Also, these two methods, as well as most previous work, has a one-to-one restriction in mapping between segments and events. This is not a well-suited assumption for real-world domains like professional soccer commentaries. One exception is the work of [Branavan *et al.*, 2010] that finds high-level comments which is specific to Windows instructions.

2 Problem Definition

The input to our problem is a representation of temporal evolution of the world state (sequence of events) and a natural language text (sequence of sentences). We use a rough alignment between every sentence S_i and a list of events $B(S_i)$; we call this list of events a *bucket* associated with the sentence S_i. We form a *pair* by combining a sentence S_i with an event in the corresponding bucket $B(S_i)$. The sentence S_i may describe zero, one, or more events in the bucket $B(S_i)$. Our objective is to find events in $B(S_i)$ that correspond to the sentence S_i. Figure 1 shows three example sentences and a portion of corresponding buckets in the dataset. Due to very loose alignments between sentences

and events one has to consider a rather big bucket size.

Every event $e(\vec{x})$ is represented by an event type e and a list of arguments $\vec{x}$. The arguments of an event can take two different types: string and categorical. String arguments correspond to lexical properties, such as player names, that can be easily found through a lexical analysis. Often only a part of the argument appears in the text. For instance, "Steven Gerrard" might be referred as "Gerard" or "Steven". Categorical arguments correspond to discrete values in the domain such as `zone` (`back, left, front, right`). These arguments might be addressed in many different ways and cannot be identified with lexical search. In fact, identical words to the categorical values do not convey any information. For ease of notations, we call each event e_j, discarding all the arguments.

Professional Soccer Commentaries Dataset (PSC) This paper introduces a dataset of professional soccer commentaries aligned with real events that occur in each match. We also provide ground-truth alignments between sentences in the commentaries and events of the game for evaluation purposes only. Our model does not use any domain specific knowledge, but we focus on soccer domain in this paper.

Commentaries: Commentaries are English texts, generated by professional soccer commentators. We collect the professionally generated commentaries for matches in the 2010-2011 season of English Premier League from `Espn.net`.

Soccer Events: The events are all actions that occur around the ball and are labeled by human annotators. We use the F24 soccer data feed collected for the EPL by Opta [Opta, 2012]. The F24 data is a time-coded feed that lists all events within the game with a player, team, event type, minute, and second for each event. Each event has a type together with a series of arguments describing it. The data include different types of events, such as goals, shots, passes (with start/end point), tackles, clearances, cards, free kicks, corners, offsides, substitutions and stoppages. This data is currently used for real-time online visualizations of events, as well as post-analysis for prominent television shows and newspaper articles.

Ground Truth Alignments: For evaluation purposes only, we generate the ground truth labels which denote the correct mapping between sentences in the commentary and events in the F24 Opta feed. Every sentence can be matched with zero, one, or more events in the bucket.

The commentaries, list of events with their time stamp, and the features are publicly available at: `http://vision.ri.cmu.edu/data-sets/psc/psc.html`.

This domain is much more complex than the RoboCup soccer, weather, or Windows instruction datasets used by previous work. The sentences are more complex. The commentator uses different phrases to refer to an identical event type. Most of the arguments in the events are categorical, which cannot be aligned using lexical analysis. For some buckets, there are no events that correspond to the sentence because the sentence is either a general statement about the game or is about game statistics, weather, etc. For many buckets, multiple events correspond to the sentence. Figure 3 shows examples of sentences with 2, 3, or 4 corresponding events that our approach finds correctly.

3 Approach

The problem of semantic understanding of commentaries can be formulated as establishing alignments between a sentence and a set of events. This, in fact, reduces to figuring out which events in the bucket $B(S_i)$ correspond to the sentence S_i. Every sentence can be aligned with multiple events in the corresponding bucket. Providing supervision at the level of sentence-event alignments is tedious and expensive. Instead, we use the weak supervision in the rough alignments between sentences and events.

To set up notations, assume that the input to our system includes s sentences represented by $S_i, i \in \{1 \dots s\}$, and each sentence S_i corresponds to a bucket of events $B(S_i)$. Without loss of generality, we can assume that each bucket has n events: $B(S_i) = \{e_1^i, e_2^i, ..., e_n^i\}$. Each sentence S_i can be aligned with a group of events, meaning that each sentence S_i can be matched to a member of the power set of $B(S_i)$, called $\mathcal{P}(B(S_i))$. The cardinality of this set is $2^{|B(S_i)|}$. The problem of finding a group of events that best aligns to the sentence S_i can be formalized as:

$$\arg \max_{\mathcal{E}_i \in \mathcal{P}(B(S_i))} \rho(S_i, \mathcal{E}_i) \quad | \mathcal{E}_i | \leq k \tag{1}$$

where ρ is a ranking function that scores *pairs* of sentences and events based on the quality of the correspondence and $\mathcal{E}$ is a group of events (called macro-event) of a cardinality less than or equal to k. This is obviously a search in an exponential space. Later, we show that Equation 1 is a form of budgeted submodular maximizations. We also show how we can search this space more efficiently.

Before going into the details of our search method, we need to specify a ranking function, ρ. The main role of ρ is to score the quality of each pair; each pair includes a sentence and an event in the corresponding bucket. The core idea is that under an "appropriate" notion of similarity, a good pair tends to appear more consistently across the data. For that, one needs to somehow count the number of appearances of a pair. Simple counting does not work because exact sentences and events may not appear more than once. However, the underlying pattern of correspondence between a sentence and an event may appear rather frequently. Therefore, we need a way of capturing the patterns of correspondences for pairs and then score them based on the popularity of the patterns in the dataset.

3.1 Learning Pair Models

We adopt a discriminative approach that learns the characteristic correspondence patterns that distinguishes a pair from all other pairs. Recently in computer vision, Exemplar Support Vector Machines (ESVM) has shown great success in learning what is unique about an image that can distinguish it from all other images [Malisiewicz *et al.*, 2011; Shrivastava *et al.*, 2011]. The main idea is very simple, yet surprisingly effective. To learn what is unique about each example, one can fit an SVM with only one positive instance and large number of negative instances. The main intuition is that an example can be defined as what it is not, rather than what it is like. Despite being susceptible to overfitting, the proposed hard negative mining method gets away from this issue.

This framework suits our problem setting very well because we learn for all pairs of sentences and events in each bucket and do not need training labels for which pairs are correct. We extend this approach to learn models of pairs (called *PairModel*). A *PairModel* demonstrates how to weigh features of a pair against each other in a discriminative manner. If a learned model for a pair produces a positive score when applied to another pair, then two pairs share analogous patterns of correspondences.

Feature Vector: The features for each pair $p_{ij} = (S_i, e^i_j)$ of a sentence S_i and event e^i_j are $\vec{\Phi}_{ij} = (\vec{\Phi}_{S_i}, \vec{\Phi}_{e^i_j}, \vec{\Phi}_{st_i})$. $\vec{\Phi}_{S_i}$ is a binary vector representing the sentence S_i, where each element in the vector shows the presence of a word in the vocabulary. The vocabulary consists of frequent words in the domain except the words that can occur in the string arguments. For instance, the vocabulary does not include the `player names`. $\vec{\Phi}_{e^i_j}$ is a binary vector representation of the event e^i_j that includes the event type together with its arguments. Each element in the vector represents the presence of the corresponding event type or the argument value in e^i_j. $\vec{\Phi}_{st_i}$ is a binary vector with one element for every string type in the event arguments. Every element in the vector st_i denotes if the string argument is matched with a word in the sentence. For instance, the feature vector has an argument to demonstrate whether or not the `player name` (argument of the event e^i_j) has occurred in the sentence S_i.

Pair Model: For each pair $p_{ij} = (S_i, e^i_j)$, we fit a linear SVM to a set of pairs with $p_{ij} = (S_i, e^i_j)$ as a positive training example and a large number of pairs as negative examples. The negative examples are selected in a way that encourages weak similarities between patterns of correspondences compared to the positive example. To do that, we try to make sure that none of the examples in the negative set contains the identical sentence or event to the positive example. If the events (resp. sentences) are similar (in Euclidean distance) to the event (resp. sentence) of the positive example, we make sure that sentences (resp. events) are very different (more details in Algorithm 1 and Section 4.1). We balance the examples by weighting them accordingly. The algorithm is sketched in Algorithm 1.

The output of the *PairModel* M_{ij} learned for the pair $p_{ij} = (S_i, e^i_j)$ is a weight vector $\vec{\Theta}_{ij}$. The confidence of applying M_{ij} over a new pair $p_{kl} = (S_k, e^k_l)$ is computed as *Conf*$(M_{ij}, p_{kl}) = \vec{\Theta}_{ij} \cdot \vec{\Phi}_{kl}$. This confidence compares the patterns of correspondence between S_i and e^i_j and patterns of correspondence between S_k and e^k_l. The *PairModel* tries to weight the features that are most important to discriminate the corresponding pair from the rest. High confidence means that those important features are "on" in an example; therefore, it is following the same pattern. Figure 2 demonstrates three *PairModels*, the top-weight words for each pair, and the nearest sentence under the learned patterns of correspondences. Non-discriminative measures of similarity like Cosine or Euclidean are not desirable because they treats all the dimensions in the same way. For comparisons and experimental evaluations please see Section 4.

A *PairModel* M_{ij} *likes* the pair p_{kl} by *Conf*(M_{ij}, p_{kl}). We score the similarity between two pairs by looking at their mutual *likeness* meaning that they both share analogous patterns of correspondence. We can now start reasoning about the popularity of *PairModels* by aggregating all mutual likeness scores. To propagate the mutual likeness information we adopt a strategy similar to Google PageRank.

3.2 Ranking Pairs

The input to this module are all of the pairs and their learned models, and the output is the popularity scores of the pairs relative to each other. A pair $p_{ij} = (S_i, e^i_j)$ is likely to be a correct match if the pattern of correspondence extracted by M_{ij} occurs frequently relative to other pairs; this means that p_{ij} is popular . A pair is frequent if many popular pairs like that pair with high confidence. This resembles similar problems in ranking Web pages.

We adopt an approach similar to the PageRank algorithm [Brin and Page, 1998] utilized by Google to order the importance of webpages. PageRank examines the graph of webpages (called Webgraph) and assigns a high score to a web page if many important pages link to the page.

Here, we are interested in computing relative scores of the pairs. We build a graph of pairs by assigning a node to each pair. Unlike the Webgraph, the edges are undirected and weighted. We assign a weighted edge between a pair p_{ij} and a pair p_{kl}, if they mutually like each other (i.e., *Conf*$(M_{ij}, p_{kl}) > 0$ and *Conf*$(M_{kl}, p_{ij}) > 0$). We call this edge the *popularity* link. The graph also has self loops; i.e., a node can also be connected to itself if *Conf*$(M_{ij}, p_{ij}) > 0$. This self loop encodes how com-

petent each *PairModel* is. The weight of an edge between pairs denotes the degree of confidence that each pair likes the other. Calibrating pairmodels against each other is an issue. For that, we use the rank of each pair among all the other pairs to model the degree of confidence. More formally, the weight of the edge between p_{ij} and p_{kl} is $1/(rank(M_{kl}, p_{ij}).rank(M_{ij}, p_{kl}))$ where $rank(M_{kl}, p_{ij})$ shows the order of p_{ij} among all the pairs p with $\mathit{Conf}(M_{ij}, p) > 0$.

Our approach, *PairRank* (sketched in Algorithm 3), first builds the adjacency matrix of the graph using the edges and their weights. The ranking function $\rho(p_{ij})$ iteratively computes the popularity score of a pair according to Equation 2. At every iteration, $\rho(p_{ij})$ is the expected sum (with probability d) of the score of the adjacent pairs (computed at the previous iteration) and the self confidence value:

$$\rho(p_{ij}) = (1-d)\mathit{Conf}(M_{ij}, p_{ij}) + d \sum_{p_{kl} \in T(p_{ij})} \frac{\rho(p_{kl})}{edge(p_{ij}, p_{kl})} \quad (2)$$

where $edge(p_{ij}, p_{kl}) = rank(M_{ij}, p_{kl}).rank(M_{kl}, p_{ij})$, $T(p_{ij})$ is the set of adjacent nodes to p_{ij}, d is a damping factor, and $\rho(p_{ij})$ is initialized by random values.

At iteration 1, only popularity links with length 1 are considered; $\rho(p_{ij})$ only adds up the scores of the pairs that are directly linked to p_{ij}. In next iterations, longer popularity paths are considered; the effect of indirectly linked pairs to p_{ij} is included in the scores of neighboring pairs. We are not interested in adding the effect of pairs with high distances from p_{ij}. We control the expected length of the popularity paths with a damping factor d.

At the end of each iteration, we divide $\rho(p_{ij})$ by the frequency of the type of the event e^i_j to discount the biases in the dataset. For instance, there are 8,597 events with type `pass` in the dataset. However, only 451 (5.2%) of the `pass` events occur in the ground-truth events. The last step of each iteration is to normalize the scores of all the pairs.

3.3 Searching for Good Correspondences

Now that we learn *PairModels* and rank the pairs based on their popularity, we return to our main goal of aligning sentences with events. If we knew that each sentence only corresponds to one event we could report the $\arg\max$ of the outputs of the *PairRank* scores. However, most of the times sentences correspond to groups of events merged together, called macro-events.

Dealing with macro events requires learning *PairModels* and performing the *PairRank* on pairs with macro-events. To pair a sentence S with a macro-event $\mathcal{E}$ we need to define how to merge events to form macro-events. Assume that we want to merge (S, e_1) and (S, e_2) to form the pair $(S, \mathcal{E})$. A *PairModel* for $(S, \mathcal{E})$ should learn the correspondences between the sentence S and all the events in $\mathcal{E}$. To learn the *PairModel* for $(S, \mathcal{E})$, we use (S, e_1) and (S, e_2) as positive examples and generate negative examples as before (Algorithm 1). This results in a *PairModel* M. The confidence of the *PairModel* M over a pair $\mathrm{P}_k = (S_k, \acute{\mathcal{E}})$ is the maximum confidence of the M on pairs of the sentence with every event in the macro event $\acute{\mathcal{E}}$ (Algorithm 2). $Conf(M, \mathrm{P}_k) = \max_{e^k_j} Conf(M, p_{kj} = (s_k, e^k_j))$.

To score the popularity of every pair with macro-event, we run the *PairRank* method by adding one node per each pair with macro-event. The edge weights to the new node are computed as the maximum score of all the events in the macro-event (Algorithm 2).

As mentioned before, the search space for each sentence is the exponential space of all possible ways of merging events in each bucket (Equation 1). However, the good news is that our main objective function in Equation 1 is submodular. Because by definition, the ranking function works as a maximization of a set function over the examples that form the macro-event. This results in $\rho(S_i, \mathcal{E}_j) + \rho(S_i, e_k) \geq \rho(S_i, (\mathcal{E}_j \oplus e_k)) + \rho(S_i, (\mathcal{E}_j \ominus e_k))$. We denote the operation of merging two events by $\oplus$ and the inverse operation by $\ominus$. These operators resemble the union and intersections over sets. Now we can adopt a greedy approximation of Equation 1 with reasonable error bounds [Goundan and Schulz, 2009; Dey *et al.*, 2012; Krause *et al.*, 2008]. The core intuition is that instead of searching the exponential space of all possible ways of merging events, we start with the best scoring event and merge events that maximizes the marginal benefits of merging them. We keep merging until we observe no benefit of doing so. More formally, our recursive greedy solution is:

$$\mathcal{E}^l = \mathcal{E}^{l-1} \oplus \mathcal{A}^*$$
$$\mathcal{A}^* = \arg \max_{A \in B(S) \setminus D} \rho(S, \mathcal{E}^l \oplus A) - \rho(S, \mathcal{E}^l) \quad (3)$$

where $\mathcal{E}^0 = \emptyset$, $\mathcal{E}^l$ is the macro-event, of cardinality at most l, that we want to grow using the merging procedure $\oplus$, and D is the set of all elements in $\mathcal{E}^l$. We stop this procedure after k steps or when merging events does not help (Algorithm 4).

To elaborate more, assume that for the sentence S we are given the bucket of events $B(S) = \{e_1, e_2, e_3\}$. We start with the best scoring event in $B(S)$, let's say e_1. We then look for the event that maximizes the marginal benefit (Equation 3), let's say e_2. We now check to see if there is any gain in forming macro-events. If there is no gain we stop and report e_1 as the answer. Otherwise, we form the macro-event $e_{12} = e_1 \oplus e_2$ by merging the two events together, ($\mathcal{E}^2 = e_{12}$). We now can go to the next layer and search for the next event that maximizes the marginal benefit, let's say e_3. If adding e_3 helps, we form a new macro-event, $\mathcal{E}^3 = e_{123} = e_{12} \oplus e_3$. This procedure may result in aligning sentences with macro-events of cardinality up to k. In our experiments we set the $k = 4$.

Algorithm 1. *PairModel*$(S, \mathcal{E})$
- Input: pair $(S, \mathcal{E})$

1. $\vec{\Phi} \leftarrow$ feature vector for (S, e_j) $\forall e_j$ in $\mathcal{E}$
2. $Pos \leftarrow \cup_j (S, e_j)$ $\forall e_j$ in $\mathcal{E}$
   // Generate Negative Examples
3. $\forall$ pairs: sort $D_e \leftarrow Dist(e, e_k)$, sort $D_S \leftarrow Dist(S, S_k)$
4. $Neg_1 \leftarrow (S_k, e_l) : S_k \in D_{S1:N}, e_l \in D_{e_{end-N:end}}$
5. $Neg_2 \leftarrow (S_k, e_l) : S_k \in D_{Send-N:end}, e_l \in D_{e_{1:N}}$
6. $Neg \leftarrow Neg_1 \cup Neg_2$
7. return $SVM(Pos, Neg)$ with the weight vector $\vec{\Theta}$

Algorithm 2. *ComputeConf*(p_{ij}, p_{kl})
- Input: pairs $p_{ij} = (S_i, \mathcal{E}_j^i)$ and $p_{kl} = (S_k, \mathcal{E}_l^k)$ with feature vector Φ_{kl}

1. $M_{ij} \leftarrow$ *PairModel*(p_{ij}) (Alg. 1)
2. $\vec{\Theta}_{ij} \leftarrow$ weight vector of M_{ij}
3. *Conf*$(M_{ij}, p_{kl}) \leftarrow \max_{e_l \in \mathcal{E}_l^k} \vec{\Theta}_{ij}.\vec{\Phi}_{kl}$

Algorithm 3. *PairRank*$(pairs)$
- Output: ranks of all the pairs

//Build adjacency matrix
1. for pairs p_{ij}, p_{kl}:
   (a) *ComputeConf*(p_{ij}, p_{kl}) (Alg. 2)
   (b) $\vec{P}_i \leftarrow$ scores of applying M_{ij} on all pairs
   (c) $\vec{P}_k \leftarrow$ scores of applying M_{kl} on all pairs
   (d) $edge(p_{ij}, p_{kl}) \leftarrow (rank(p_{kl})$ in $\vec{P}_i)$ and $(rank(p_{ij})$ in $\vec{P}_k)$

//Iteration t
2. while $|\vec{\rho}_t - \vec{\rho}_{t-1}| \geq \epsilon$
   (a) for every pair $p_{ij} = (S_i, e_j^i)$
      i. $\rho_t(p_{ij}) \leftarrow$ Equation 2
      ii. $\rho_t(p_{ij}) \leftarrow \frac{\rho_t(p_{ij})}{\text{frequency}(e_j^i.type)}$
      iii. $\rho_t(p_{ij}) \leftarrow \frac{\rho_t(p_{ij})}{sum(\vec{\rho})}$
3. return $\vec{\rho}_t$

Algorithm 4. *ReturnMacroEvent*$(pairs,k)$
- Output: best macro event for each bucket

1. Iteration $l = 0$:
   (a) $\mathcal{E} = \emptyset$
   (b) $\vec{\rho} \leftarrow$ *PairRank*$(pairs)$
   (c) $\mathcal{A}^* \leftarrow$ highest ranked pair in each bucket according to $\vec{\rho}$
2. Iteration l:
   (a) $\mathcal{E} \leftarrow \mathcal{E} \oplus \mathcal{A}^*$
   (b) $\vec{\rho}_1 \leftarrow$ *PairRank*$(pairs \cup \mathcal{A}^*)$
   (c) for e_i in every bucket:
      i. $\vec{\rho}_2 \leftarrow$ *PairRank*$(pairs \cup \mathcal{E} \oplus e_i)$
      ii. $\Delta_{e_i} \leftarrow \rho_2(S, \mathcal{E} \oplus e_i) - \rho_1(S, \mathcal{E})$
   (d) $\mathcal{A}^* \leftarrow \arg\max_{e_i} \vec{\Delta}$
3. repeat until $l \leq k$
4. return $\mathcal{E}$ for each bucket

4 Experiments

We evaluate our method on how accurately it aligns sentences in professional soccer commentaries with events in the actual games. We use our professional soccer dataset as the main testbed and compare our method with state-of-the-art methods and several different baselines. For comparison, we also test our model on a benchmark dataset of RoboCup soccer [Chen and Mooney, 2008].

4.1 Professional Soccer Commentaries

Our dataset consists of time-stamped commentaries and event logs of 8 games in 2010-2011 season of English Premier League. For evaluation purposes, we label ground-truth annotations for the correspondences between sentences and events throughout the dataset. There are 935 sentences, 14,845 events, 2,147 words, and 306 players in total. Each sentence on average has 16.62 words. For each sentence in the commentaries, we assign a bucket by selecting events that occur in an interval of 150 seconds around the time the sentence has been generated. We then pair each sentence with all of the events in the corresponding bucket. This results in 38,332 pairs. On average, there are 42 pairs in each bucket. Of course, not all of these pairs are correct correspondences. There are in total 1,404 pairs labeled as correct matches in the ground-truth labels. On average there are 2 correct pairs in each bucket.

Each event is represented with an event type followed by a list of its arguments. There are 55 event types and several arguments defined in the event logs. Examples of the arguments are the `time` that the event occurred, the `player name` that is the agent of the event, the `team name`, the `outcome` of the event, and the `body part`. Most of the arguments are categorical. The only string field is the `player name` that can provide useful information for finding initial guesses for correct correspondences. However, identical player names occur in multiple events in each bucket.

We apply our method (Algorithm 4) on this dataset. We start by pairing sentences with events in the corresponding buckets. This step is followed by learning *PairModels* for every pair $p_{ij} = (S_i, e_j^i)$ that has at least one matching player name between the sentence S_i and the event e_j^i. The features for the pair p_{ij} are $\Phi_{ij} = (\Phi_{S_i}, \Phi_{e_j^i}, \Phi_{st})$ where Φ_{S_i} and $\Phi_{e_j^i}$ are binary vectors representing the sentence and the event, and Φ_{st} is an integer value corresponding to the number of players matched between the sentence and 3 consecutive events. At the first step, each *PairModel* M_{ij} is trained using one positive example (S_i, e_j^i) and 100 negative examples generated automatically as follows. We sort all sentences according to their Euclidean distance to the sentence S_i and all events based on their Euclidean distance to the event e_j^i. We generate negative examples from the sentences and events that are not identical to S_i and e_j^i. Half of the negative pairs are generated from the sentences with lowest distance to S_i and events with highest distance to e_j^i. The other half are generated by pairing sentences with highest distance to S_i and events with lowest distance to e_j^i. We make sure that no pairs with matching player names between sentences and events exist in the negative set. This way we try to make sure that negative examples do not contain the same patterns of correspondences as the positive examples.

Sentence	Chelsea looking for a penalty as Malouda's header hits Koscielny, not a chance as it hit him in the stomach.	First attack for Drogba, outmuscling Sagna and sending an effort in from the edge of the box which is blocked, and Song then brings down Drogba for a free kick	Two poor efforts for the price of one from the free kick as Nasri's shot hits the wall before Vermaelen fires wide.
Event	**E:**Foul **Q:**Head Pass **T:**Chelsea **P:**Malouda	**E:**Challenge **Q:** Through Ball **T:**Arsenal **P:**Sagna	**E:**Miss **Q:** Pass **T:**Arsenal **P:**Sagna
Top Weights	'penalty', 'looking', 'hits', 'Foul', 'header', 'chance', 'chelsea', 'city', 'clear','half'	'box', 'first','edge','block', 'attack','effort', 'kick', 'free','wide', break	'shot', 'wide', 'two', 'hits', 'poor','one' 'free', 'kick', 'Miss', 'chelsea'
Top Sentences	Alex Song is too strong for Malouda, who goes down looking for a foul. Nothing given by the referee.	Foul by Arshavin as he blocks Essien's attack. He's lucky to escape a card there.	A choppy start from both sides with possession swapping hands regularly. Wilshere gets stuck into Jovanovic and gives away a free kick.

Figure 2: Qualitative analysis of *PairModels*: Each column corresponds to a *PairModel* trained with a sentence on the first row and the event in the second row as the positive example. The learned *PairModel* assigns high values to the features corresponding to the words in the third row. The closest sentence under the learned pattern of correspondences between the sentence and the event in the pair is shown in the fourth row.

We then fit linear SVMs [Fan *et al.*, 2008] to the positive and negative examples for each pair. We use LibLinear with $c = 100$. We weight positive instances to avoid the affects of unbalanced data. We then apply the *PairRank* with the damping factor $d = 0.5$ to rank the pairs based on their consistency. To create macro-events we run Algorithm 4 with $k = 4$.

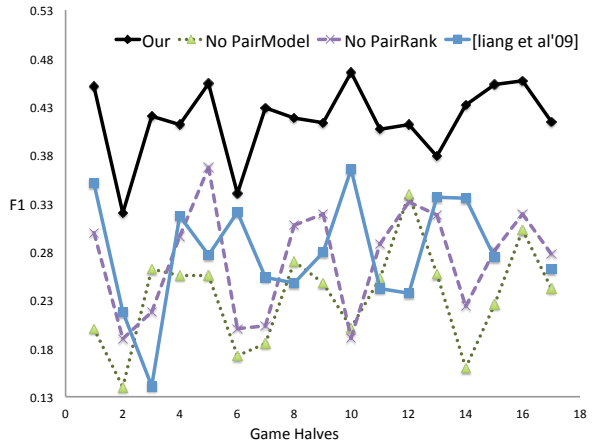


Figure 4: Our method outperforms the state-of-the-art on all the games.

Method	F_1	AUC	Precision	Recall
No PairModel	23.3	36.4	37.3	17.1
No PairRank	27.3	37.4	**39.4**	21.1
MIL	11.0	36.8	37.3	7.0
[Liang *et al.*, 2009]	27.6	N/A	27.2	28.4
Our approach	**41.4**	**46.8**	33.9	**54.0**

Table 1: Average performance of different approaches over all games in our dataset.

4.1.1 Comparisons

We compare the performance of our method with the state-of-the-art method of [Liang *et al.*, 2009], a Multiple Instance Learning (MIL) method of [Andrews *et al.*, 2002], and two baselines.As the main testbed, we use our professional soccer dataset where we have 16 half games (8 games). Figure 4 plots the comparisons mentioned above on all 16 half games. We use F_1 as the measure of performance per half game. Table 1 contains the results of all the aforementioned methods using F_1 and AUC measures averaged over all half games.

[Liang *et al.*, 2009] This approach uses a generative model that learns the correspondence between sentences and events. We use their publicly available code and run their method for 5 iterations. To have a generous comparison, we report the best results achieved during 5 iterations (sometimes the best performance is achieved earlier than 5 iterations). Table 1 shows the average accuracy of this method over all the games. Our model outperforms this method by more than 14% in F_1. Due to the complexity of the model, [Liang *et al.*, 2009] cannot take advantage of the full capacity of the domain. It runs out of memory on a machine with $8GB$ of memory when using 6 arguments. The reason is that this approach grows exponentially with the number of arguments whereas our approach grows linearly. To be compatible, we decrease the number of arguments of every event to 3 arguments `team, player-name, qualifier`, in all the comparisons. Even after decreasing the number of arguments, the method of [Liang *et al.*, 2009] still runs out of memory for one game that consist of long sentences (half-game 16). Due to memory limitations, this method is not applicable to all games at the same time; we perform all comparisons on half game basis.

Multiple Instance Learning (MIL): Finding correct correspondences given the rough alignments between sentences and events can be formulated as a multiple instance learning problem. Pairs of sentences with all the events in the corresponding bucket can be considered as bags. A positive bag includes at least one correct pair. Negative bags do not include any correct pairs. We utilize the same procedure that we used to generate negative pairs to produce neg-

Sentences	Discovered Events			
1: Chelsea looking for penalty as Malouda's header hits Koscielny, not a chance as it hit him in the stomach.	**E:** Pass	**Q:** Head Pass	**T:** Chelsea	**P:** F. Malouda
	E: Ball touch	**Q:** Through ball	**T:** Arsenal	**P:** L. Koscielny
	E: Foul	**Q:** through ball	**T:** Arsenal	**P:** L. Koscielny
	E: Foul	**Q:** Head Pass	**T:** Chelsea	**P:** F. Malouda
2: GOAL, Drogba opens the scoring! Ramires finds Ashley Cole with a perfectly waited pass, Cole's low cross finds Drogba at the near post who back-heels it beyond a stranded Fabianski. 1-0 to the champions.	**E:** Pass	**Q:** Cross	**T:** Chelsea	**P:** A. Cole
	E: Goal	**Q:** Head Pass	**T:** Chelsea	**P:** D. Drogba
3: Essien dispossesses Arshavin and earns Chelsea's fourth corner of the match	**E:** Intercept	**Q:** Cross	**T:** Chelsea	**P:** M. Essien
	E: Pass	**Q:** Pass	**T:** Chelsea	**P:** M. Essien
	E: Corner Awarded	**Q:** Pass	**T:** Chelsea	**P:** M. Essien
4: Cole is sent off for a lunge on Koscielny, it was poor, it was late but I'm not entirely sure that should have been red.	**E:** Foul	**Q:** Long Ball	**T:** Liverpool	**P:** J. Cole
	E: Foul	**Q:** Through Ball	**T:** Arsenal	**P:** L. Koscielny
	E: Card	**Q:** None	**T:** Liverpool	**P:** J. Cole
5: First attack for Drogba, outmuscling Sagna and sending an effort in from the edge of the box which is blocked, and Song then brings down Drogba for a free kick.	**E:** Take On	**Q:** Head Pass	**T:** Chelsea	**P:** D. Drogba
	E: Challenge	**Q:** Through Ball	**T:** Arsenal	**P:** B. Sagna
	E: Save	**Q:** Through Ball	**T:** Arsenal	**P:** A. Song
	E: Foul	**Q:** Through Ball	**T:** Arsenal	**P:** A. Song
6: Two poor efforts for the price of one from the free kick as Nasri's shot hits the wall before Vermaelen fires wide.	**E:** Attempt Saved	**Q:** Head Pass	**T:** Arsenal	**P:** S. Nasri
	E: Miss	**Q:** Head Pass	**T:** Arsenal	**P:** T. Vermaelen

Figure 3: Qualitative exampels of macro-events discovered by our method in correspondence to sentences. For instance, in sentence 4 the commentator is implicitly describing a foul occurred in the game which led to a card but there is no explicit mention of 'Foul' nor 'Card' in the sentence. However, our method can discover both events correctly.

ative bags. We use the publicly available implementation of mi-SVM [Andrews *et al.*, 2002] for comparisons. Table 1 compares the performance of our method with that of MIL. Our method outperforms MIL significantly (by about 30% in F_1). We postulate that, the complex structure in the correspondences between sentences and events cannot be discovered by latent methods like mi-SVM.

No PairModel Baseline: To analyze the importance of discriminative notion of correspondences, we also compare our method with a baseline that uses a non-discriminative notion of correspondences between sentences and events. In our model, we replace the *PairModel* with a non-discriminative similarity metric (such as Cosine or Euclidean). In our experiments, we did not find any considerable difference between the non-discriminative distances. Table 1 shows performance numbers using Euclidean distance. Replacing the *PairModel* with non-discriminative distances decreases the performance by 16%.

No PairRank Baseline: We also replace the PairRank component in our model with a voting scheme. In this baseline, we compute the confidence scores of applying all *PairModels* on all pairs. We then compute the score of each pair by counting the number of non-negative confidence values. Replacing the *PairRank* model with a voting scheme decreases the performance by 13%.

We also analyze the importance of incorporating macro-events. To this purpose, we enforce our model to produce macro-events at multiple different lengths. Figure 5 plots the F_1 measures against the maximum cardinality of macro events. Forming macro-events up to the cardinality 4 dramatically boosts the performance. Our experiments do not show any boost by adding longer macro-events.

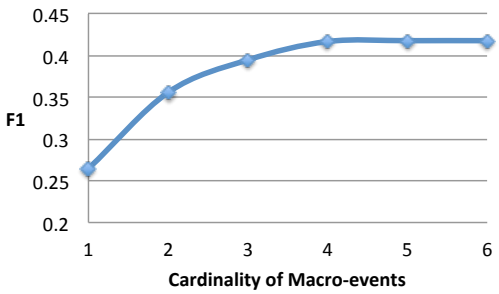


Figure 5: Average F_1 over all games by increasing the number of iterations.

4.1.2 Qualitative Examples

Figure 3 shows examples of macro-events discovered by our method in correspondence to the sentences in the first column. The correspondences are not obvious and show deep understanding of sentences in commentaries. To further analyze the intermediate result of our model, we looked at the top scoring dimensions in the features for specific events. For example for event `Foul`, the highest weights correspond to 'Foul','free','kick','card', 'dangerous', and 'late'. For the event `Save`, the highest dimensions corresponds to 'Save','goal', 'shot','block','ball'.

4.2 RoboCup Soccer Commentary

We also compare our approach with the state-of-the-art methods in the RoboCup soccer dataset [Chen and Mooney, 2008]. The data is based on commentaries of four championship games of the RoboCup simulation league. Each game is associated with a sequence of human comments in English and the Robocup soccer events that happen in the original game tagged with time. Sentences are on average 5.7 words long. There are 17 events, including actions with the ball or other game information. The buckets are generated by mapping sentences and the events that occurred within 5 time steps of when the comments were recorded. There are in total 1,919 pairs and average of 2.4 events per bucket. It is assumed that every sentence is matched with at most one event. Following previous approach, we adopt the scheme of 4 fold cross validation and report the micro-averaged results for four games in terms of F_1. Table 2 demonstrates that our method outperforms the state-of-the-art. [Hajishirzi *et al.*, 2011] uses domain knowledge about soccer events instead of buckets information. [Chen *et al.*, 2010] can achieve the F_1 of 79.1% by initializing with the output generated by [Liang *et al.*, 2009]. We cannot directly compare [Bordes *et al.*, 2010] to other methods in the table because a) they use Bigram and Trigram features. This gives a strong boost for games like the game 3 where most sentences are 3 or 4 words long. b) they use post processing heuristics to find sentences with no matching events. With the same heuristics our model can go up to an F1 of 84.57 (cf. 83.0 of [Bordes *et al.*, 2010]).

Approach	F_1
[Chen and Mooney, 2008]	67.0
[Chen *et al.*, 2010]	73.5
[Liang *et al.*, 2009]	75.7
[Hajishirzi *et al.*, 2011]	77.9
Our approach	**81.6**

Table 2: The average F_1 scores of 4 fold cross validation in the Roboup dataset.

5 Discussion and Future Work

In this paper, we present an approach that can form macro-events that best describe sentences given a bucket of events that correspond to each sentence. Our method takes advantage of a discriminative notion of correspondence coupled with a ranking technique to find popular pairs in our professional soccer dataset. To avoid exponential searches over all possible ways of merging events in a bucket we use a greedy approximation with reasonable bounds. We update the macro-events incrementally by combining events that provide maximum marginal benefits. Our experiments show significant improvement over the state-of-the-art methods. In fact, our method achieves an F_1 measure of 41.4% comparing to the state of the art performance of 27.6% in professional soccer commentaries. Also, our method outperforms state-of-the-art on RoboCup dataset.

Our method can assign a macro-event to a part of the text that cannot be further segmented. For instance, the first part of the sentence 1 in Figure 3 is mapped to both pass and corner. We argue that we need to first align sentences with macro-events and then start the segmentation with a relaxed one-to-one assumption. Discovering strategy-level macro events still remains open. For example our method cannot align attack or coming forward with series of passes. We believe that we can make progress by augmenting our method with more powerful Natural Language Processing tools that improves lexical analysis. Also, our method cannot address the cases where there is no event corresponding to a sentence. Our formulation forces at least one event per sentence. This is not desirable for many domains including soccer commentaries where there are several sentences that do not correspond to any ball-related event in the game. Our ultimate goal is to build an automatic commentary generation for professional soccer games. This paper is one step forward toward that big goal.

Acknowledgments

The authors would like to thank Opta for providing the data on event logs of the soccer games, Emma Brunskill for the useful discussions, and anonymous reviewers for their insightful comments on the paper. Ali Farhadi is supported by the ONR-MURI grant N000141010934.

References

[Andrews *et al.*, 2002] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *NIPS*, pages 561–568, 2002.

[Bordes *et al.*, 2010] A. Bordes, N. Usunier, and J. Weston. Label ranking under ambiguous supervision for learning semantic correspondences. In *ICML*, pages 103–110, 2010.

[Branavan *et al.*, 2009] S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *ACL/AFNLP*, pages 82–90, 2009.

[Branavan *et al.*, 2010] S. R. K. Branavan, Luke S. Zettlemoyer, and Regina Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *ACL*, pages 1268–1277, 2010.

[Brin and Page, 1998] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[Chen and Mooney, 2008] David L. Chen and Raymond J. Mooney. Learning to sportscast: a test of grounded language acquisition. In *ICML*, pages 128–135, 2008.

[Chen *et al.*, 2010] David L. Chen, Joohyun Kim, and Raymond J. Mooney. Training a multilingual sportscaster: Using perceptual context to learn language. *J. Artif. Intell. Res. (JAIR)*, 37:397–435, 2010.

[Dey *et al.*, 2012] Debadeepta Dey, Tian Yu Liu, Martial Hebert, and J. Andrew Bagnell. Predicting contextual sequences via submodular function maximization. *CoRR*, abs/1202.2112, 2012.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[Ge and Mooney, 2006] Ruifang Ge and Raymond J. Mooney. Discriminative reranking for semantic parsing. In *ACL*, 2006.

[Goundan and Schulz, 2009] P.R. Goundan and A.S. Schulz. Revisiting the greedy approach to submodular set function maximization. *MIT*, 2009.

[Hajishirzi *et al.*, 2011] Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. Reasoning about robocup soccer narratives. In *UAI*, pages 291–300, 2011.

[Kate and Mooney, 2007] Rohit J. Kate and Raymond J. Mooney. Learning language semantics from ambiguous supervision. In *AAAI'07*, pages 895–900, 2007.

[Krause *et al.*, 2008] Andreas Krause, Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *Journal of Machine Learning Research (JMLR)*, 9:2761–2801, December 2008.

[Liang *et al.*, 2009] Percy Liang, Michael I. Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *ACL/AFNLP*, pages 91–99, 2009.

[Malisiewicz *et al.*, 2011] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[Opta, 2012] Opta. http://www.optasports.com, 2012.

[Poon and Domingos, 2009] Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In *EMNLP*, pages 1–10, 2009.

[Shrivastava *et al.*, 2011] Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Transaction of Graphics (TOG) (Proceedings of ACM SIGGRAPH ASIA)*, 30(6), 2011.

[Snyder and Barzilay, 2007] Benjamin Snyder and Regina Barzilay. Database-text alignment via structured multilabel classification. In *IJCAI*, pages 1713–1718, 2007.

[Vogel and Jurafsky, 2010] Adam Vogel and Daniel Jurafsky. Learning to follow navigational directions. In *ACL*, pages 806–814, 2010.

[Zettlemoyer and Collins, 2005] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666, 2005.

Weighted Sets of Probabilities and Minimax Weighted Expected Regret: New Approaches for Representing Uncertainty and Making Decisions*

Joseph Y. Halpern
Cornell University
halpern@cs.cornell.edu

Samantha Leung
Cornell University
samlyy@cs.cornell.edu

Abstract

We consider a setting where an agent's uncertainty is represented by a set of probability measures, rather than a single measure. Measure-by-measure updating of such a set of measures upon acquiring new information is well-known to suffer from problems; agents are not always able to learn appropriately. To deal with these problems, we propose using *weighted sets of probabilities*: a representation where each measure is associated with a *weight*, which denotes its significance. We describe a natural approach to updating in such a situation and a natural approach to determining the weights. We then show how this representation can be used in decision-making, by modifying a standard approach to decision making—minimizing expected regret—to obtain *minimax weighted expected regret* (MWER). We provide an axiomatization that characterizes preferences induced by MWER both in the static and dynamic case.

1 Introduction

Agents must constantly make decisions; these decisions are typically made in a setting with uncertainty. For decisions based on the outcome of the toss of a fair coin, the uncertainty can be well characterized by probability. However, what is the probability of you getting cancer if you eat fries at every meal? What if you have salads instead? Even experts would not agree on a single probability.

Representing uncertainty by a single probability measure and making decisions by maximizing expected utility leads to further problems. Consider the following stylized problem, which serves as a running example in this paper. The baker's delivery robot, T-800, is delivering $1,000$ cupcakes from the bakery to a banquet. Along the way, T-800 takes a tumble down a flight of stairs and breaks some of the cupcakes. The robot's map indicates that this flight of stairs must be either ten feet or fifteen feet high. For simplicity, assume that a fall of ten feet results in one broken cupcake, while a fall of fifteen feet results in ten broken cupcakes.

	1 broken	10 broken
$cont$	10,000	-10,000
$back$	0	0
$check$	5,001	-4,999

Table 1: Payoffs for the robot delivery problem. Acts are in the leftmost column. The remaining two columns describe the outcome for the two sets of states that matter.

T-800's choices and their consequences are summarized in Table 1. Decision theorists typically model decision problems with states, acts, and outcomes: the world is in one of many possible states, and the decision maker chooses an *act*, a function mapping states to outcomes. A natural state space in this problem is $\{$*good,broken*$\}^{1000}$, where each state is a possible state of the cupcakes. However, all that matters about the state is the number of broken cakes, so we can further restrict to states with either one or ten broken cakes.

T-800 can choose among three acts: $cont$: continue the delivery attempt; $back$: go back for new cupcakes; or $check$: open the container and count the number of broken cupcakes, and then decide to continue or go back, depending on the number of broken cakes. The client will tolerate one broken cupcake, but not ten broken cupcakes. Therefore, if T-800 chooses $cont$, it obtains a utility of $10,000$ if there is only one broken cake, but a utility of $-10,000$ if there are ten broken cakes. If T-800 chooses to go $back$, then it gets a utility of 0. Finally, checking the cupcakes costs $4,999$ units of utility but is reliable, so if T-800 chooses $check$, it ends up with a utility of $5,001$ if there is one broken cake, and a utility of $-4,999$ if there are ten broken cakes.

*Work supported in part by NSF grants IIS-0534064, IIS-0812045, and IIS-0911036, by AFOSR grants FA9550-08-1-0438 and FA9550-09-1-0266, and by ARO grant W911NF-09-1-0281.

If we try to maximize expected utility, we must assume some probability over states. What measure should be used? There are two hypotheses that T-800 entertains: (1) the stairs are ten feet high and (2) the stairs are fifteen feet high. Each of these places a different probability on states. If the stairs are ten feet high, we can take all of the $1,000$ states where there is exactly one broken cake to be equally probable, and take the remaining states to have probability 0; if the stairs are fifteen feet high, we can take all of the $C(1000, 10)$ states where there are exactly ten broken cakes to be equally probable, and take the remaining states to have probability 0. One way to model T-800's uncertainty about the height of the stairs is to take each hypothesis to be equally likely. However, not having any idea about which hypothesis holds is very different from believing that all hypotheses are equally likely. It is easy to check that taking each hypothesis to be equally likely makes $check$ the act that maximizes utility, but taking the probability that the stairs are fifteen feet high to be .51 makes $back$ the act that maximizes expected utility, and taking the probability that the stairs are ten feet high to be .51 makes $cont$ the act that maximizes expected utility. What makes any of these choices the "right" choice?

It is easy to construct many other examples where a single probability measure does not capture uncertainty, and does not result in what seem to be reasonable decisions, when combined with expected utility maximization. A natural alternative, which has often been considered in the literature, is to represent the agent's uncertainty by a *set* of probability measures. For example, in the delivery problem, the agent's beliefs could be represented by two probability measures, $\Pr_1$ and $\Pr_{10}$, one for each hypothesis. Thus, $\Pr_1$ assigns uniform probability to all states with exactly one broken cake, and $\Pr_{10}$ assigns uniform probability to all states with exactly ten broken cakes.

But this representation also has problems. Consider the delivery example again. In a more realistic scenario, why should T-800 be sure that there is exactly either one broken cake or ten broken cakes? Of course, we can replace these two hypotheses by hypotheses that say that the probability of a cake being broken is either .001 or .01, but this doesn't solve the problem. Why should the agent be sure that the probability is either exactly .001 or exactly .01? Couldn't it also be .0999? Representing uncertainty by a set of measures still places a sharp boundary on what measures are considered possible and impossible.

A second problem involves updating beliefs. How should beliefs be updated if they are represented by a set of probability measures? The standard approach for updating a single measure is by conditioning. The natural extension of conditioning to sets of measure is measure-by-measure updating: conditioning each measure on the information (and also removing measures that give the information probability 0).

However, measure-by-measure updating can produce some rather counterintuitive outcomes. In the delivery example, suppose that a passer-by tells T-800 the information E: the first 100 cupcakes are good. Assuming that the passer-by told the truth, intuition tells us that there is now more reason to believe that there is only one broken cupcake.

However, $\Pr_1 \mid E$ places uniform probability on all states where the first 100 cakes are good, and there is exactly one broken cake among the last 900. Similarly, $\Pr_{10} \mid E$ places uniform probability on all states where the first 100 cakes are good, and there are exactly ten broken cakes among the last 900. $\Pr_1 \mid E$ still places probability 1 on there being one broken cake, just like $\Pr_1$, $\Pr_{10} \mid E$ still places probability 1 on there being ten broken cakes. There is no way to capture the fact that T-800 now views the hypothesis $\Pr_{10}$ as less likely, even if the passer-by had said instead that the first 990 cakes are all good!

Of course, both of these problems would be alleviated if we placed a probability on hypotheses, but, as we have already observed, this leads to other problems. In this paper, we propose an intermediate approach: representing uncertainty using *weighted sets of probabilities*. That is, each probability measure is associated with a weight. These weights can be viewed as probabilities; indeed, if the set of probabilities is finite, we can normalize them so that they are effectively probabilities. Moreover, in one important setting, we update them in the same way that we would update probabilities, using likelihood (see below). On the other hand, these weights do not act like probabilities if the set of probabilities is infinite. For example, if we had a countable set of hypotheses, we could assign them all weight 1 (so that, intuitively, they are all viewed as equally likely), but there is no uniform measure on a countable set.

More importantly, when it comes to decision making, we use the weights quite differently from how we would use second-order probabilities on probabilities. Second-order probabilities would let us define a probability on events (by taking expectation) and maximize expected utility, in the usual way. Using the weights, we instead define a novel decision rule, *minimax weighted expected regret (MWER)*, that has some rather nice properties, which we believe will make it widely applicable in practice. If all the weights are 1, then MWER is just the standard *minimax expected regret (MER)* rule (described below). If the set of probabilities is a singleton, then MWER agrees with (subjective) expected utility maximization (SEU). More interestingly perhaps, if the weighted set of measures converges to a single measure (which will happen in one important special case, discussed below), MWER converges to SEU. Thus, the weights give us a smooth, natural way of interpolating between MER and SEU.

In summary, weighted sets of probabilities allow us to represent ambiguity (uncertainty about the correct probability

distribution). Real individuals are sensitive to this ambiguity when making decisions, and the MWER decision rule takes this into account. Updating the weighted sets of probabilities using likelihood allows the initial ambiguity to be resolved as more information about the true distribution is obtained.

We now briefly explain MWER, by first discussing MER. MER is a probabilistic variant of the minimax regret decision rule proposed by Niehans [10] and Savage [13]. Most likely, at some point, we've second-guessed ourselves and thought "had I known this, I would have done that instead". That is, in hindsight, we regret not choosing the act that turned out to be optimal for the realized state, called the *ex post* optimal act. The *regret* of an act a in a state s is the difference (in utility) between the ex post optimal act in s and a. Of course, typically one does not know the true state at the time of decision. Therefore the regret of an act is the worst-case regret, taken over all states. The *minimax regret* rule orders acts by their regret.

The definition of regret applies if there is no probability on states. If an agent's uncertainty is represented by a single probability measure, then we can compute the *expected regret* of an act a: just multiply the regret of an act a at a state s by the probability of s, and then sum. It is well known that the order on acts induced by minimizing expected regret is identical to that induced by maximizing expected utility (see [8] for a proof). If an agent's uncertainty is represented by a set $\mathcal{P}$ of probabilities, then we can compute the expected regret of an act a with respect to each probability measure $\Pr \in \mathcal{P}$, and then take the worst-case expected regret. The MER (Minimax Expected Regret) rule orders acts according to their worst-case expected regret, preferring the act that minimizes the worst-case regret. If the set of measures is the set of *all* probability measures on states, then it is not hard to show that MER induces the same order on acts as (probability-free) minimax regret. Thus, MER generalizes both minimax regret (if $\mathcal{P}$ consists of all measures) and expected utility maximization (if $\mathcal{P}$ consists of a single measure).

MWER further generalizes MER. If we start with a *weighted* set of measures, then we can compute the weighted expected regret for each one (just multiply the expected regret with respect to $\Pr$ by the weight of $\Pr$) and compare acts by their worst-case weighted expected regret.

Sarver [12] also proves a representation theorem that involves putting a multiplicative weight on a regret quantity. However, his representation is fundamentally different from MWER. In his representation, regret is a factor only when comparing two *sets* of acts; the ranking of individual acts is given by expected utility maximization. By way of contrast, we do not compare sets of acts.

It is standard in decision theory to axiomatize a decision rule by means of a representation theorem. For example, Savage [14] showed that if an agent's preferences $\succeq$ satisfied several axioms, such as completeness and transitivity, then the agent is behaving as if she is maximizing expected utility with respect to some utility function and probabilistic belief.

If uncertainty is represented by a set of probability measures, then we can generalize expected utility maximization to *maxmin expected utility (MMEU)*. MMEU compares acts by their worst-case expected utility, taken over all measures. MMEU has been axiomatized by Gilboa and Schmeidler [7]. MER was axiomatized by Hayashi [8] and Stoye [16]. We provide an axiomatization of MWER. We make use of ideas introduced by Stoye [16] in his axiomatization of MER, but the extension seems quite nontrivial.

We also consider a dynamic setting, where beliefs are updated by new information. If observations are generated according to a probability measure that is stable over time, then, as we suggested above, there is a natural way of updating the weights given observations, using ideas of likelihood. The idea is straightforward. After receiving some information E, we update each probability $\Pr \in \mathcal{P}$ to $\Pr \mid E$, and take its weight to be $\alpha_{\Pr} = \Pr(E)/\sup_{\Pr' \in \mathcal{P}} \Pr'(E)$. Thus, the weight of $\Pr$ after observing E is modified by taking into account the likelihood of observing E assuming that $\Pr$ is the true probability. We refer to this method of updating weights as *likelihood updating*.

If observations are generated by a stable measure (e.g., we observe the outcomes of repeated flips of a biased coin) then, as the agent makes more and more observations, the weighted set of probabilities of the agent will, almost surely, look more and more like a single measure. The weight of the measures in $\mathcal{P}$ closest to the measure generating the observations will converge to 1, and the weight of all other measures will converge to 0. This would not be the case if uncertainty were represented by a set of probability measures and we did measure-by-measure updating, as is standard. As we mentioned above, this means that MWER will converge to SEU.

We provide an axiomatization for dynamic MWER with likelihood updating. We remark that a dynamic version of MMEU with measure-by-measure updating has been axiomatized by Jaffray [9], Pires [11], and Siniscalchi [15].

Likelihood updating is somewhat similar in spirit to an updating method implicitly proposed by Epstein and Schneider [5]. They also represented uncertainty by using (unweighted) sets of probability measures. They choose a threshold α with $0 < \alpha < 1$, update by conditioning, and eliminate all measures whose relative likelihood does not exceed the threshold. This approach also has the property that, over time, all that is left in $\mathcal{P}$ are the measures closest to the measure generating the observations; all other measures are eliminated. However, it has the drawback that it introduces a new, somewhat arbitrary, parameter α.

Chateauneuf and Faro [2] also consider weighted sets of probabilities (they model the weights using what they call *confidence functions*), although they impose more constraints on the weights than we do. They then define and provide a representation of a generalization of MMEU using weighted sets of probabilities that parallels our generalization of MER. Chateauneuf and Faro do not discuss the dynamic situation; specifically, they do not consider how weights should be updated in the light of new information.

The rest of this paper is organized as follows. Section 2 introduces the weighted sets of probabilities representation, and Section 3 introduces the MWER decision rule. Axiomatic characterizations of static and dynamic MWER are provided in Sections 4 and 5, respectively. We conclude in Section 6. All proofs can be found in the full paper (http://cs.cornell.edu/home/halpern/papers/mwr.pdf).

2 Weighted Sets of Probabilities

A set $\mathcal{P}^+$ of *weighted probability measures* on a set S consists of pairs $(\Pr, \alpha_{\Pr})$, where $\alpha_{\Pr} \in [0,1]$ and $\Pr$ is a probability measure on S.[1] Let $\mathcal{P} = \{\Pr : \exists \alpha (\Pr, \alpha) \in \mathcal{P}^+\}$. We assume that, for each $\Pr \in \mathcal{P}$, there is exactly one α such that $(\Pr, \alpha) \in \mathcal{P}^+$. We denote this number by $\alpha_{\Pr}$, and view it as the *weight of* $\Pr$. We further assume for convenience that weights have been normalized so that there is at least one measure $\Pr \in \mathcal{P}$ such that $\alpha_{\Pr} = 1$.[2] We remark that, just as we do, Chateaunef and Faro [2] take weights to be in the interval $[0,1]$. They impose additional requirements on the weights. For example, they require that the weight of a convex combination of two probability measures is at least as high as the weight of each one. This does not seem reasonable in our applications. For example, an agent may know that one of two measures is generating his observations, and give them both weight 1, while giving all other distributions weight 0.

As we observed in the introduction, one way of updating weighted sets of probabilities is by using likelihood updating. We use $\mathcal{P}^+ \mid E$ to denote the result of applying likelihood updating to $\mathcal{P}^+$. Define $\overline{\mathcal{P}}^+(E) = \sup\{\alpha_{\Pr} \Pr(E) : \Pr \in \mathcal{P}\}$; if $\overline{\mathcal{P}}^+(E) > 0$, set $\alpha_{\Pr|E} = \sup_{\{\Pr' \in \mathcal{P} : \Pr'|E = \Pr|E\}} \frac{\alpha_{\Pr'} \Pr'(E)}{\overline{\mathcal{P}}^+(E)}$. Note that given a measure $\Pr \in \mathcal{P}$, there may be several distinct measures $\Pr'$ in $\mathcal{P}$ such that $\Pr' \mid E = \Pr \mid E$. Thus, we take the weight of $\Pr \mid E$ to be the sup of the possible candidate values of $\alpha_{\Pr|E}$. By dividing by $\overline{\mathcal{P}}^+(E)$, we guarantee that $\alpha_{\Pr|E} \in [0,1]$, and that there is some measure $\Pr$ such that $\alpha_{\Pr|E} = 1$, as long as there is some pair $(\alpha_{\Pr}, \Pr) \in \mathcal{P}$ such that $\alpha_{\Pr} \Pr(E) = \overline{\mathcal{P}}^+(E)$. If $\overline{\mathcal{P}}^+(E) > 0$, we take $\mathcal{P}^+ \mid E$ to be

$$\{(\Pr \mid E, \alpha_{\Pr|E}) : \Pr \in \mathcal{P}\}.$$

If $\overline{\mathcal{P}}^+(E) = 0$, then $\mathcal{P}^+ \mid E$ is undefined.

In computing $\mathcal{P}^+ \mid E$, we update not just the probability measures in $\mathcal{P}$, but also their weights. The new weight combines the old weight with the likelihood. Clearly, if all measures in $\mathcal{P}$ assign the same probability to the event E, then likelihood updating and measure-by-measure updating coincide. This is not surprising, since such an observation E does not give us information about the relative likelihood of measures. We stress that using likelihood updating is appropriate only if the measure generating the observations is assumed to be stable. For example, if observations of heads and tails are generated by coin tosses, and a coin of possibly different bias is tossed in each round, then likelihood updating would not be appropriate.

It is well known that, when conditioning on a single probability measure, the order that information is acquired is irrelevant; the same observation easily extends to sets of probability measures. As we now show, it can be further extended to weighted sets of probability measures.

Proposition 1. *Likelihood updating is consistent in the sense that for all* $E_1, E_2 \subseteq S$, $(\mathcal{P}^+ \mid E_1) \mid E_2 = (\mathcal{P}^+ \mid E_2) \mid E_1 = \mathcal{P}^+ \mid (E_1 \cap E_2)$, *provided that* $\mathcal{P}^+ \mid (E_1 \cap E_2)$ *is defined.*

3 MWER

We now define MWER formally. Given a set S of states and a set X of outcomes, an *act* f (over S and X) is a function mapping S to X. For simplicity in this paper, we take S to be finite. Associated with each outcome $x \in X$ is a utility: $u(x)$ is the utility of outcome x. We call a tuple (S, X, u) a *(non-probabilistic) decision problem*. To define regret, we need to assume that we are also given a set $M \subseteq X^S$ of feasible acts, called the *menu*. The reason for the menu is that, as is well known (and we will demonstrate by example shortly), regret can depend on the menu. Moreover, we assume that every menu M has utilities bounded from above. That is, we assume that for all menus M, $\sup_{g \in M} u(g(s))$ is finite. This ensures that the regret of each act is well defined.[3] For a menu M and act

[1] In this paper, for ease of exposition, we take the state space S to be finite, and assume that all sets are measurable. We can easily generalize to arbitrary measure spaces.

[2] While we could take weights to be probabilities, and normalize them so that they sum to 1, if $\mathcal{P}$ is finite, this runs into difficulties if we have an infinite number of measures in $\mathcal{P}$. For example, if we are tossing a coin, and $\mathcal{P}$ includes all probabilities on heads from $1/3$ to $2/3$, using a uniform probability, we would be forced to assign each individual probability measure a weight of 0, which would not work well in the definition of MWER.

[3] Stoye [17] assumes that, for each menu M, there is a finite set A_M of acts such that M consists of all the convex combinations of the acts in A_M. Our assumption is clearly much weaker than Stoye's.

$f \in M$, the regret of f with respect to M and decision problem (S, X, u) in state s is

$$reg_M(f, s) = \left(\max_{g \in M} u(g(s))\right) - u(f(s)).$$

That is, the regret of f in state s (relative to menu M) is the difference between $u(f(s))$ and the highest utility possible in state s (among all the acts in M). The regret of f with respect to M and decision problem (S, X, u) is the worst-case regret over all states:

$$\max_{s \in S} reg_M(f, s).$$

We denote this as $reg_M^{(S,X,u)}(f)$, and usually omit the superscript (S, X, u) if it is clear from context. If there is a probability measure $\Pr$ over the states, then we can consider the *probabilistic decision problem* $(S, X, u, \Pr)$. The *expected regret* of f with respect to M is

$$reg_{M,\Pr}(f) = \sum_{s \in S} \Pr(s) reg_M(f, s).$$

If there is a set $\mathcal{P}$ of probability measures over the states, then we consider the $\mathcal{P}$-decision problem $(S, X, u, \mathcal{P})$. The maximum expected regret of $f \in M$ with respect to M and $(S, X, u, \mathcal{P})$ is

$$reg_{M,\mathcal{P}}(f) = \sup_{\Pr \in \mathcal{P}} \left(\sum_{s \in S} \Pr(s) reg_M(f, s)\right).$$

Finally, if beliefs are modeled by weighted probabilities $\mathcal{P}^+$, then we consider the $\mathcal{P}^+$-decision problem $(S, X, u, \mathcal{P}^+)$. The maximum weighted expected regret of $f \in M$ with respect to M and $(S, X, u, \mathcal{P}^+)$ is

$$reg_{M,\mathcal{P}^+}(f) = \sup_{\Pr \in \mathcal{P}} \left(\alpha_{\Pr} \textstyle\sum_{s \in S} \Pr(s) reg_M(f, s)\right).$$

The MER decision rule is thus defined for all $f, g \in X^S$ as

$$f \succeq_{M,\mathcal{P}}^{S,X,u} g \text{ iff } reg_{M,\mathcal{P}}^{(S,X,u)}(f) \leq reg_{M,\mathcal{P}}^{(S,X,u)}(g).$$

That is, f is preferred to g if the maximum expected regret of f is less than that of g. We can similarly define $\succeq_{M,reg}$, $\succeq_{M,\Pr}^{S,X,u}$, and $\succeq_{M,\mathcal{P}^+}^{S,X,u}$ by replacing $reg_{M,\mathcal{P}}^{(S,X,u)}$ by $reg_M^{(S,X,u)}$, $reg_{M,\Pr}^{(S,X,u)}$, and $reg_{M,\mathcal{P}^+}^{(S,X,u)}$, respectively. Again, we usually omit the superscript (S, X, u) and subscript $\Pr$ or $\mathcal{P}^+$, and just write $\succeq_M$, if it is clear from context.

To see how these definitions work, consider the delivery example from the introduction. There are $1,000$ states with one broken cake, and $C(1000, 10)$ states with ten broken cakes. The regret of each action in a state depends only on the number of broken cakes, and is given in Table 2. It is easy to see that the action that minimizes regret is *check*, with *cont* and *back* having equal regret. If we represent uncertainty using the two probability measures $\Pr_1$ and $\Pr_{10}$,

	1 broken cake		10 broken cakes	
	Payoff	Regret	Payoff	Regret
cont	10,000	0	-10,000	10,000
back	0	10,000	0	0
check	5,001	4,999	-4,999	4,999

Table 2: Payoffs and regrets for delivery example.

	1 broken cake		10 broken cakes	
	Payoff	Regret	Payoff	Regret
cont	10,000	10,000	-10,000	10,000
back	0	20,000	0	0
check	5,001	14,999	-4,999	4,999
new	20,000	0	-20,000	20,000

Table 3: Payoffs and regrets for the delivery problem with a new choice added.

the expected regret of each of the acts with respect to $\Pr_1$ (resp., $\Pr_{10}$) is just its regret with respect to states with one (resp. ten) broken cakes. Thus, the action that minimizes maximum expected regret is again *check*.

As we said above, the ranking of acts based on MER or MWER can change if the menu of possible choices changes. For example, suppose that we introduce a new choice in the delivery problem, whose gains and losses are twice those of *cont*, resulting in the payoffs and regrets described in Table 3. In this new setting, *cont* has a lower maximum expected regret $(10,000)$ than *check* $(14,999)$, so MER prefers *cont* over *check*. Thus, the introduction of a new choice can affect the relative order of acts according to MER (and MWER), even though other acts are preferred to the new choice. By way of contrast, the decision rules MMEU and SEU are *menu-independent*; the relative order of acts according to MMEU and SEU is not affected by the addition of new acts.

We next consider a dynamic situation, where the agent acquires information. Specifically, in the context of the delivery problem, suppose that T-800 learns E—the first 100 items are good. Initially, suppose that T-800 has no reason to believe that one hypothesis is more likely than the other, so assigns both hypotheses weight 1. Note that $P_1(E) = 0.9$ and $\Pr_{10}(E) = C(900, 10)/C(1000, 10) \approx 0.35$. Thus, $\mathcal{P}^+ \mid E = \{(\Pr_1 \mid E, 1), (\Pr_{10} \mid E, C(900, 10)/(.9C(1000, 10))\}$.

We can also see from this example that MWER interpolates between MER and expected utility maximization. Suppose that a passer-by tells T-800 that the first N cupcakes are good. If $N = 0$, MWER with initial weights 1 is the same as MER. On the other hand, if $N \geq 991$, then the likelihood of $\Pr_{10}$ is 0, and the only measure that has effect is $\Pr_1$, which means minimizing maximum weighted expected regret is just maximizing expected utility with respect to $\Pr_1$. If $0 < N < 991$, then the likelihoods (hence weights)

of $\Pr_1$ and $\Pr_{10}$ are 1 and $\frac{C(1000-N,10)}{C(1000,10)} \times \frac{1000}{1000-N} < ((999-N)/999)^9$. Thus, as N increases, the weight of $\Pr_{10}$ goes to 0, while the weight of $\Pr_1$ stays at 1.

4 An axiomatic characterization of MWER

We now provide a representation theorem for MWER. That is, we provide a collection of properties (i.e., axioms) that hold of MWER such that a preference order on acts that satisfies these properties can be viewed as arising from MWER. To get such an axiomatic characterization, we restrict to what is known in the literature as the *Anscombe-Aumann* (AA) framework [1], where outcomes are restricted to lotteries. This framework is standard in the decision theory literature; axiomatic characterizations of SEU [1], MMEU [7], and MER [8, 16] have already been obtained in the AA framework. We draw on these results to obtain our axiomatization.

Given a set Y (which we view as consisting of *prizes*), a *lottery* over Y is just a probability with finite support on Y. Let $\Delta(Y)$ consist of all finite probabilities over Y. In the AA framework, the set of outcomes has the form $\Delta(Y)$. So now acts are functions from S to $\Delta(Y)$. (Such acts are sometimes called *Anscombe-Aumann acts*.) We can think of a lottery as modeling objective uncertainty, while a probability on states models subjective uncertainty; thus, in the AA framework we have both objective and subjective uncertainty. The technical advantage of considering such a set of outcomes is that we can consider convex combinations of acts. If f and g are acts, define the act $\alpha f+(1-\alpha)g$ to be the act that maps a state s to the lottery $\alpha f(s)+(1-\alpha)g(s)$.

In this setting, we assume that there is a utility function U on prizes in Y. The utility of a lottery l is just the expected utility of the prizes obtained, that is,

$$u(l) = \sum_{\{y \in Y:\, l(y)>0\}} l(y)U(y).$$

This makes sense since $l(y)$ is the probability of getting prize y if lottery l is played. The expected utility of an act f with respect to a probability $\Pr$ is then just $u(f) = \sum_{s \in S} \Pr(s)u(f(s))$, as usual. We also assume that there are at least two prizes y_1 and y_2 in Y, with different utilities $U(y_1)$ and $U(y_2)$.

Given a set Y of prizes, a utility U on prizes, a state space S, and a set $\mathcal{P}^+$ of weighted probabilities on S, we can define a family $\succeq_{M,\mathcal{P}^+}^{S,\Delta(Y),u}$ of preference orders on Anscombe-Aumann acts determined by weighted regret, one per menu M, as discussed above, where u is the utility function on lotteries determined by U. For ease of exposition, we usually write $\succeq_{M,\mathcal{P}^+}^{S,Y,U}$ rather than $\succeq_{M,\mathcal{P}^+}^{S,\Delta(Y),u}$.

We state the axioms in a way that lets us clearly distinguish the axioms for SEU, MMEU, MER, and MWER. The axioms are universally quantified over acts f, g, and h, menus M and M', and $p \in (0,1)$. We assume that $f, g \in M$ when we write $f \succeq_M g$.[4] We use l^* to denote a constant act that maps all states to l.

Axiom 1. *(Transitivity)* $f \succeq_M g \succeq_M h \Rightarrow f \succeq_M h$.

Axiom 2. *(Completeness)* $f \succeq_M g$ *or* $g \succeq_M f$.

Axiom 3. *(Nontriviality)* $f \succ_M g$ *for some acts* f *and* g *and menu* M.

Axiom 4. *(Monotonicity) If* $(f(s))^* \succeq_{\{(f(s))^*,(g(s))^*\}} (g(s))^*$ *for all* $s \in S$, *then* $f \succeq_M g$.

Axiom 5. *(Mixture Continuity) If* $f \succ_M g \succ_M h$, *then there exists* $q, r \in (0,1)$ *such that*

$$\begin{array}{ll} & qf + (1-q)h \succ_{M \cup \{qf+(1-q)h\}} g \\ \textit{and} & g \succ_{M \cup \{rf+(1-r)h\}} rf + (1-r)h. \end{array}$$

Menu-independent versions of Axioms 1–5 are standard. Clearly (menu-independent versions of) Axioms 1, 2, 4, and 5 hold for MMEU, MER, and SEU; Axiom 3 is assumed in all the standard axiomatizations, and is used to get a unique representation.

Axiom 6. *(Ambiguity Aversion)*

$$f \sim_M g \Rightarrow pf + (1-p)g \succeq_{M \cup \{pf+(1-p)g\}} g.$$

Ambiguity Aversion says that the decision maker weakly prefers to hedge her bets. It also holds for MMEU, MER, and SEU, and is assumed in the axiomatizations for MMEU and MER. It is not assumed for the axiomatization of SEU, since it follows from the Independence axiom, discussed next. Independence also holds for MWER, provided that we are careful about the menus involved. Given a menu M and an act h, let $pM+(1-p)h$ be the menu $\{pf+(1-p)h : p \in M\}$.

Axiom 7. *(Independence)*

$$f \succeq_M g \textit{ iff } pf + (1-p)h \succeq_{pM+(1-p)h} pg + (1-p)h.$$

Independence holds in a strong sense for SEU, since we can ignore the menus. The menu-independent version of Independence is easily seen to imply Ambiguity Aversion. Independence does not hold for MMEU.

Although we have menu independence for SEU and MMEU, we do not have it for MER or MWER. The following two axioms are weakened versions of menu independence that do hold for MER and MWER.

[4] Stoye [17] assumed that menus were convex, so that if $f, g \in M$, then so is $pf + (1-p)g$. We do not make this assumption, although our results would still hold if we did (with the axioms slightly modified to ensure that menus are convex). While it may seem reasonable to think that, if f and g are feasible for an agent, then so is $pf + (1-p)g$, this not always the case. For example, it may be difficult for the agent to randomize, or it may be infeasible for the agent to randomize with probability p for some choices of p (e.g., for p irrational).

Axiom 8. *(Menu independence for constant acts) If l^* and $(l')^*$ are constant acts, then $l^* \succeq_M (l')^*$ iff $l^* \succeq_{M'} (l')^*$.*

In light of this axiom, when comparing constant acts, we omit the menu.

An act h is *never strictly optimal relative to* M if, for all states $s \in S$, there is some $f \in M$ such that $(f(s))^* \succeq (h(s))^*$.

Axiom 9. *(Independence of Never Strictly Optimal Alternatives (INA)) If every act in M' is never strictly optimal relative to M, then $f \succeq_M g$ iff $f \succeq_{M \cup M'} g$.*

Axiom 10. *(Boundedness of menus) For every menu M, there exists a lottery $\bar{l} \in \Delta(Y)$ such that for all $f \in M$ and $s \in S$, $(f(s))^* \preceq \bar{l}^*$.*

The boundedness axiom enforces the assumption that we made earlier that every menu has utilities that are bounded from above. Recall that this assumption is necessary for regret to be finite.

Theorem 1. *For all Y, U, S, and $\mathcal{P}^+$, the family of preference orders $\succeq^{S,Y,U}_{M,\mathcal{P}^+}$ satisfies Axioms 1–10. Conversely, if a family of preference orders $\succeq_M$ on the acts in $\Delta(Y)^S$ satisfies Axioms 1–10, then there exist a weighted set $\mathcal{P}^+$ of probabilities on S and a utility U on Y such that $\succeq_M = \succeq^{S,Y,U}_{M,\mathcal{P}^+}$. Moreover, U is unique up to affine transformations, and $\mathcal{P}^+$ can be taken to be* maximal*, in the sense that if $\succeq_M = \succeq^{S,Y,U}_{M,(\mathcal{P}')^+}$, and $(\alpha, \Pr) \in (\mathcal{P}')^+$, then there exists $\alpha' \geq \alpha$ such that $(\alpha', \Pr) \in \mathcal{P}^+$.*

Showing that $\succeq^{S,Y,U}_{M,\mathcal{P}^+}$ satisfies Axioms 1–10 is fairly straightforward; we leave details to the reader. The proof of the converse is quite nontrivial, although it follows the lines of the proof of other representation theorems. We provide an outline of the proof here; details can be found in the full paper (http://cs.cornell.edu/home/halpern/papers/mwr.pdf).

Using standard techniques, we can show that the axioms guarantee the existence of a utility function U on prizes that can be extended to lotteries in the obvious way, so that $l^* \succeq (l')^*$ iff $U(l) \geq U(l')$. We then use techniques of Stoye [17] to show that it suffices to get a representation theorem for a single menu, rather than all menus: the menu consisting of all acts f such that $U(f(s)) \leq 0$ for all states $s \in S$. This allows us to use techniques in the spirit of those used by by Gilboa and Schmeidler [6] to represent (unweighted) MMEU. However, there are technical difficulties that arise from the fact that we do not have a key axiom that is satisfied by MMEU: C-independence (discussed below). The heart of the proof involves dealing with the lack of C-independence; We leave details of the proof to the full paper.

In standard representation theorems, not only is the utility function unique (up to affine transformations, so that we can replace U by $aU + b$, where $a > 0$ and b are constants), but the probability (or set of probabilities) is unique as well. We were not able to find natural conditions on weighted sets of probabilities that guarantee uniqueness. In the case of sets of probabilities, we need to assume that the set is convex and closed to get uniqueness. But there seems to be no natural notion of convexity for a set $\mathcal{P}^+$ of weighted probabilities, and the requirement that $\mathcal{P}^+$ be closed seems unreasonable. For example, if $\mathcal{P}^+$ consists of a single probability measure $\Pr$ with weight $\alpha_{\Pr} = 1$, then there are sequences $\Pr_n \to \Pr$ with $\alpha_{\Pr_n} = 0$, and yet $\alpha_{\Pr} = 1$. To see why something like convexity is needed for uniqueness, consider the delivery example and the expected regrets in Table 2, and the distribution $0.5\Pr_1 + 0.5\Pr_{10}$. The weighted expected regret of any act with respect to $0.5\Pr_1 + 0.5\Pr_{10}$ is bounded above by the maximum weighted expected regret of that act with respect to $\Pr_1$ and $\Pr_{10}$. Therefore, adding $0.5\Pr_1 + 0.5\Pr_{10}$ to $\mathcal{P}^+$, with any weight in $(0, 1]$, yields another representation for the MWER preferences. Although we do not get a unique set $\mathcal{P}^+$ in the representation, the maximality requirement allows us to view the set $\mathcal{P}^+$ as canonical in a certain sense.

It is instructive to compare Theorem 1 to other representation results in the literature. Anscombe and Aumann [1] showed that the menu-independent versions of axioms 1–5 and 7 characterize SEU. The presence of Axiom 7 (menu-independent Independence) greatly simplifies things. Gilboa and Schmeidler [7] showed that axioms 1–6 together with one more axiom that they call *Certainty-independence* characterizes MMEU. Certainty-independence, or C-independence for short, is a weakening of independence (which, as we observed, does not hold for MMEU), where the act h is required to be a constant act. Since MMEU is menu-independent, we state it in a menu-independent way.

Axiom 11. *(C-Independence) If h is a constant act, then $f \succeq g$ iff $pf + (1-p)h \succeq pg + (1-p)h$.*

As we observed, in general, we have Ambiguity Aversion (Axiom 6) for regret. *Betweenness* [3] is a stronger notion than ambiguity aversion, which states that if an agent is indifferent between two acts, then he must also be indifferent among all convex combinations of these acts. While betweenness does not hold for regret, Stoye [16] gives a weaker version that does hold. A menu M has *state-independent outcome distributions* if the set $L(s) = \{y \in \Delta(Y) : \exists f \in M, f(s) = y\}$ is the same for all states s.

Axiom 12. *If h is a constant act, and M has state-independent outcome distributions, then*

$$h \sim_M f \Rightarrow pf + (1-p)h \sim_{M \cup \{pf+(1-p)h\}} f.$$

The assumption that the menu has state-independent outcome distributions is critical in Axiom 12.

Stoye [16] shows that Axioms 1–9 together with Axiom 12

	SEU	REG	MER	MWER	MMEU
Ax. 1-6,8-10	✓	✓	✓	✓	✓
Ind	✓	✓	✓	✓	
C-Ind	✓				✓
Ax. 12	✓	✓	✓		
Symmetry	✓	✓			

Table 4: Characterizing axioms for several decision rules.

characterize MER.[5] Non-probabilistic regret (which we denote REG) can be viewed as a special case of MER, where $\mathcal{P}$ consists of all distributions. This means that it satisfies all the axioms that MER satisfies. As Stoye [17] shows, REG is characterized by Axioms 1–9 and one additional axiom, which he calls Symmetry. We omit the details here.

The assumption that the menu has state-independent outcome distributions is critical in Axiom 12. In the full paper, we give an example showing that the variant of Axiom 12 without the state-independent outcome distribution requirement does not hold.

Table 4 describes the relationship between the axioms characterizing the decision rules.

5 Characterizing MWER with Likelihood Updating

We next consider a more dynamic setting, where agents learn information. For simplicity, we assume that the information is always a subset E of the state space. If the agent is representing her uncertainty using a set $\mathcal{P}^+$ of weighted probability measures, then we would expect her to update $\mathcal{P}^+$ to some new set $\mathcal{Q}^+$ of weighted probability measures, and then apply MWER with uncertainty represented by $\mathcal{Q}^+$. In this section, we characterize what happens in the special case that the agent uses likelihood updating, so that $\mathcal{Q}^+ = (\mathcal{P}^+ \mid E)$.

For this characterization, we assume that the agent has a family of preference orders $\succeq_{E,M}$ indexed not just by the menu M, but by the information E. Each preference order $\succeq_{E,M}$ satisfies Axioms 1–10, since the agent makes decisions after learning E using MWER. Somewhat surprisingly, all we need is one extra axiom for the characterization; we call this axiom MDC, for 'menu-dependent dynamic consistency'.

To explain the axiom, we need some notation. As usual, we take fEh to be the act that agrees with f on E and with h off of E; that is

$$fEh(s) = \begin{cases} f(s) & \text{if } s \in E \\ h(s) & \text{if } s \notin E. \end{cases}$$

[5] Stoye actually worked with choice correspondences; see Section 6.

In the delivery example, the act $check$ can be thought of as $(cont)E(back)$, where E is the set of states where there is only one broken cake.

Roughly speaking, MDC says that you prefer f to g once you learn E if and only if, for any act h, you also prefer fEh to gEh before you learn anything. This seems reasonable, since learning that the true state was in E is conceptually similar to knowing that none of your choices matter off of E.

To state MDC formally, we need to be careful about the menus involved. Let $MEh = \{fEh : f \in M\}$. We can identify unconditional preferences with preferences conditional on S; that is, we identify $\succeq_M$ with $\succeq_{S,M}$. We also need to restrict the sets E to which MDC applies. Recall that conditioning using likelihood updating is undefined for an event such that $\overline{\mathcal{P}}^+(E) = 0$. That is, $\alpha_{\Pr} \Pr(E) = 0$ for all $\Pr \in \mathcal{P}$. As is commonly done, we capture the idea that conditioning on E is possible using the notion of a *non-null* event.

Definition 1. *An event E is* null *if, for all $f, g \in \Delta(Y)^S$ and menus M with $fEg, g \in M$, we have $fEg \sim_M g$.*

MDC. For all non-null events E, $f \succeq_{E,M} g$ iff $fEh \succeq_{MEh} gEh$ for some $h \in M$.[6]

The key feature of MDC is that it allows us to reduce all the conditional preference orders $\succeq_{E,M}$ to the unconditional order $\succeq_M$, to which we can apply Theorem 1.

Theorem 2. *For all Y, U, S, and $\mathcal{P}^+$, the family of preference orders $\succeq^{S,Y,U}_{M,\mathcal{P}^+|E}$ for events E such that $\overline{\mathcal{P}}^+(E) > 0$ satisfies Axioms 1–10 and MDC. Conversely, if a family of preference orders $\succeq_{E,M}$ on the acts in $\Delta(Y)^S$ satisfies Axioms 1–10 and MDC, then there exist a weighted set $\mathcal{P}^+$ of probabilities on S and a utility U on Y non-null E, $\succeq_{E,M} = \succeq^{S,Y,U}_{M,\mathcal{P}^+|E}$. Moreover, U is unique up to affine transformations, and $\mathcal{P}^+$ can be taken to be maximal.*

Proof. Since $\succeq_M = \succeq_{S,M}$ satisfies Axioms 1–10, there must exist a weighted set $\mathcal{P}^+$ of probabilities on S and a utility function U such that $f \succeq_M g$ iff $f \succeq^{S,Y,U}_{M,\mathcal{P}^+} g$. We now show that if E is non-null, then $\overline{\mathcal{P}}^+(E) > 0$, and $f \succeq_{E,M} g$ iff $f \succeq^{(S,X,u)}_{M,\mathcal{P}^+|E} g$.

For the first part, it clearly is equivalent to show that if $\overline{\mathcal{P}}^+(E) = 0$, then E is null. So suppose that $\overline{\mathcal{P}}^+(E) = 0$. Then $\alpha_{\Pr} \Pr(E) = 0$ for all $\Pr \in \mathcal{P}$. This means that $\alpha_{\Pr} \Pr(s) = 0$ for all $\Pr \in \mathcal{P}$ and $s \in E$. Thus, for all acts

[6] Although we do not need this fact, it is worth noting that the MWER decision rule has the property that $fEh \succeq_{MEh} gEh$ for some act h iff $fEh \succeq_{MEh} gEh$ for all acts h. Thus, this property follows from Axioms 1–10.

f and g,

$$\begin{aligned} & reg_{M,\mathcal{P}^+}(fEg) \\ = \; & \sup_{\Pr\in\mathcal{P}} \left(\alpha_{\Pr} \textstyle\sum_{s\in S} \Pr(s) reg_M(fEg, s)\right) \\ = \; & \sup_{\Pr\in\mathcal{P}} \left(\alpha_{\Pr} \left(\textstyle\sum_{s\in E} \Pr(s) reg_M(f, s)\right)\right. \\ & \left. + \textstyle\sum_{s\in E^c} \Pr(s) reg_M(g, s)\right) \\ = \; & \sup_{\Pr\in\mathcal{P}} \left(\alpha_{\Pr} \textstyle\sum_{s\in S} \Pr(s) reg_M(g, s)\right) \\ = \; & reg_{M,\mathcal{P}^+}(g). \end{aligned}$$

Thus, $fEg \sim_M g$ for all acts f, g and menus M containing fEg and g, which means that E is null.

For the second part, we first show that if $\overline{\mathcal{P}}^+(E) > 0$, then for all $f, h \in M$, we have that

$$reg_{MEh,\mathcal{P}^+}(fEh) = \overline{\mathcal{P}}^+(E) reg_{M,\mathcal{P}^+|E}(f).$$

We proceed as follows:

$$\begin{aligned} & reg_{MEh,\mathcal{P}^+}(fEh) \\ = \; & \sup_{\Pr\in\mathcal{P}} \left(\alpha_{\Pr} \textstyle\sum_{s\in S} \Pr(s) reg_{MEh}(fEH, s)\right) \\ = \; & \sup_{\Pr\in\mathcal{P}} \left(\alpha_{\Pr} \Pr(E) \textstyle\sum_{s\in E} \Pr(s \mid E) reg_M(f, s)\right. \\ & \left. + \alpha_{\Pr} \textstyle\sum_{s\in E^c} \Pr(s) reg_{\{h\}}(h, s)\right) \\ = \; & \sup_{\Pr\in\mathcal{P}} \left(\alpha_{\Pr} \Pr(E) \textstyle\sum_{s\in E} \Pr(s|E) reg_M(s, f)\right) \\ = \; & \sup_{\Pr\in\mathcal{P}} \left(\overline{\mathcal{P}}^+(E) \alpha_{\Pr|E} \textstyle\sum_{s\in E} \Pr(s|E) reg_M(f, s)\right) \\ & [\text{since } \alpha_{\Pr|E} = \sup_{\{\Pr'\in\mathcal{P}:\Pr'|E=\Pr|E\}} \tfrac{\alpha_{\Pr'} \Pr'(E)}{\overline{\mathcal{P}}^+(E)}] \\ = \; & \overline{\mathcal{P}}^+(E) \cdot reg_{M,\mathcal{P}^+|E}(f). \end{aligned}$$

Thus, for all $h \in M$,

$$\begin{aligned} & reg_{MEh,\mathcal{P}^+}(fEh) \le reg_{MEh,\mathcal{P}^+}(gEh) \\ \text{iff } & \overline{\mathcal{P}}^+(E) \cdot reg_{M,\mathcal{P}^+|E}(f) \le \overline{\mathcal{P}}^+(E) \cdot reg_{M,\mathcal{P}^+|E}(g) \\ \text{iff } & reg_{M,\mathcal{P}^+|E}(f) \le reg_{M,\mathcal{P}^+|E}(g). \end{aligned}$$

It follows that the order induced by $\Pr^+$ satisfies MDC.

Moreover, if 1–10 and MDC hold, then for the weighted set $\mathcal{P}^+$ that represents $\succeq_M$, we have

$$\begin{aligned} & f \succeq_{E,M} g \\ \text{iff} \quad & \text{for some } h \in M, fEh \succeq_{MEh} gEh \\ \text{iff} \quad & reg_{M,\mathcal{P}^+|E}(f) \le reg_{M,\mathcal{P}^+|E}(g), \end{aligned}$$

as desired. □

Analogues of MDC have appeared in the literature before in the context of updating preference orders. In particular, Epstein and Schneider [4] discuss a menu-independent version of MDC, although they do not characterize updating in their framework. Sinischalchi [15] also uses an analogue of MDC in his axiomatization of measure-by-measure updating of MMEU. Like us, he starts with an axiomatization for unconditional preferences, and adds an axiom called *constant-act dynamic consistency* (CDC), somewhat analogous to MDC, to extend the axiomatization of MMEU to deal with conditional preferences.

6 Conclusion

We proposed an alternative belief representation using *weighted sets of probabilities*, and described a natural approach to updating in such a situation and a natural approach to determining the weights. We also showed how weighted sets of probabilities can be combined with regret to obtain a decision rule, MWER, and provided an axiomatization that characterizes static and dynamic preferences induced by MWER.

We have considered preferences indexed by menus here. Stoye [17] used a different framework: *choice functions*. A choice function maps every finite set M of acts to a subset M' of M. Intuitively, the set M' consists of the 'best' acts in M. Thus, a choice function gives less information than a preference order; it gives only the top elements of the preference order. The motivation for working with choice functions is that an agent can reveal his most preferred acts by choosing them when the menu is offered. In a menu-independent setting, the agent can reveal his whole preference order; to decide if $f \succ g$, it suffices to present the agent with a choice among $\{f, g\}$. However, with regret-based choices, the menu matters; the agent's most preferred choice(s) when presented with $\{f, g\}$ might no longer be the most preferred choice(s) when presented with a larger menu. Thus, a whole preference order is arguably not meaningful with regret-based choices. Stoye [17] provides a representation theorem for MER where the axioms are described in terms of choice functions. The axioms that we have attributed to Stoye are actually the menu-based analogue of his axioms. We believe that it should be possible to provide a characterization of MWER using choice functions, although we have not yet proved this.

Finally, we briefly considered the issue of dynamic consistency and consistent planning. As we showed, making this precise in the context of regret involves a number of subtleties. We hope to return to this issue in future work.

Acknowledgments: We thank Joerg Stoye and Edward Lui for useful comments.

References

[1] F. Anscombe and R. Aumann. A definition of subjective probability. *Annals of Mathematical Statistics*, 34:199–205, 1963.

[2] A. Chateauneuf and J. Faro. Ambiguity through confidence functions. *Journal of Mathematical Economics*, 45:535 – 558, 2009.

[3] S. H. Chew. A generalization of the quasilinear mean with applications to the measurement of income inequality and decision theory resolving the allais paradox. *Econometrica*, 51(4):1065–92, July 1983.

[4] L. G. Epstein and M. Le Breton. Dynamically consistent beliefs must be Bayesian. *Journal of Economic Theory*, 61(1):1–22, 1993.

[5] L. G. Epstein and M. Schneider. Learning under ambiguity. *Review of Economic Studies*, 74(4):1275–1303, 2007.

[6] I. Gilboa and D. Schmeidler. Maxmin expected utility with a non-unique prior. *Journal of Mathematical Economics*, 18:141–153, 1989.

[7] I. Gilboa and D. Schmeidler. Maxmin expected utility with non-unique prior. *Journal of Mathematical Economics*, 18(2):141–153, 1989.

[8] T. Hayashi. Regret aversion and opportunity dependence. *Journal of Economic Theory*, 139(1):242–268, 2008.

[9] J.-Y. Jaffray. Dynamic decision making with belief functions. In R. R. Yager, J. Kacprczyk, and M. Fedrizzi, editors, *Advances in the Dempster-Shafer Theory of Evidence*, pages 331–352. Wiley, New York, 1994.

[10] J. Niehans. Zur preisbildung bei ungewissen erwartungen. *Schweizerische Zeitschrift für Volkswirtschaft und Statistik*, 84(5):433–456, 1948.

[11] C. P. Pires. A rule for updating ambiguous beliefs. *Theory and Decision*, 53(2):137–152, 2002.

[12] T. Sarver. Anticipating regret: Why fewer options may be better. *Econometrica*, 76(2):263–305, 2008.

[13] L. Savage. The theory of statistical decision. *Journal of the American Statistical Association*, 46:55–67, 1951.

[14] L. Savage. *The Foundations of Statistics*. Wiley, New York, 1954.

[15] M. Siniscalchi. Dynamic choice under ambiguity. *Theoretical Economics*, 6(3):379–421, 2011.

[16] J. Stoye. Axioms for minimax regret choice correspondences. *Journal of Economic Theory*, 146(6):2226 – 2251, 2011.

[17] J. Stoye. Statistical decisions under ambiguity. *Theory and Decision*, 70(2):129–148, 2011.

Selecting Computations: Theory and Applications

Nicholas Hay and Stuart Russell
Computer Science Division
University of California
Berkeley, CA 94720
{nickjhay,russell}@cs.berkeley.edu

David Tolpin and Solomon Eyal Shimony
Department of Computer Science
Ben-Gurion University of the Negev
Beer Sheva 84105, Israel
{tolpin,shimony}@cs.bgu.ac.il

Abstract

Sequential decision problems are often approximately solvable by simulating possible future action sequences. *Metalevel* decision procedures have been developed for selecting *which* action sequences to simulate, based on estimating the expected improvement in decision quality that would result from any particular simulation; an example is the recent work on using bandit algorithms to control Monte Carlo tree search in the game of Go. In this paper we develop a theoretical basis for metalevel decisions in the statistical framework of Bayesian *selection problems*, arguing (as others have done) that this is more appropriate than the bandit framework. We derive a number of basic results applicable to Monte Carlo selection problems, including the first finite sampling bounds for optimal policies in certain cases; we also provide a simple counterexample to the intuitive conjecture that an optimal policy will necessarily reach a decision in all cases. We then derive heuristic approximations in both Bayesian and distribution-free settings and demonstrate their superiority to bandit-based heuristics in one-shot decision problems and in Go.

1 Introduction

The broad family of sequential decision problems includes combinatorial search problems, game playing, robotic path planning, model-predictive control problems, Markov decision processes (MDP), whether fully or partially observable, and a huge range of applications. In almost all realistic instances, exact solution is intractable and approximate methods are sought. Perhaps the most popular approach is to simulate a limited number of possible future action sequences, in order to find a move in the current state that is (hopefully) near-optimal. In this paper, we develop a theoretical framework to examine the problem of selecting *which* future sequences to simulate. We derive a number of new results concerning optimal policies for this selection problem as well as new heuristic policies for controlling Monte Carlo simulations. As described below, these policies outperform previously published methods for "flat" selection and game-playing in Go.

The basic ideas behind our approach are best explained in a familiar context such as game playing. A typical game-playing algorithm chooses a move by first exploring a tree or graph of move sequences and then selecting the most promising move based on this exploration. Classical algorithms typically explore in a fixed order, imposing a limit on exploration depth and using pruning methods to avoid irrelevant subtrees; they may also reuse some previous computations (see Section 6.2). Exploring unpromising or highly predictable paths to great depth is often wasteful; for a given amount of exploration, decision quality can be improved by directing exploration towards those actions sequences whose outcomes are helpful in selecting a good move. Thus, the *metalevel* decision problem is to choose what future action sequences to explore (or, more generally, what deliberative computations to do), while the *object-level* decision problem is to choose an action to execute in the real world.

That the metalevel decision problem can itself be formulated and solved decision-theoretically was noted by Matheson (1968), borrowing from the related concept of *information value theory* (Howard, 1966). In essence, computations can be selected according to the expected improvement in decision quality resulting from their execution. I. J. Good (1968) independently proposed using this idea to control search in chess, and later defined "Type II rationality" to refer to agents that optimally solve the metalevel decision problem before acting. As interest in probabilistic and decision-

theoretic approaches in AI grew during the 1980s, several authors explored these ideas further (Dean and Boddy, 1988; Doyle, 1988; Fehling and Breese, 1988; Horvitz, 1987). Work by Russell and Wefald (1988, 1991a,b) formulated the metalevel sequential decision problem, employing an explicit model of the results of computational actions, and applied this to the control of game-playing search in Othello with encouraging results.

An independent thread of research on metalevel control began with work by Kocsis and Szepesvári (2006) on the UCT algorithm, which operates in the context of *Monte Carlo tree search* (MCTS) algorithms. In MCTS, each computation takes the form of a simulation of a randomized sequence of actions leading from a leaf of the current tree to a terminal state. UCT is primarily a method for selecting a leaf from which to conduct the next simulation, and forms the core of the successful MoGo algorithm for Go playing (Gelly and Silver, 2011). The UCT algorithm is based on the the theory of bandit problems (Berry and Fristedt, 1985) and the asymptotically near-optimal UCB1 bandit algorithm (Auer et al., 2002). UCT applies UCB1 recursively to select actions to perform within simulations.

It is natural to consider whether the two independent threads are consistent; for example, are bandit algorithms such as UCB1 approximate solutions to some particular case of the metalevel decision problem defined by Russell and Wefald? The answer, perhaps surprisingly, is no. The essential difference is that, in bandit problems, every trial involves executing a real object-level action with real costs, whereas in the metareasoning problem the trials are *simulations* whose cost is usually independent of the utility of the action being simulated. Hence, as Audibert et al. (2010) and Bubeck et al. (2011) have also noted, UCT applies bandit algorithms to problems that are not bandit problems.

One consequence of the mismatch is that bandit policies are inappropriately biased away from exploring actions whose current utility estimates are low. Another consequence is the absence of any notion of "stopping" in bandit algorithms, which are designed for infinite sequences of trials. A metalevel policy needs to decide when to stop deliberating and execute a real action.

Analyzing the metalevel problem within an appropriate theoretical framework ought to lead to more effective algorithms than those obtained within the bandit framework. For Monte Carlo computations, in which samples are gathered to estimate the utilities of actions, the metalevel decision problem is an instance of the *selection problem* studied in statistics (Bechhofer, 1954; Swisher et al., 2003). Despite some recent work (Frazier and Powell, 2010; Tolpin and Shimony, 2012b), the theory of selection problems is less well understood than that of bandit problems. Most work has focused on the probability of selection error rather than optimal policies in the Bayesian setting (Bubeck et al., 2011). Accordingly, we present in Sections 2 and 3 a number of results concerning optimal policies for the general case as well as specific finite bounds on the number of samples collected by optimal policies for Bernoulli arms with beta priors. We also provide a simple counterexample to the intuitive conjecture that an optimal policy should not spend more on deciding than the decision is worth; in fact, it is possible for an optimal policy to compute forever. We also show by counterexample that optimal *index policies* (Gittins, 1989) may not exist for selection problems.

Motivated by this theoretical analysis, we propose in Sections 4 and 5 two families of heuristic approximations, one for the Bayesian case and one for the distribution-free setting. We show empirically that these rules give better performance than UCB1 on a wide range of standard (non-sequential) selection problems. Section 6 shows similar results for the case of guiding Monte Carlo tree search in the game of Go.

2 On optimal policies for selection

In a selection problem the decision maker is faced with a choice among alternative arms[1]. To make this choice, they may gather evidence about the utility of each of these alternatives, at some cost. The objective is to maximize the *net utility*, i.e., the expected utility of the final arm selected, less the cost of gathering the evidence. In the classical case (Bechhofer, 1954), evidence might consist of physical samples from a product batch; in a metalevel problem with Monte Carlo simulations, the evidence consists of outcomes of sampling computations:

Definition 1. *A **metalevel probability model** is a tuple $(U_1, \ldots, U_k, \mathcal{E})$ consisting of jointly distributed random variables:*

- *Real random variables $U_1, \ldots, U_k$, where U_i is the utility of arm i, and*
- *A countable set $\mathcal{E}$ of random variables, each variable $E \in \mathcal{E}$ being a computation that can be performed and whose value is the result of that computation, where $e \in E$ will denote that e is a potential value of the computation E.*

[1] Alternative actions are known as *arms* in the bandit setting; we borrow this terminology for uniformity.

For simplicity, in the below we'll assume the utilities U_i are bounded, without loss of generality in $[0, 1]$.

Example 1 (Bernoulli sampling)**.** *In the* ***Bernoulli metalevel probability model****, each arm will either succeed or not $U_i \in \{0, 1\}$, with an unknown latent frequency of success Θ_i, and a set of stochastic simulations of possible consequences $\mathcal{E} = \{E_{ij} | 1 \leq i \leq k, j \in \mathbb{N}\}$ that can be performed:*

$$\begin{aligned} \Theta_i &\overset{\text{iid}}{\sim} \text{Uniform}[0, 1] && \text{for } i \in \{1, \ldots, k\} \\ U_i \mid \Theta_i &\sim \text{Bernoulli}(\Theta_i) && \text{for } i \in \{1, \ldots, k\} \\ E_{ij} \mid \Theta_i &\overset{\text{iid}}{\sim} \text{Bernoulli}(\Theta_i) && \text{for } i \in \{1, \ldots, k\},\ j \in \mathbb{N} \end{aligned}$$

The ***one-armed Bernoulli metalevel probability model*** *has $k = 2$, $\Theta_1 = \lambda \in [0, 1]$ a constant, and $\Theta_2 \sim \text{Uniform}[0, 1]$.*

A metalevel probability model, when combined with a cost of computation $c > 0$,[2] defines a metalevel decision problem: what is the optimal strategy with which to choose a sequence of computations $E \in \mathcal{E}$ in order to maximize the agent's net utility? Intuitively, this strategy should choose the computations that give the most evidence relevant to deciding which arm to use, stopping when the cost of computation outweighs the benefit gained. We formalize the selection problem as a Markov Decision Process (see, e.g., Puterman (1994)):

Definition 2. *A (countable state, undiscounted)* ***Markov Decision Process*** *(MDP) is a tuple $M = (S, s_0, A_s, T, R)$ where: S is a countable set of states, $s_0 \in S$ is the fixed initial state, A_s is a countable set of actions available in state $s \in S$, $T(s, a, s')$ is the transition probability from $s \in S$ to $s' \in S$ after performing action $a \in A_s$, and $R(s, a, s')$ is the expected reward received on such a transition.*

To formulate the metalevel decision problem as an MDP, we define the states as sequences of computation outcomes and allow for a terminal state when the agent chooses to stop computing and act:

Definition 3. *Given a metalevel probability model*[3] *$(U_1, \ldots, U_k, \mathcal{E})$ and a cost of computation $c > 0$, a corresponding* ***metalevel decision problem*** *is any MDP $M = (S, s_0, A_s, T, R)$ such that*

$$\begin{aligned} S &= \{\bot\} \cup \{\langle e_1 \ldots, e_n\rangle : e_i \in E_i \text{ for all } i, \\ &\qquad \text{for finite } n \geq 0 \text{ and distinct } E_i \in \mathcal{E}\} \\ s_0 &= \langle\rangle \\ A_s &= \{\bot\} \cup \mathcal{E}_s \end{aligned}$$

where $\bot \in S$ is the unique terminal state, where $\mathcal{E}_s \subseteq \mathcal{E}$ is a state-dependent subset of allowed computations, and when given any $s = \langle e_1, \ldots, e_n\rangle \in S$, computational action $E \in \mathcal{E}$, and $s' = \langle e_1, \ldots, e_n, e\rangle \in S$ where $e \in E$, we have:

$$\begin{aligned} T(s, E, s') &= P(E = e \mid E_1 = e_1, \ldots, E_n = e_n) \\ T(s, \bot, \bot) &= 1 \\ R(s, E, s') &= -c \\ R(s, \bot, \bot) &= \max_i \mu_i(s) \end{aligned}$$

where $\mu_i(s) = \mathbb{E}[U_i \mid E_1 = e_1, \ldots, E_n = e_n]$.

Note that when stopping in state s, the expected utility of action i is by definition $\mu_i(s)$, so the optimal action to take is $i^* \in \text{argmax}_i\, \mu_i(s)$ which has expected utility $\mu_{i*}(s) = \max_i \mu_i(s)$.

One can optionally add an external constraint on the number of computational actions, or their total cost, in the form of a deadline or *budget*. This bridges with the related area of budgeted learning (Madani et al., 2004). Although this feature is not formalized in the MDP, it can be added by including either time or past total cost as part of the state.

Example 2 (Bernoulli sampling)**.** *In the Bernoulli metalevel probability model (Example 1), note that:*

$$\Theta_i \mid E_{i1}, \ldots, E_{in_i} \sim \text{Beta}(s_i + 1, f_i + 1) \quad (1)$$

$$E_{i(n_i+1)} \mid E_{i1}, \ldots, E_{in_i} \sim \text{Bernoulli}\left(\frac{s_i + 1}{n_i + 2}\right) \quad (2)$$

$$\mathbb{E}[U_i \mid E_{i1}, \ldots, E_{in_i}] = (s_i + 1)/(n_i + 2) \quad (3)$$

by standard properties of these distributions, where $s_i = \sum_{j=1}^{n_i} E_{in_i}$ is the number of simulated successes of arm i, and $f_i = n_i - s_i$ the failures. By Equation (1), the state space is the set of all k pairs (s_i, f_i); Equations (2) and (3) suffice to give the transition probabilities and terminal rewards, respectively. The one-armed Bernoulli case is similar, requiring as state just (s, f) defining the posterior over Θ_2.

Given a metalevel decision problem $M = (S, s_0, A_s, T, R)$ one defines policies and value functions as in any MDP. A (deterministic, stationary) **metalevel policy** π is a function mapping states $s \in S$ to actions to take in that state $\pi(s) \in A_s$.

[2] The assumption of a fixed cost of computation is a simplification; precise conditions for its validity are given by Harada (1997).

[3] Definition 1 made no assumption about the computational result variables $E_i \in \mathcal{E}$, but for simplicity in the following we'll assume that each E_i takes one of a countable set of values. Without loss of generality, we'll further assume the domains of the computational variables $E \in \mathcal{E}$ are disjoint.

The **value function** for a policy π gives the expected total reward received under that policy starting from a given state $s \in S$, and the **Q-function** does the same when starting in a state $s \in S$ and taking a given action $a \in A_s$:

$$V_M^\pi(s) = \mathbb{E}_M^\pi \left[\sum_{i=0}^{N} R(S_i, \pi(S_i), S_{i+1}) \mid S_0 = s \right] \quad (4)$$

where $N \in [0, \infty]$ is the random time the MDP is terminated, i.e., the unique time where $\pi(S_N) = \perp$, and similarly for the Q-function $Q_M^\pi(s, a)$.

As usual, an **optimal policy** π^*, when it exists, is one that maximizes the value from every state $s \in S$, i.e., if we define for each $s \in S$

$$V_M^*(s) = \sup_\pi V_M^\pi(s),$$

then an optimal policy π^* satisfies $V_M^{\pi^*}(s) = V_M^*(s)$ for all $s \in S$, where we break ties in favor of stopping.

The optimal policy must balance the cost of computations with the improved decision quality that results. This tradeoff is made clear in the value function:

Theorem 4. *The value function of a metalevel decision process $M = (S, s_0, A_s, T, R)$ is of the form*

$$V_M^\pi(s) = \mathbb{E}_M^\pi[-c\,N + \max_i \mu_i(S_N) \mid S_0 = s]$$

where N denotes the (random) total number of computations performed; similarly for $Q_M^\pi(s, a)$.

In many problems, including the Bernoulli sampling model of Example 2, the state space is infinite. Does this preclude solving for the optimal policy? Can infinitely many computations be performed?

There is in full generality an upper bound on the *expected* number of computations a policy performs:

Theorem 5. *The optimal policy's expected number of computations is bounded by the value of perfect information (Howard, 1966) times the inverse cost $1/c$:*

$$\mathbb{E}^{\pi^*}[N \mid S_0 = s] \leq \frac{1}{c} \left(\mathbb{E}[\max_i U_i \mid S_0 = s] - \max_i \mu_i(s) \right).$$

Further, any policy π with infinite expected number of computations has negative infinite value, hence the optimal policy stops with probability one.

Although the *expected* number of computations is always bounded, there are important cases in which the *actual* number is not, such as the following inspired by the sequential probability ratio test (Wald, 1945):

Example 3. *Consider the Bernoulli sampling model for two arms but with a different prior: $\Theta_1 = 1/2$, and Θ_2 is $1/3$ or $2/3$ with equal probability. Simulating arm 1 gains nothing, and after (s, f) simulated successes and failures of arm 2 the posterior odds ratio is*

$$\frac{P(\Theta_2 = 2/3 \mid s, f)}{P(\Theta_2 = 1/3 \mid s, f)} = \frac{(2/3)^s (1/3)^f}{(1/3)^s (2/3)^f} = 2^{s-f}.$$

Note that this ratio completely specifies the posterior distribution of Θ_2, and hence the distribution of the utilities and all future computations. Thus, whether it is optimal to continue is a function only of this ratio, and thus of $s - f$. For sufficiently low cost, the optimal policy samples when $s - f$ equals -1, 0, or 1. But with probability $1/3$, a state with $s - f = 0$ transitions to another state $s - f = 0$ after two samples, giving finite, although exponentially decreasing, probability to arbitrarily long sequences of computations.

However, in a number of settings, including the original Bernoulli model of Example 1, we can prove an upper bound on the number of computations. For reasons of space, and for its later use in Section 4, we prove here the bound for the one-armed Bernoulli model.

Before we can do this, we need to get an analytical handle on the optimal policy. The key is through a natural approximate policy:

Definition 6. *Given a metalevel decision problem $M = (S, s_0, A_s, T, R)$, the **myopic policy** $\pi^m(s)$ is defined to equal* $\operatorname{argmax}_{a \in A_s} Q^m(s, a)$ *where $Q^m(s, \perp) = \max_i \mu_i(s)$ and*

$$Q^m(s, E) = \mathbb{E}_M[-c + \max_i \mu_i(S_1) \mid S_0 = s, A_0 = E].$$

The myopic policy (known as the metalevel greedy approximation with single-step assumption in (Russell and Wefald, 1991a)) takes the best action, to either stop or perform a computation, under the assumption that at most one further computation can be performed. It has a tendency to stop too early, because changing one's mind about which real action to take often takes more than one computation. In fact, we have:

Theorem 7. *Given a metalevel decision problem $M = (S, s_0, A_s, T, R)$ if the myopic policy performs some computation in state $s \in S$, then the optimal policy does too, i.e., if $\pi^m(s) \neq \perp$ then $\pi^*(s) \neq \perp$.*

Despite this property, the stopping behavior of the myopic policy does have a close connection to that of the optimal policy:

Definition 8. *Given a metalevel decision problem $M = (S, s_0, A_s, T, R)$, a subset $S' \subseteq S$ of states is **closed under transitions** if whenever $s' \in S'$, $a \in A_{s'}$, $s'' \in S$, and $T(s', a, s'') > 0$, we have $s'' \in S'$.*

Theorem 9. *Given a metalevel decision problem $M = (S, s_0, A_s, T, R)$ and a subset $S' \subseteq S$ of states closed under transitions, if the myopic policy stops in all states $s' \in S'$ then the optimal policy does too.*

Using these results connecting the behavior of the optimal and myopic policies, we can prove our bound:

Theorem 10. *The one-armed Bernoulli decision process with constant arm $\lambda \in [0, 1]$ performs at most $\lambda(1-\lambda)/c - 3 \leq 1/4c - 3$ computations.*

Proof. Using Definition 6 and Example 2, we determine which states the myopic policy stops in by bounding $Q^m(s, E)$. For a state (s, f), let $\mu = (s+1)/(n+2)$ be the mean utility for arm 2, where $n = s + f$. Fixing n and maximizing over μ, we get sufficient condition for stopping Since the set of states satisfying Equation (5) is closed under

$$c \geq \frac{\lambda(1-\lambda)}{(n+3)} \qquad n \geq \frac{\lambda(1-\lambda)}{c} - 3 \qquad (5)$$

Since the set of states satisfying Equation (5) is closed under transitions (n only increases), by Theorem 7. Finally, note $\max_{\lambda \in [0,1]} \lambda(1-\lambda) = 1/4$. □

A key implication is that the *optimal* policy can be computed in time $O(1/c^2)$, i.e., quadratic in the inverse cost. This is particularly appropriate when the cost of computation is relatively high, such as in simulation experiments (Swisher et al., 2003), or when the decision to be made is critical.

3 Context effects and non-indexability

The Gittins index theorem (Gittins, 1979) is a famous structural result for bandit problems. It states that in bandit problems with independent reward distribution for each arm and geometric discounting, the optimal policy is an **index policy**: each arm is assigned a real-valued index based on its state only, such that it is optimal to sample the arm with greatest index.

The analogous result does *not* hold for metalevel decision problems, even when the action's values are independent (this formalized later in Definition 13):

Example 4 (Non-indexability)**.** *Consider a metalevel probability model with three actions. U_1 is equally likely to be -1.5 or 1.5 (low mean, high variance), U_2 is equally likely to be 0.25 or 1.75 (high mean, low variance), and $U_3 = \lambda$ has a known value (the context). The two computations are to observe exactly U_1 and U_2, respectively, each with cost 0.2. The corresponding metalevel MDP has 9 states and can be solved exactly. Figure 1 plots $Q^*_\lambda(s_0, U_i) - Q^*_\lambda(s_0, \perp)$ as a function of the known value λ. As the context λ varies the optimal action inverts from observing 1 to observing 2. Inversions like this are impossible for index policies.*

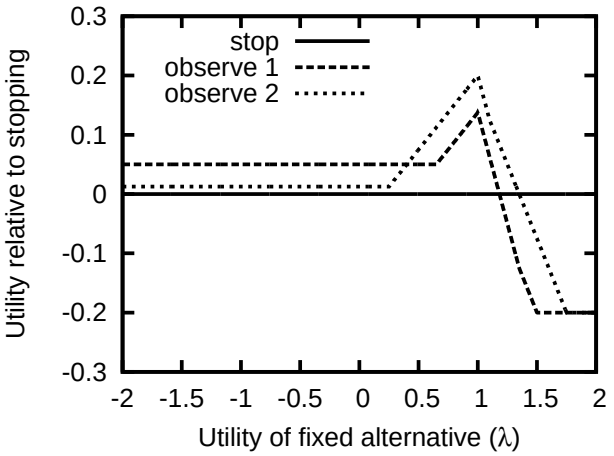


Figure 1: Optimal Q-values of computing relative to stopping as a function of the utility of the fixed alternative. Note the inversion where for low λ observing action 1 is strictly optimal, while for medium λ observing action 2 is strictly optimal.

There is, however, a restriction on what kind of influence the context can have:

Definition 11. *Given a metalevel decision problem $M = (S, s_0, A_s, T, R)$, and a constant $\lambda \in \mathbb{R}$, define $M_\lambda = (S, s_0, A_s, T, R_\lambda)$ to be M with an additional action of known value λ, defined by:*

$$R_\lambda(s, E, s') = R(s, E, s')$$
$$R_\lambda(s, \perp, \perp) = \max\{\lambda, R(s, \perp, \perp)\}$$

Note this is equivalent to adding an extra arm with constant value $U_{k+1} = \lambda$.

Theorem 12. *Given a metalevel decision problem $M = (S, s_0, A_s, T, R)$, there exists a real interval $I(s)$ for every state $s \in S$ such that it is optimal to stop in state s in M_μ iff $\mu \notin I(s)$. Furthermore, $I(s)$ contains $\max_i \mu_i(s)$ whenever it is nonempty.*

4 The blinkered policy

The myopic policy is an extreme approximation, often stopping far too early. A better approximation can be obtained, at least for the case where each computation can only affect the value of one action. The technical definition (closely related to *subtree independence* in Russell and Wefald's work) is as follows:

Definition 13. *A metalevel probability model $(U_1, \ldots, U_k, \mathcal{E})$ has **independent actions** if the computational variables can be partitioned $\mathcal{E} = \mathcal{E}_1 \cup \cdots \cup \mathcal{E}_k$*

such that such that the sets $\{U_i\} \cup \mathcal{E}_i$ *are independent of each other for different* i*.*

With independent actions, we can talk about metalevel policies that focus on computations affecting a single action. These policies are not myopic—they can consider arbitrarily many computations—but they are *blinkered* because they can look in only a single direction at a time:

Definition 14. *Given a metalevel decision problem* $M = (S, s_0, A_s, T, R)$ *with independent actions, the* ***blinkered policy*** π^b *is defined by* $\pi^b(s) = \text{argmax}_{a \in A_s} Q^b(s, a)$ *where* $Q^b(s, \bot) = \bot$ *and for* $E_i \in \mathcal{E}_i$

$$Q^b(s, E_i) = \sup_{\pi \in \Pi_i^b} Q^\pi(s, E_i) \tag{6}$$

where Π_i^b *is the set of policies* π *where* $\pi(s) \in \mathcal{E}_i$ *for all* $s \in S$*.*

Clearly, blinkered policies are better than myopic: $Q^m(s, a) \leq Q^b(s, a) \leq Q^*(s, a)$. Moreover, the blinkered policy can be computed in time proportional to the number of arms, by breaking the decision problem into separate subproblems:

Definition 15. *Given a metalevel decision problem* $M = (S, s_0, A_s, T, R)$ *with independent actions, a* ***one-action metalevel decision problem*** *for* $i = 1, \ldots, k$ *is the metalevel decision problem* $M^1_{i,\lambda} = (S_i, s_0, A_{s0}, T_i, R_i)$ *defined by the metalevel probability model* $(U_0, U_i, \mathcal{E}_i)$ *with* $U_0 = \lambda$*.*

Note that given a state s of a metalevel decision problem, we can form a state s_i by taking only the results of computations in $\mathcal{E}_i$ (see Definition 3). By action independence, $\mu_i(s)$ is a function only of s_i.

Theorem 16. *Given a metalevel decision problem* $M = (S, s_0, A_s, T, R)$ *with independent actions, let* M^1_{i,λ_i} *be the* i*th one-action metalevel decision problem for* $i = 1, \ldots, k$*. Then for any* $s \in S$*, whenever* $E_i \in A_s \cap \mathcal{E}_i$ *we have:*

$$Q^b_M(s, E_i) = Q^*_{M^1_{i,\mu^*_{-i}}}(s_i, E_i)$$

where $\mu^*_{-i} = \max_{j \neq i} \mu_j(s)$*.*

Theorem 16 shows that to compute the blinkered policy we need only compute the optimal policies for k separate one-action problems.

For the Bernoulli problem with k actions, the one-action metalevel decision problems are all one-action Bernoulli problems (Example 1). By Theorem 10 these policies perform at most $1/4c - 3$ computations. As a result, the blinkered policy can be numerically computed in time $O(D/c^2)$ independent of k by backwards induction, where D is the number of points $\lambda \in [0, 1]$ for which we compute $Q^*_{M^1_{i,m}}(s)$.[4] This will be worth the cost in the same situations as mentioned at the end of Section 2.

Figure 2 compares the blinkered policy to several other policies from the literature, using a Bernoulli sampling problem with $k = 25$ and a wide range of values for the step cost c. Performance is measured by expected *regret*, where the regret includes the cost of sampling: $R = (\max_i U_i) - U_j + c\,n$ where n is the number of computations and j is the action actually selected. The blinkered policy significantly outperforms all others. The myopic policy plateaus as it quickly reaches a position where no single computation can change the final action choice. ESPb performs quite well given that is making a normal approximation to the Beta posterior. The curves for UCB1-B and UCB1-b show that even given a good stopping rule, UCB1's choice of actions to sample is not ideal.

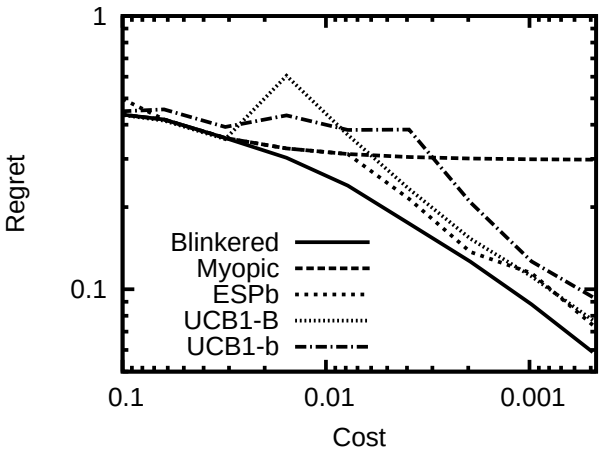


Figure 2: Average regret of various policies as a function of the cost in a 25-action Bernoulli sampling problem, over 1000 trials. Error bars omitted as they are negligible (the relative error is at most 0.03).

5 Upper bounds on Value of Information

In many practical applications of the selection problem, such as search in the game of Go, prior distributions are unavailable.[5] In such cases, one can still bound the value of information of myopic policies using *concentration inequalities* to derive distribution-independent bounds on the VOI. We obtain such bounds under the following assumptions:

[4] In our experiments below, $D = 129$ points are equally spaced, using linear interpolation between points.

[5] The analysis is also applicable to some Bayesian settings, using "fake" samples to simulate prior distributions.

1. Samples are iid given the value of the arms (variables), as in the Bayesian schemes such as Bernoulli sampling.

2. The expectation of a selection in a belief state is equal to the sample mean (and therefore, after sampling terminates, the arm with the greatest sample mean will be selected).

When considering possible samples in the blinkered semi-myopic setting, two cases are possible: either the arm α with the highest sample mean $\overline{X}_\alpha$ is tested, and $\overline{X}_\alpha$ becomes lower than $\overline{X}_\beta$ of the second-best arm β; or, another arm i is tested, and $\overline{X}_i$ becomes higher than $\overline{X}_\alpha$.

Our bounds below are applicable to any bounded distribution (without loss of generality bounded in $[0,1]$). Similar bounds can be derived for certain unbounded distributions, such as the normally distributed prior value with normally distributed sampling. We derive a VOI bound for testing an arm a fixed N times, where N can be the remaining budget of available samples or any other integer quantity. Denote by Λ_i^b the intrinsic VOI of testing the ith arm N times, and the number of samples already taken from the ith arm by n_i.

Theorem 17. *Λ_i^b is bounded from above as*

$$\Lambda_\alpha^b \leq \frac{N\overline{X}_\beta^{n_\beta}}{n_\alpha} \Pr(\overline{X}_\alpha^{n_\alpha+N} \leq \overline{X}_\beta^{n_\beta})$$
$$\Lambda_{i|i\neq\alpha}^b \leq \frac{N(1-\overline{X}_\alpha^{n_\alpha})}{n_i} \Pr(\overline{X}_i^{n_i+N} \geq \overline{X}_\alpha^{n_\alpha}) \quad (7)$$

The probabilities can be bounded from above using the Hoeffding inequality (Hoeffding, 1963):

Theorem 18. *The probabilities in Equation (7) are bounded from above as*

$$\Pr(\overline{X}_\alpha^{n_\alpha+N} \leq \overline{X}_\beta^{n_\beta}) \leq 2\exp\left(-\varphi(\overline{X}_\alpha^{n_\alpha} - \overline{X}_\beta^{n_\beta})^2 n_\alpha\right)$$
$$\Pr(\overline{X}_{i|i\neq\alpha}^{n_\alpha+N} \geq \overline{X}_\beta^{n_\beta}) \leq 2\exp\left(-\varphi(\overline{X}_\alpha^{n_\alpha} - \overline{X}_i^{n_i})^2 n_i\right) \quad (8)$$

where $\varphi = \min\left(2(\frac{1+n/N}{1+\sqrt{n/N}})^2\right) = 8(\sqrt{2}-1)^2 > 1.37.$

Corollary 19. *An upper bound on the VOI estimate Λ_i^b is obtained by substituting Equation (8) into (7).*

$$\Lambda_\alpha^b \leq \hat{\Lambda}_\alpha^b = \frac{2N\overline{X}_\beta^{n_\beta}}{n_\alpha} \exp\left(-\varphi(\overline{X}_\alpha^{n_\alpha} - \overline{X}_\beta^{n_\beta})^2 n_\alpha\right) \quad (9)$$
$$\Lambda_{i|i\neq\alpha}^b \leq \hat{\Lambda}_i^b = \frac{2N(1-\overline{X}_\alpha^{n_\alpha})}{n_i} \exp\left(-\varphi(\overline{X}_\alpha^{n_\alpha} - \overline{X}_i^{n_i})^2 n_i\right)$$

More refined bounds can be obtained through tighter estimates on the probabilities in Equation (7), for example, based on the empirical Bernstein inequality (Maurer and Pontil, 2009), or through a more careful application of the Hoeffding inequality, resulting in:

$$\Lambda_i^b \leq \frac{N\sqrt{\pi}}{n_i\sqrt{n_i}} \left[\mathrm{erf}\left((1-\overline{X}_i^{n_i})\sqrt{n_i}\right) - \mathrm{erf}\left((\overline{X}_\alpha^{n_\alpha} - \overline{X}_i^{n_i})\sqrt{n_i}\right)\right]$$
$$\Lambda_\alpha^b \leq \frac{N\sqrt{\pi}}{n_\alpha\sqrt{n_\alpha}} \left[\mathrm{erf}\left(\overline{X}_\alpha^{n_\alpha}\sqrt{n_\alpha}\right) - \mathrm{erf}\left((\overline{X}_\alpha^{n_\alpha} - \overline{X}_\beta^{n_\beta})\sqrt{n_\alpha}\right)\right] \quad (10)$$

Selection problems usually separate out the decision of *whether* to sample or to stop (called the stopping policy), and *what* to sample. We'll examine the first issue here, along with the empirical evaluation of the above approximate algorithms, and the second in the following section.

Assuming that the sample costs are constant, a semi-myopic policy will decide to test the arm that has the best current VOI estimate. When the distributions are unknown, it makes sense to use the upper bounds established in Theorem 17, as we do in the following. This evaluation assumes a fixed budget of samples, which is completely used up by each of the candidate schemes, making a stopping criterion irrelevant.

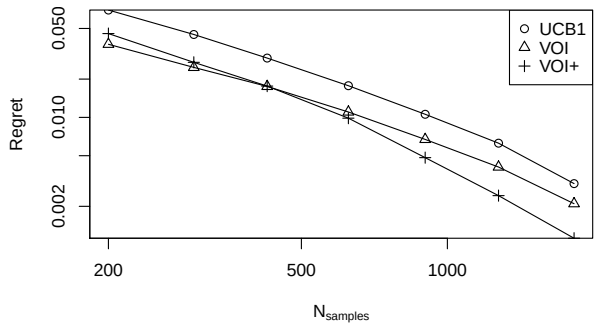


Figure 3: Average regret of various policies as a function of the fixed number of samples in a 25-action Bernoulli sampling problem, over 10000 trials.

The sampling policies are compared on random Bernoulli selection problem instances. Figure 3 shows results for randomly-generated selection problems with 25 Bernoulli arms, where the mean rewards of the arms are distributed uniformly in $[0,1]$, for a range of sample budgets 200..2000, with multiplicative step of 2, averaging over 10000 trials. We compare UCB1 with the policies based on the bounds in Equation (9) (VOI) and Equation (10) (VOI+). UCB1 is always considerably worse than the VOI-aware sampling policies.

6 Sampling in trees

The previous section addressed the selection problem in the flat case. Selection in trees is more complicated. The goal of Monte-Carlo tree search (Chaslot

et al., 2008) at the root node is usually to select an action that appears to be the best based on outcomes of *search rollouts*. But the goal of rollouts at non-root nodes is different than at the root: here it is important to better approximate the value of the node, so that selection at the root can be more informed. The exact analysis of sampling at internal nodes is outside the scope of this paper. At present we have no better proposal for internal nodes than to use UCT there.

We thus propose the following hybrid sampling scheme (Tolpin and Shimony, 2012a): at the *root node*, sample based on the VOI estimate; at *non-root nodes*, sample using UCT.

Strictly speaking, even at the root node the stationarity assumptions[6] underlying our belief-state MDP for selection do not hold exactly. UCT is an adaptive scheme, and therefore the values generated by sampling at non-root nodes will typically cause values observed at children of the root node to be non-stationary. Nevertheless, sampling based on VOI estimates computed as for stationary distributions works well in practice. As illustrated by the empirical evaluation (Section 6), estimates based on upper bounds on the VOI result in good sampling policies, which exhibit performance comparable to the performance of some state-of-the-art heuristic algorithms.

6.1 Stopping criterion

When a sample has a known cost commensurable with the value of information of a measurement, an upper bound on the intrinsic VOI can also be used to stop the sampling if the intrinsic VOI of any arm is less than the total cost of sampling C: $\max_i \Lambda_i \leq C$.

The VOI estimates of Equations (7) and (9) include the remaining sample budget N as a factor, but given the cost of a single sample c, the cost of the remaining samples accounted for in estimating the intrinsic VOI is $C = cN$. N can be dropped on both sides of the inequality, giving a reasonable stopping criterion:

$$\begin{aligned} \frac{1}{N}\Lambda^b_\alpha \leq & \frac{\overline{X}^{n_\beta}_\beta}{n_\alpha} \Pr(\overline{X}^{n_\alpha+N}_\alpha \leq \overline{X}^{n_\alpha}_\beta) \leq c \\ \frac{1}{N}\max_i \Lambda^b_i \leq & \max_i \frac{(1-\overline{X}^{n_\alpha}_\alpha)}{n_i} \Pr(\overline{X}^{n_i+N}_i \geq \overline{X}^{n_\alpha}_\alpha) \leq c \\ & \forall i : i \neq \alpha \end{aligned} \tag{11}$$

The empirical evaluation (Section 6) confirms the viability of this stopping criterion and illustrates the influence of the sample cost c on the performance of the sampling policy. When the sample cost c is unknown, one can perform initial calibration experiments to determine a reasonable value, as done in the following.

6.2 Sample redistribution in trees

The above hybrid approach assumes that the information obtained from rollouts in the current state is discarded after an real-world action is selected. In practice, many successful Monte-Carlo tree search algorithms reuse rollouts generated at earlier search states, if the sample traverses the current search state during the rollout; thus, the value of information of a rollout is determined not just by the influence on the choice of the action at the current state, but also by its potential influence on the choice at future search states.

One way to account for this reuse would be to incorporate the 'future' value of information into a VOI estimate. However, this approach requires a nontrivial extension of the theory of metareasoning for search. Alternately, one can behave myopically with respect to the search tree depth:

1. Estimate VOI as though the information is discarded after each step,
2. Stop early if the VOI is below a certain threshold (see Section 6.1), and
3. Save the unused sample budget for search in future states, such that if the nominal budget is N, and the unused budget in the last state is N_u, the search budget in the next state will be $N + N_u$.

In this approach, the cost c of a sample in the current state is the VOI of increasing the budget of a future state by one sample. It is unclear whether this cost can be accurately estimated, but supposing a fixed value for a given problem type and algorithm implementation would work. Indeed, the empirical evaluation (Section 6.3) confirms that stopping and sample redistribution based on a learned fixed cost substantially improve the performance of the VOI-based sampling policy in game tree search.

6.3 Playing Go against UCT

The hybrid policies were compared on the game Go, a search domain in which UCT-based MCTS has been particularly successful (Gelly and Wang, 2006). A modified version of Pachi (Braudiš and Loup Gailly, 2011), a state of the art Go program, was used for the experiments:

- The UCT engine of Pachi was extended with VOI-aware sampling policies at the first step.

[6] This is not a restriction, however, of the general formalism in Section 2.

- The stopping criterion for the VOI-aware policy was modified and based solely on the sample cost, specified as a constant parameter. The heuristic stopping criterion for the original UCT policy was left unchanged.
- The time-allocation model based on the fixed number of samples was modified for *both the original UCT policy and the VOI-aware policies* such that
  - Initially, the same number of samples is available to the agent at each step, independently of the number of pre-simulated games;
  - If samples were unused at the current step, they become available at the next step.

While the UCT engine is not the most powerful engine of Pachi, it is still a strong player. On the other hand, additional features of more advanced engines would obstruct the MCTS phenomena which are the subject of the experiment. The engines were compared on the 9x9 board, for 5000, 7000, 1000, and 15000 samples (game simulations) per ply, each experiment repeated 1000 times. Figure 4 depicts a calibration experiment, showing the winning rate of the VOI-aware policy against UCT as a function of the stopping threshold c (if the maximum VOI of a sample is below the threshold, the simulation is stopped, and a move is chosen). Each curve in the figure corresponds to a certain number of samples per ply. For the stopping threshold of 10^{-6}, the VOI-aware policy is almost always better than UCT, and reaches the winning rate of 64% for 10000 samples per ply.

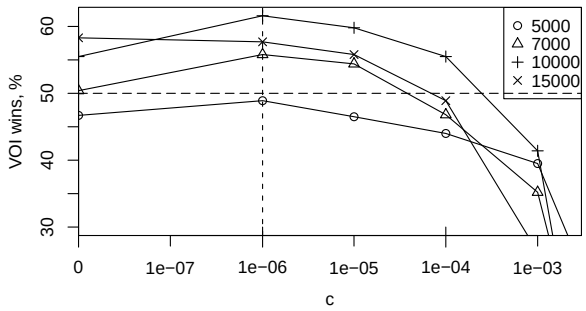


Figure 4: Winning rate of the VOI-aware policy in Go as a function of the cost c, for varying numbers of samples per ply.

Figure 5 shows the winning rate of VOI against UCT $c = 10^{-6}$. In agreement with the intuition (Figure 6.2), VOI-based stopping and sample redistribution is most influential for intermediate numbers of samples per ply. When the maximum number of samples is too low, early stopping would result in poorly selected moves. On the other hand, when the maximum number of samples is sufficiently high, the VOI of increasing the maximum number of samples in a future state is low.

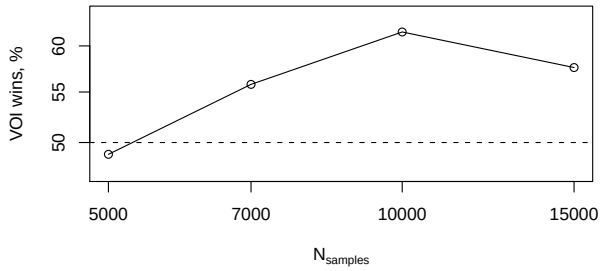


Figure 5: Winning rate of the VOI-aware policy in Go as a function of the number of samples, fixing cost $c = 10^{-6}$.

Note that if we disallowed reuse of samples in both Pachi and in our VOI-based scheme, the VOI based-scheme win rate is even higher than shown in Figure 5. This is as expected, as this setting (which is somewhat unfair to Pachi) is closer to meeting the assumptions underlying the selection MDP.

7 Conclusion

The selection problem has numerous applications. This paper formalized the problem as a belief-state MDP and proved some important properties of the resulting formalism. An application of the selection problem to control of sampling was examined, and the insights provided by properties of the MDP led to approximate solutions that improve the state of the art. This was shown in empirical evaluation both in "flat" selection and when extending the methods to game-tree search for the game of Go.

The methods proposed in the paper open up several new research directions. The first is a better approximate solution of the MDP, that should lead to even better flat sampling algorithms for selection. A more ambitious goal is extending the formalism to trees—in particular, achieving better sampling at non-root nodes, for which the purpose of sampling differs from that at the root.

Acknowledgments

The research is partially supported by Israel Science Foundation grant 305/09, National Science Foundation grant IIS-0904672, DARPA DSO FA8650-11-1-7153, the Lynne and William Frankel Center for Computer Sciences, and by the Paul Ivanier Center for Robotics Research and Production Management.

References

J. Audibert, S. Bubeck, et al. Best arm identification in multi-armed bandits. In *COLT*, 2010.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 2002.

R. Bechhofer. A single-sample multiple decision procedure for ranking means of normal populations with known variances. *The Annals of Mathematical Statistics*, 25(1):16–39, 1954.

D. Berry and B. Fristedt. *Bandit problems: sequential allocation of experiments*. Chapman and Hall London, 1985.

P. Braudiš and J. Loup Gailly. Pachi: State of the art open source Go program. In *ACG 13*, 2011.

S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in finitely-armed and continuous-armed bandits. *Theor. Comput. Sci.*, 412(19):1832–1852, 2011.

G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-Carlo tree search: A new framework for game AI. In *AIIDE*, 2008.

T. Dean and M. Boddy. An analysis of time-dependent planning. In *AAAI-88*, pages 49–54, 1988.

J. Doyle. Artificial intelligence and rational self-government. Technical Report CMU-CS-88-124, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 1988.

M. Fehling and J. Breese. A computational model for the decision-theoretic control of problem solving under uncertainty. In *Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence, Minneapolis, MN: AAAI*, 1988.

P. Frazier and W. Powell. Paradoxes in learning and the marginal value of information. *Decision Analysis*, 2010.

S. Gelly and D. Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 2011.

S. Gelly and Y. Wang. Exploration exploitation in Go: UCT for Monte-Carlo Go. *Computer*, 2006.

J. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

J. Gittins. *Multi-armed bandit allocation indices*. John Wiley & Sons, 1989.

I. J. Good. A five-year plan for automatic chess. In E. Dale and D. Michie, editors, *Machine Intelligence 2*, pages 89–118. Oliver and Boyd, 1968.

D. Harada. Reinforcement learning with time. In *Proc. Fourteenth National Conference on Artificial Intelligence*, pages 577–582. AAAI Press, 1997.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):pp. 13–30, 1963. ISSN 01621459.

E. J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *UAI*. American Association for Artificial Intelligence, July 1987. Also in L. Kanal, T. Levitt, and J. Lemmer, ed., *Uncertainty in Artificial Intelligence 3,* Elsevier, 1988, pps. 301-324.

R. A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 1966.

L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. *ECML*, 2006.

O. Madani, D. Lizotte, and R. Greiner. Active model selection. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 357–365. AUAI Press, 2004.

J. E. Matheson. The economic value of analysis and computation. *Systems Science and Cybernetics*, 4: 325–332, 1968.

A. Maurer and M. Pontil. Empirical Bernstein bounds and sample-variance penalization. In *COLT*, 2009.

M. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 1994.

S. Russell and E. Wefald. *Do The Right Thing*. The MIT Press, 1991a.

S. Russell and E. Wefald. Principles of metareasoning. *Artificial Intelligence*, 1991b.

S. J. Russell and E. H. Wefald. Decision-theoretic control of search: General theory and an application to game-playing. Technical Report UCB/CSD 88/435, Computer Science Division, University of California, Berkeley, 1988.

J. R. Swisher, S. H. Jacobson, and E. Yücesan. Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. *ACM Transactions on Modeling and Computer Simulation*, 2003.

D. Tolpin and S. E. Shimony. MCTS based on simple regret. In *AAAI*. AAAI Press, 2012a. To appear.

D. Tolpin and S. E. Shimony. Semimyopic measurement selection for optimization under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 42(2):565–579, 2012b.

A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16:117–186, 1945.

Tightening Fractional Covering Upper Bounds on the Partition Function for High-Order Region Graphs

Tamir Hazan
TTI Chicago
Chicago, IL 60637

Jian Peng
TTI Chicago
Chicago, IL 60637

Amnon Shashua
The Hebrew University
Jerusalem, Israel

Abstract

In this paper we present a new approach for tightening upper bounds on the partition function. Our upper bounds are based on fractional covering bounds on the entropy function, and result in a concave program to compute these bounds and a convex program to tighten them. To solve these programs effectively for general region graphs we utilize the entropy barrier method, thus decomposing the original programs by their dual programs and solve them with dual block optimization scheme. The entropy barrier method provides an elegant framework to generalize the message-passing scheme to high-order region graph, as well as to solve the block dual steps in closed-form. This is a key for computational relevancy for large problems with thousands of regions.

1 Introduction

A large set of reasoning problems can be framed as a set of dependency relations among possible structures. These dependencies, usually expressed by a graph, define a joint probability function which drives an inference engine over those structures [18]. Such graphical models, also known as Markov Random Fields (MRFs), are found in a wide variety of fields and applications, including object detection [5], stereo vision [30], parsing [19], or protein design [29], as well as other broad disciplines which include artificial intelligence, signal processing and statistical physics.

The inference problem in MRFs involves assessing the likelihood of possible structures, whether objects, parsers, or molecular structures. The structures are specified by assignments of random variables, whose scores are described in a concise manner by interactions over small subsets of variables. In a fully probabilistic treatment, all possible alternative assignments are considered. However, this requires summing over the assignments with their respective weights – evaluating the partition function. The partition function has a special role which goes beyond that of assigning a probability to the alternative assignments. In addition it plays a fundamental role in various contexts including approximate inference [26], maximum-likelihood parameter estimation [20] and large deviation bounds [4]. Computing the partition function requires summing over the assignments with their respective weights, thus it is a #P hard problem (i.e., efficient weighted counting is unlikely achievable) [31]. Instead there is much focus on finding approximate solutions and bounds [34].

In this paper we propose an iteration of concave and convex programs, each using a primal-dual iterative scheme, to compute increasingly tight upper bounds on the partition function that are based on fractional covering bounds on the entropy [6, 23]. Our work is distinct on a couple of fronts: we handle general region graphical models, unlike previous attempts that focus on particular graphs like bipartite forms (outer and inner regions) or on limited size factors (like pairwise). Secondly, our method is based on achieving a message-passing constellation between nodes of the region graph thereby utilizing the local structure of the problem. A message-passing framework, or more generally the dual decomposition framework, is key for computational relevancy for large problems which contain scores of thousands of regions where the factors are defined on a relatively small number of variables (e.g., when variables correspond to a large grid of points).

We begin by introducing the notation and the fractional covering upper bounds of interest. We subsequently present a message-passing algorithm to compute the upper bound, using dual decomposition. We also introduce a new approach to tighten the fractional covering upper bounds based on the entropy barrier

function and dual block optimization, and demonstrate the effectiveness of the approach.

2 Notations, Problem Setup and Background

Let $x = (x_1, ..., x_n)$ be the realizations of n discrete random variables where the range of the $i'th$ random variable is $\{1, ..., n_i\}$, i.e., $x_i \in \{1, ..., n_i\}$. We consider a joint distribution $p(x_1, ..., x_n)$ and assume that it factors into a product of non-negative functions $\psi_r(x_r)$, known as *potentials*. Usually, the potentials are defined over a small subset of indexes $r \subset \{1, ..., n\}$, called *regions*:

$$p(x_1, ..., x_n) = \frac{1}{Z} \prod_{r \in \mathcal{R}} \psi_r(x_r)$$

where for singletons, i.e. $|r| = 1$, the functions $\psi_r(x_r)$ represent "local evidence" or prior data on the states of x_i, and for $|r| > 1$ the potential functions describe the interactions of their variables $x_r \subset \{x_1, ..., x_n\}$, and Z is the normalization constant, also called the *partition function*. For convenience, we adopt the additive form by setting $\phi_r(x_r) = \ln \psi_r(x_r)$ thereby having the joint probability take the form of a Gibbs distribution:

$$p(x_1, , x_n) = e^{\sum_{r \in \mathcal{R}} \phi_r(x_r) - \ln Z}$$

For example, $p(x_1, x_2, x_3) \propto \exp(\phi_2(x_2) + \phi_{123}(x_1, x_2, x_3) + \phi_{23}(x_2, x_3))$ has three factors with regions $\mathcal{R} = \{2\}, \{1, 2, 3\}, \{2, 3\}$. The factorization structure above defines a hypergraph whose nodes represent the n random variables, and the regions $r \in \mathcal{R}$ correspond to its hyperedges. A convenient way to represent this hypergraph is by a *region graph*. A region graph is a directed graph whose nodes represent the regions and its direct edges correspond to the inclusion relation, i.e., a directed edge from node r to s is possible only if $s \subset r$. We adopt the terminology where $P(r)$ and $C(r)$ stand for all nodes that are parents and children of the node r, respectively. Also, we define R_i to be the set of regions which contains the variable i.

Inference is closely coupled with the ability to evaluate the logarithm of the partition function $\ln Z$. From a variational perspective, there is a relationship between the (minus) Gibbs-Helmholtz free-energy and $\ln Z$:

$$\ln Z = \max_{\substack{p(x) \geq 0, \\ \sum_x p(x) = 1}} \Big\{ \sum_{r \in \mathcal{R}} \sum_{x_r} p(x_r) \phi_r(x_r) + H(p) \Big\} \quad (1)$$

where $p(x_r) = \sum_{x \setminus x_r} p(x)$ is the marginal probability and $H(p) = -\sum_x p(x) \ln p(x)$ is the entropy of the distribution. However, the complexity of the variational representation is unwieldy because both the entropy and the simplex constraint require an evaluation over all possible states of the system $x = (x_1, ..., x_n)$, which is exponential in n. Instead one looks for an approximation or bounds. An upper bound is designed with a tractable approximation of the free-energy by (i) upper bounding the entropy term $H(p)$ by a combination of local entropies over marginal probabilities $p(x_r)$, and (ii) by outer bounding the probability simplex constraints by the so called "local consistency" constraints.

An upper bound on the entropy function $H(p)$ proceeds by replacing the entropy by the fractional covering entropy bounds [6, 23]. These upper bounds are defined as sum of local entropies over the marginal probabilities $H(p(x_r)) = -\sum_{x_r} p(x_r) \ln p(x_r)$:

$$H(p) \leq \sum_{r \in \mathcal{R}} c_r H(p(x_r))$$

These upper bounds hold whenever $c_r \geq 0$ and for every $i = 1, ..., n$ there holds $\sum_{r \in R_i} c_r = 1$, where R_i is the set of all regions that contain i. The second step in obtaining an efficient upper bound is replacing the marginal distributions $p(x_r)$ by "beliefs" $b_r(x_r)$. The beliefs form "pseudo distributions" in the sense that the beliefs might not necessarily arise as marginal probabilities of some distribution $p(x_1, ..., x_n)$. To maintain local consistency between beliefs which share the same variables, we define the *local consistency polytope* $L(G)$ for the region graph G, as follows:

$$L(G) = \left\{ \{b_r\}_{r \in \mathcal{R}} : \begin{array}{ll} \sum\limits_{x_s \setminus x_r} b_s(x_s) = b_r(x_r) & \forall r, s \in P(r) \\ b_r \geq 0, \ \sum_{x_r} b_r(x_r) = 1 & \forall r \in \mathcal{R} \end{array} \right.$$

For example, assume $\mathcal{R}$ consists of the three factors $\{1\}, \{1, 2\}, \{1, 3\}$ then the consistency constraints on the beliefs $b_1(x_1), b_{1,2}(x_1, x_2), b_{1,3}(x_1, x_3)$ enforce their distribution constraints and marginalization constraints $\sum_{x_3} b_{1,3}(x_1, x_3) = b_1(x_1)$ and $\sum_{x_2} b_{1,2}(x_1, x_2) = b_1(x_1)$.

Taken together, the upper bound on the log-partition is defined as follows:

$$\ln Z \leq \max_{b_r(x_r) \in L(G)} \Big\{ \sum_{r, x_r} b_r(x_r) \phi_r(x_r) + \sum_r c_r H(b_r) \Big\} \quad (2)$$

Thus we introduce a family of upper bounds for the partition function, for the set of fractional covering numbers c_r. The computational complexity of these upper bounds is no longer exponential in n, but linear in the number of the regions and the number of assignments in each region.

Computing the fractional covering upper bound: Given potential functions $\phi_r(x_r)$, and nonnegative covering numbers c_r. Set $c_{p,r} = c_p/(c_r + \sum_{p' \in P(r)} c_{p'})$. for every initial values of messages $\lambda_{c \to p}(x_c)$:

1. For $t = 1, 2, ...$

   (a) For $r \in \mathcal{R}$ do:

$$\forall x_r, \forall p \in P(r) \quad \mu_{p \to r}(x_r) = c_p \log \Big(\sum_{x_p \setminus x_r} \exp \Big((\phi_p(x_p) + \sum_{c \in C(p) \setminus r} \lambda_{c \to p}(x_c) - \sum_{p' \in P(p)} \lambda_{p \to p'}(x_p)) \Big/ c_p \Big) \Big)$$

$$\forall x_r, \forall p \in P(r) \quad \lambda_{r \to p}(x_r) = c_{p,r} \Big(\phi_r(x_r) + \sum_{c \in C(r)} \lambda_{c \to r}(x_c) + \sum_{p' \in P(r)} \mu_{p' \to r}(x_r) \Big) - \mu_{p \to r}(x_r)$$

2. Output:

$$\text{(beliefs)} \quad \forall r \in \mathcal{R} \quad c_r \neq 0 : b^*_r(x_r) \propto \exp \Big((\phi_r(x_r) + \sum_{c \in C(r)} \lambda_{c \to r}(x_c) - \sum_{p \in P(r)} \lambda_{r \to p}(x_r)) \Big/ c_r \Big)$$

$$\forall r \in \mathcal{R} \quad c_r = 0 : support(b^*_r) \subset argmax_{x_r} \{ \phi_r(x_r) + \sum_{c \in C(r)} \lambda_{c \to r}(x_c) - \sum_{p \in P(r)} \lambda_{r \to p}(x_r) \}$$

$$\text{(bound)} \quad \sum_{r, x_r} b^*_r(x_r) \phi_r(x_r) + \sum_r c_r H(b^*_r)$$

Figure 1: *The fractional covering upper bound appears in equation (2). The support of the beliefs are their non-zero entries, and when $c_r = 0$ it corresponds to the max-beliefs. When considering bipartite region graphs, this algorithm reduces to many of the previous message-passing algorithms, see Section 6.*

3 Computing High-Order Upper Bounds

In the following we develop an efficient message-passing method to compute the region based upper bounds and their optimal beliefs $b_r(x_r)$, for fixed value of covering numbers c_r, as described in equation (2). These upper bounds depend on the non-negative fractional covering numbers, therefore correspond to maximizing a concave function subject to convex constraints. Such concave programs can be solved by minimizing their dual convex programs. Nevertheless, there are potentially many different convex dual programs, depending on the set of constraints, or Lagrange multipliers, one aims at satisfying. We realize that the probability simplex constraints for $b_r(x_r)$ are easier to satisfy, therefore we derive a dual program which ignores these constraints. For this purpose we use the entropy function as a barrier function over the probability simplex.

Theorem 1. *Define the entropy as a barrier function over the probability simplex,*

$$H(b_r) = \begin{cases} -\sum_{x_r} b_r(x_r) \log b_r(x_r) & if \;\; b_r \in \Delta \\ -\infty & otherwise \end{cases}$$

where $b_r \in \Delta$ if $b_r(x_r) \geq 0$ and $\sum_{x_r} b_r(x_r) = 1$. The fractional covering numbers in equation (2) are non-negative, thus the bound is a concave function and its dual function takes the form

$$D(\lambda) = \sum_r c_r \log \Big(\sum_{x_r} \exp(\hat{\phi}_r(x_r)/c_r) \Big)$$

where

$$\hat{\phi}_r(x_r) = \phi_r(x_r) + \sum_{c \in C(r)} \lambda_{c \to r}(x_c) - \sum_{p \in P(r)} \lambda_{r \to p}(x_r)$$

In particular, strong duality holds and the primal optimal solution can be derived from the dual optimal solution

$$b_r(x_r) \propto \exp \Big(\hat{\phi}_r(x_r) \Big/ c_r \Big)$$

Whenever $c_r = 0$ the corresponding primal optimal solution $b_r(x_r)$ corresponds to the max-beliefs, i.e., probability distributions over the maximal arguments of $\hat{\phi}_r(x_r)$.

Proof: Since we use the entropy as a barrier function over the probability simplex we only need to apply Lagrange multipliers $\lambda_{r \to p}(x_r)$ for the marginalization constraints $b_r(x_r) = \sum_{x_p \setminus x_r} b_p(x_p)$ for every region $r \in \mathcal{R}$, every assignment x_r and every parent $p \in P(r)$. Therefore the Lagrangian takes the form

$$L(b_r, \lambda_{c \to p}) = \sum_{r, x_r} b_r(x_r) \phi_r(x_r) + \sum_r c_r H(b_r)$$
$$+ \sum_{r,\, x_r,\, p \in P(r)} \lambda_{r \to p}(x_r) \Big(\sum_{x_p \setminus x_r} b_p(x_p) - b_r(x_r) \Big)$$

The dual function is recovered by maximizing the beliefs $b_r(x_r)$ in the Lagrangian. Thus

$$\begin{aligned} D(\lambda) &= \sum_r \max_{b_r} \{\sum_{x_r} b_r(x_r)\hat{\phi}_r(x_r) + c_r H(b_r)\} \\ &= \sum_r c_r \max_{b_r} \{\sum_{x_r} b_r(x_r)(\hat{\phi}_r(x_r)/c_r) + H(b_r)\} \end{aligned}$$

The maximization over the beliefs in the Lagrangian is done while satisfying the probability simplex constraint as they are encoded in the domain of the entropy function. The result follows from the duality between the entropy barrier function over the probability simplex and the log-partition function, c.f. equation (1). Strong duality holds using Theorem 6.2.5 in [1]. The primal optimal solution is derived from the dual optimal solution as the primal arguments that maximize the Lagrangian. □

The theorem above uses the conjugate duality between the entropy barrier function, weighted by the non-negative number c_r, and the weighted extension of the log-partition. The weighted log-partition is called the soft-max function and it is used as a smooth approximation for the max function whenever $c_r \to 0$. In particular, when $c_r = 0$ the soft-max function reduces to the max-function, and Theorem 1 demonstrates the known conjugate duality between the indicator function over the probability simplex and the max-function.

One of the most important properties of the dual function is that it decomposes the constraints set, a feature that is typically called dual decomposition. In Theorem 1 every dual variable $\lambda_{c\to p}(x_c)$ represents a marginalization constraint $\sum_{x_p \setminus x_c} b_p(x_p) = b_c(x_c)$ in the primal. Therefore optimizing a single dual variable amounts to solving the primal problem with a single constraint. Therefore, the dual coordinate descent algorithm decomposes the primal problem complexity to smaller sub-problems, while the dual function encodes the consistency between the sub-problems solutions. Moreover, since the marginalization constraints encode the graphical model, performing block coordinate descent results in sending messages along the edges of the region graph, thus we are able to cope with large-scale and high-order graphical models.

Theorem 2. *Block coordinate descent on the dual in Theorem 1 takes the form: For every region $r \in \mathcal{R}$*

$$\forall x_r, \forall p \in P(r)$$

$$\mu_{p\to r}(x_r) = c_p \log\Big(\sum_{x_p \setminus x_r} \exp(\phi_{p,r}(x_p)/c_p)\Big)$$

$$\lambda_{r\to p}(x_r) = c_{p,r}\Big(\phi_r(x_r) + \sum_{c\in C(r)} \lambda_{c\to r}(x_c) + \sum_{p'\in P(r)} \mu_{p'\to r}(x_r)\Big) - \mu_{p\to r}(x_r)$$

where $c_{p,r} = c_p/(c_r + \sum_{p'\in P(r)} c_{p'})$ and $\phi_{p,r}(x_p) = \phi_p(x_p) + \sum_{c\in C(p)\setminus r} \lambda_{c\to p}(x_c) - \sum_{p'\in P(p)} \lambda_{p\to p'}(x_p)$. If a region covering number in the dual objective equals zero, i.e., $c_r = 0$, then the block coordinate descent update rules for this region hold for every non-negative c_r.

Proof: The gradient equals to

$$\frac{\partial D}{\partial \lambda_{r\to p}(x_r)} = \sum_{x_p \setminus x_r} b_p(x_p) - b_r(x_r)$$

where the probability distributions $b_r(x_r), b_p(x_p)$ are proportional to $\exp(\hat{\phi}(x_r)/c_r)$ and $\exp(\hat{\phi}(x_p)/c_p)$ respectively while $\hat{\phi}$ is defined in Theorem 1. Whenever c_r or c_p are zero, their respective (sub)gradients are their max-beliefs or equivalently probability distributions over their maximal arguments of $\hat{\phi}$, c.f., Danskin theorem [1].

The optimal dual variables are those for which $\partial D/\partial \lambda_{r\to p}(x_r) = 0$, i.e., the corresponding beliefs agree on their marginal probabilities. When setting $\mu_{p\to r}(x_r)$ as above, the marginalization of $b_p(x_p)$ equals

$$\sum_{x_p \setminus x_r} b_p(x_p) \propto \exp\Big((\mu_{p\to r}(x_r) + \lambda_{r\to p}(x_r))/c_p\Big)$$

Therefore, by taking the logarithm, the gradient vanishes whenever the beliefs numerators agree up to an additive constant

$$\frac{\mu_{p\to r}(x_r) + \lambda_{r\to p}(x_r)}{c_p} = \frac{\phi'_r(x_r) - \sum_{p\in P(r)} \lambda_{r\to p}(x_r)}{c_r} \quad (3)$$

where $\phi'_r(x_r) = \phi_r(x_r) + \sum_{c\in C(r)} \lambda_{c\to r}(x_c)$. Multiplying both sides by $c_r c_p$ and summing both sides with respect to $p' \in P(r)$ we are able to obtain

$$\sum_{p'\in P(r)} \lambda_{r\to p'}(x_r) = c_{p,r}\Big(\phi'_r(x_r) + \sum_{p'\in P(r)} \mu_{p'\to r}(x_r))\Big)$$

Plugin it into equation (3) results in the desired block dual descent update rule, i.e., $\lambda_{r\to p}(x_r)$ for which the partial derivatives vanish. □

For convenience, the explicit update rules also appear in Fig. 1. The above theorem provides a closed-form solution for block coordinate descent on the dual objective. Since the dual values are always lower bounded by the primal values, this algorithm is guaranteed to converge. However, the dual objective is not strictly convex therefore, a-priori, coordinate descent might not converge to the optimal solution [27]. Nevertheless, our proof technique demonstrates the dual-primal relation of dual coordinate descent, as throughout its runtime it generates a sequence of primal variables,

namely the beliefs, which agree on their marginal probabilities. In the following we describe the conditions for which we can recover the primal optimal and dual optimal solutions.

Theorem 3. *The dual block descent algorithm in Theorem 2 is guaranteed to converge whenever c_r are non-negative. Moreover, if $c_r > 0$ the dual block descent algorithm converges to the dual optimal value, and the beliefs $b_r(x_r)$ that are generated throughout the algorithm runtime converge to the primal optimal solution ; whenever its dual sequence is bounded every of its limit points is an optimal dual solution.*

Proof: See supplementary material. □

The above theorem does not constrain the dual variables, namely the messages. Specifically, although the dual value is guaranteed to converge to the optimum, the messages are not guaranteed to be bounded. This may happen since the dual function is not strictly convex, and the messages can be updated along an unbounded plateau that does not reduce the dual objective. The above result can be extended to guarantee optimality even if $c_r > 0$ only for the maximal regions, i.e., regions that are not contained by other regions. However, this result is mathematically involved as it does not correspond anymore to Bregman divergences and it is beyond the scope of this work.

The algorithm in Theorem 2 has an important meaning even when the primal function is not concave. In this case we lose all convergence guarantees, but the algorithm represents the Lagrangian saddle points. Therefore, whenever the algorithm converges it reaches a local maximum of the primal program.

Theorem 4. *Assume the program in equation (2) has mixed numbers $c_r \gtrless 0$. Then this program is not concave and the algorithm in Theorem 2 is not guaranteed to converge, but whenever it converges it reaches a stationary point for this program.*

Proof: See Supplementary material. □

The saddle point theorem does not consider programs for which the covering numbers can be zero, since in these cases we cannot uniquely restrict the beliefs. Nevertheless, this theorem applies to many important cases, such as the Bethe entropy and the tree reweighed entropies. In these cases, the negative coefficients for variables $c_i = 1 - \sum_{\alpha \supset i} c_\alpha$, where α are the non-singleton regions, play an important role. Specifically, the Bethe entropy provides an exact characterization for the entropy function over tree models, and the tree-reweighted entropy provides tighter bounds than the fractional covering bounds when restricting the fractional coverings to pairwise entropies. Unfortunately, the message-passing algorithm is not guaranteed to converge in these cases but the theorem above proves that when it converges it reaches a (local) maximum of the model.

4 Tightening High-Order Upper Bounds

Up to this point, the fractional covering numbers c_r were held fixed, while we computed the upper bound for the partition function that is described in equation (2). We now consider how to find the optimal covering numbers which minimize the upper bound, while assuming we are able to compute the upper bound and its optimal arguments efficiently as described in Fig. 1. First we state some of the properties of these upper bounds as a function of their fractional covering numbers:

Theorem 5. (Danskin Theorem) *Denote the log-partition upper bound by*

$$U(c) = \max_{b_r(x_r) \in L(G)} \Big\{ \sum_{r,x_r} b_r(x_r)\phi_r(x_r) + \sum_r c_r H(b_r) \Big\}$$

Assume $c_r \geq 0$ and that the maximal regions covering numbers are positive. Then $U(c)$ is a convex and differentiable function, and its partial derivatives are

$$\frac{\partial U}{\partial c_r} = H(b_r^*)$$

where

$$b_r^*(x_r) = \underset{b_r(x_r) \in L(G)}{argmax} \Big\{ \sum_{r,x_r} b_r(x_r)\phi_r(x_r) + \sum_r c_r H(b_r) \Big\}$$

Proof: This is a special case of Danskin theorem, since $L(G)$ is a compact set. [1] □

To find the minimal upper bound on the partition function we need to minimize a convex and differentiable function over the convex set of fractional covering.

$$\min_{c_r \geq 0, \sum_{r \in R_i} c_r = 1} U(c)$$

Minimizing a convex function over a convex set is typically done using the conditional gradient method, for which one finds a direction of descent while respecting the fractional covering constraints [2]. Finding the direction of descent amounts to solving the linear program

$$d = \underset{c_r \geq 0, \sum_{r \in R_i} c_r = 1}{argmin} \sum_r c_r H(b_r^*) \tag{4}$$

The direction of descent is always attained, since it is the minimal argument of a continuous function over a compact set. However, when considering

Tightening the fractional covering upper bound: Given functions $\phi_i(r_i)$. For every initial values of $\lambda_i(r)$:

1. For $t = 1, 2, ...$
   (a) For $r \in \mathcal{R}$ do:

$$\forall i \in r \quad \lambda_i(r) = \epsilon \log \Big(\sum_{r_i \neq r} \exp \Big((\phi_i(r_i) + \lambda_i(r_i) \Big/ \epsilon \Big) \Big) - \phi_i(r)$$

$$\text{additively normalize } \lambda_i(r) \text{ such that } \quad \sum_{i \in r} \lambda_i(r) = 0$$

2. Output: $\forall i$, set $q_i(r_i) \propto \exp \Big((\phi_i(r_i) + \lambda_i(r_i)) \Big/ \epsilon \Big)$ and $\forall r$ set $c_r = q_i(r)$, for any of $i \in r$.

Figure 2: *The dual decomposition algorithm for recovering the direction of descent, described in equation (4). This algorithm can be seen as message-passing by reformulating the indexes $\lambda_i(r) \leftrightarrow \lambda_{i \to r}$. This demonstrates the differences between our two algorithms. The similarity between the two algorithms is a consequence of the block dual steps and the use of the entropy barrier method.*

large-scale problems, this linear program cannot be efficiently solved using standard linear programming solvers. For example, the simplex algorithms or interior point methods typically involve inverting the constraints matrix, an operation that cannot be done efficiently when dealing with graphical models that consist of millions of regions.

In this section we develop a new type of efficient linear programming solver for finding the minimal upper bound. Importantly, we cannot use the message-passing solvers in Section 3 to minimize these upper bounds, since the objective as well as the constraint sets in equation (4) are significantly different. However, we are able to construct efficient solvers while using the entropy barrier function in a similar manner. This emphasizes the computational importance of the entropy barrier function, since it results in closed-form update rules which turn to be crucial for large-scale linear programs.

The constraints of the linear program in equation (4) restrict the covering numbers $(c_r)_{r \in R_i}$ to the probability simplex. In order to use dual decomposition effectively, we need to decouple these simplex constraints, such that they restrict *separate* distributions. Therefore we inflate the covering numbers c_r to non-overlapping probability distribution $q_i(r_i)$ for every $i = 1, ..., n$, while enforcing these distributions to agree, namely $q_i(r) = q_r$ for every $i \in r$. One can verify that these constraints are equivalent to the constraints over c_r in equation (4). Thus we reformulate the linear program in equation (4) as

$$\underset{\substack{\forall i \; q_i \text{ is probability} \\ \forall i \in r \quad q_i(r) = q_r}}{\operatorname{argmin}} \; \sum_{i, r_i} q_i(r_i) \Big(H(b^*_{r_i}) / |r_i| \Big)$$

For computational efficiency we solve the above linear program by dual block ascent. Unfortunately, dual coordinate ascent may be sub-optimal when the dual program is not smooth, or equivalently the primal program is not strictly convex. Therefore we use the entropy barrier method to make the primal program strictly convex, which results in a smooth dual program. The following theorem shows that using the barrier method does not affect the quality of the solution.

Theorem 6. *Let $\phi_i(r_i) = H(b^*_{r_i})/|r_i|$ and consider the linear program for finding the direction of descent*

$$\min_{\substack{\forall i \; q_i \; is \; probability \\ \forall i \in r \quad q_i(r) = q_r}} \; \sum_{i, r_i} q_i(r_i) \phi_i(r_i)$$

Then the primal-dual programs

(*primal*)

$$\min_{\substack{\forall i \; q_i \; is \; probability \\ \forall i \in r \quad q_i(r) = q_r}} \; \sum_{i, r_i} q_i(r_i) \phi_i(r_i) - \epsilon \sum_i H(q_i)$$

(*dual*)

$$\max_{\sum_{i \in r} \lambda_i(r) = 0} \; \epsilon \sum_i \log \Big(\sum_{r_i} \exp \Big((\phi_i(r_i) + \lambda_i(r_i)) \Big/ \epsilon \Big) \Big)$$

are $\delta-$approximations of the original linear program, where $\delta = \sum_i \epsilon \log |r_i|$

Proof: The entropy $H(q_i)$ is a non-negative measure, thus the primal program lower bounds the original linear program. The bound holds since $H(q_i) \leq \log |r_i|$. The dual program is also a $\delta-$approximation since strong duality holds. □

Having a smooth dual program which relates to a strictly convex primal program, we can perform dual

block ascent to reach their optimum. Using the entropy barrier function enables us to derive a closed-form update rules.

Theorem 7. *Consider the primal and dual programs in Theorem 6. Then the block dual ascent takes the form*

$$\forall r,\ \ \forall i \in r$$
$$\lambda_i(r) = \epsilon \log\Big(\sum_{r_i \neq r} \exp\Big(\big(\phi_i(r_i) + \lambda_i(r_i)\big/\epsilon\Big)\Big) - \phi_i(r)$$

additively normalize $\lambda_i(r)$ *such that* $\sum_{i \in r} \lambda_i(r) = 0$

Also, block dual ascent is guaranteed to converge to the dual optimum, the probabilities $q_i(r_i)$ *that are generated throughout the algorithm runtime converge to the primal optimal point and whenever the dual sequence is bounded it is guaranteed to converge to an optimal dual solution.*

Proof: See supplementary material. □

For convenience, the algorithm appears in its explicit form in Fig. 2. The above block dual ascent algorithm can be seen as message-passing, where messages are sent between regions and variables, $\lambda_i(r) \leftrightarrow \lambda_{i \to r}$. This demonstrates the differences between our two algorithms, where the first sends two types of real-valued vectors $\lambda_{c \to p}(x_c), \mu_{p \to c}(x_c)$ along the edges of the regions graph, and the second sends a real number $\lambda_{i \to r}$ between variables and regions. The similarity between the two algorithms is a consequence of the block dual steps and the entropy barrier function.

5 Empirical Evaluation

In our experiments we first compared our message-passing algorithm for computing the fractional covering upper bound in equation (2) with the current state-of-the-art solver [12]. We compared these algorithms on the grid shape spin glass model. A spin glass model consists of n spins $x_1, ..., x_n \in \{-1, 1\}$, whose local potentials are $\phi_i(x_i) = \theta_i x_i$ and its pairwise potentials are $\phi_{i,j}(x_i, x_j) = \theta_{i,j} x_i x_j$. The field parameters θ_i were chosen uniformly at random from $[-0.05, 0.05]$ and the coupling parameters $\theta_{i,j}$ were chosen uniformly at random from $[-1, 1]$. We used the same covering numbers for all edges and squares in the grid, setting $c_{i,j} = c_{i,j,k,l} = 1/9$ and $c_i = 1 - \sum_{r \in R_i \setminus i} c_r$. Therefore we satisfy the covering number constraints, $c_r \geq 0$ and $\sum_{r \in R_i} c_r = 1$. Our algorithm described in Fig. 1 used a region graph which connects squares to pairs and pairs to singletons, namely the Hesse diagram. [12] use a bipartite inner-outer region graph, for which all outer regions, i.e., squares, are connected to all inner regions, i.e., pairs and singletons. We used the same Matlab code, on a single core of Intel I5 with 8GB RAM, for both algorithm as our algorithm can be applied to bipartite region graphs as well. We compared both methods on 5×5 ,..., 100×100 grids and the stopping criteria was a primal-dual gap of 10^{-4}. Fig. 3 shows that passing messages over the Hesse-diagram is better than working over the bipartite inner-outer region graph, and the gap between these methods is significant for large graphical models. We attribute this behavior to the fact that on the Hesse diagram we can pass many messages between pairs and singletons, while using time consuming square messages only when they are needed. In contrast [12] use square messages in every message-passing iteration.

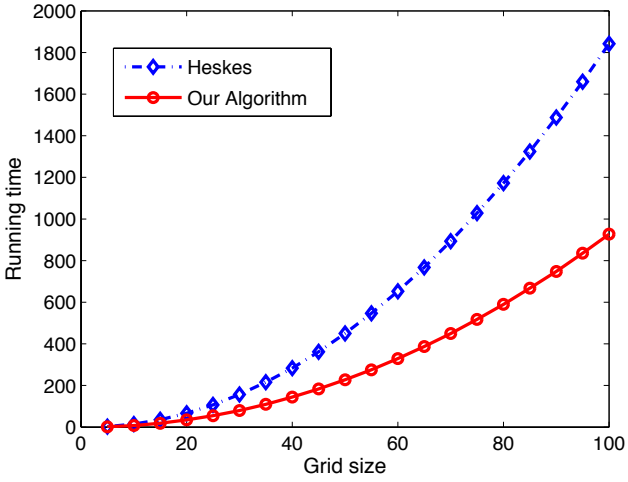


Figure 3: *Comparing our message-passing algorithm for computing the fractional covering upper bound in Section 3 with [12]. Comparison is for* 5×5*,...,* 100×100 *grid shape spin glass models, with singletons, pairs and squares regions. Our approach can utilize intermediate size message, between pairs and singletons, while [12] use the inner-outer region graph, thus sending square based messages in every iteration.*

In our experiments we also compared our dual decomposition algorithm for recovering the direction of descent over the fractional covering numbers in the tightening procedure, as described in Fig. 2. In this experiment we used the singletons, pairs and squares regions that correspond to the grid shape graphs. We used $\phi_i(r_i) = H(b_r^*)/|r|$, where $b_r^*(x_r)$ were the optimal beliefs in our previous experiments. The results in Fig. 4 show that the state-of-the-art off-the-shelf solver, the CPLEX, is good for small scale problems but significantly worse when applied to large scale graphical models. We note that our implementation is in Matlab, which has a significant overhead when applied to small scale problems.

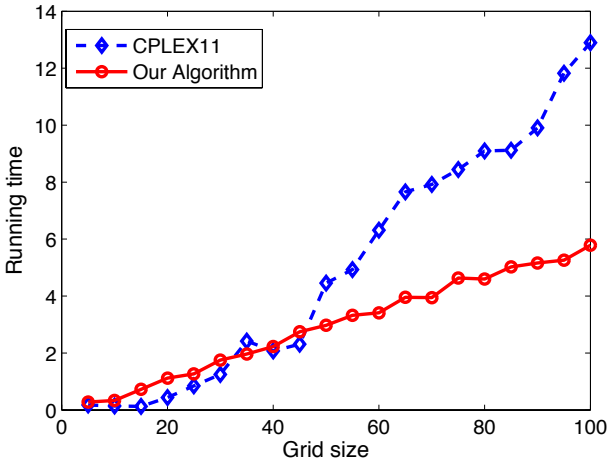


Figure 4: *Comparing our dual decomposition algorithm for recovering the direction of descent over fractional covering numbers in Section 4 with the CPLEX algorithm. Comparison is for singletons, pairs and squares regions that correspond to* 5×5*,...,* 100×100 *grid shape graphs.*

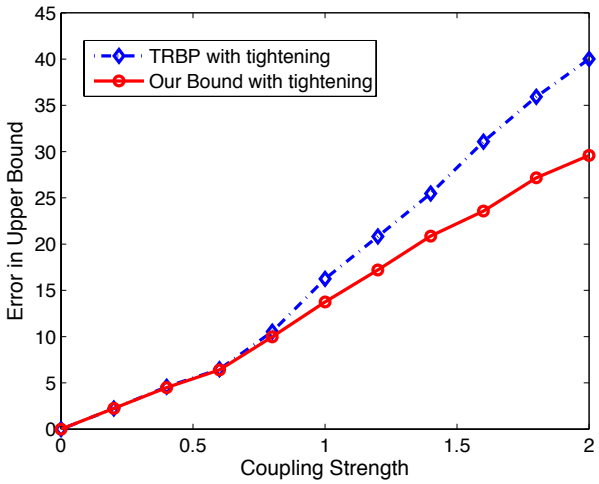


Figure 5: *Comparing tightening tree-reweighted bounds with fractional covering bounds over grid shaped graphs. Our approach can use covering numbers of high-order regions therefore achieves better bounds.*

Tightening the fractional covering upper bounds can be applied to general region graphs. However, we cannot compare our approach on high-order region graph, as all previous approaches are efficient only when the regions consist of at most pairs of indexes. Therefore we compared our approach to tightening tree reweighed upper bounds [32] implemented by [14]. In our experiments we used the grid shaped spin glass model with local field parameters $\theta_i \in [-0.05, 0.05]$ and mixed coupling potentials $\theta_{i,j} \in [-c, c]$ that ranges over $c = 0, ..., 2$. Our tightening algorithm used singletons, pairs and squares regions. The results appear in Fig. 5 showing that going to high-order regions graphs provides tighter upper bounds.

6 Related Work

In this work we investigate upper bounds on the partition function over regions graphs. For this purpose we use the known fractional covering upper bounds for the entropy function [6, 23]. We applied these bound to the partition function through conjugate duality.

Tightening upper bounds for the partition function results in two programs: A concave program for computing the upper bounds and a convex program which tightens these bounds. The field of convex (or concave) optimization is large and contains many different solvers. However, most of these solvers cannot be applied efficiently to our problems, since they ignore their structures [3, 1]. Our work differs from these works in an important respect, as we exploit the structure of the problem, decomposing the constraints through the dual program and then performing efficient closed-form steps in every iteration. This block coordinate steps in the dual enable us to efficiently deal with large-scale problems.

Upper bounds on the partition function were extensively studied in the last decade. [32] presents upper bounds for graphical models that are based on spanning trees, as well as a method to tighten these bounds using conditional gradient descent. The conditional gradient is recovered by looking for a maximal spanning tree in the graphical model. The upper bound for the graphical model is computed by a message-passing algorithm called sum-TRBP, a method that extends to region graphs [35]. This work differs from ours in important respects: First, the spanning tree method best fits graph and encounters computational difficulties when dealing with hypergraphs, or equivalently region graphs. Working with region graphs, their conditional gradient method looks for a spanning hypertree, which is a NP-hard problem [17]. This problem was already pointed out in [32] and motivates this work. Thus in our work we suggest to use the known fractional covering upper bounds over regions graphs [6, 23], and we compute the conditional gradient through dual decomposition and the entropy barrier method. Second, the sum-TRBP and its high-order extension compute a bound that has covering numbers with mixed signs. Thus the sum-TRBP is not guaranteed to converge. In contrast our work considers bounds that have only positive covering numbers, thus our bound computation is guaranteed to converge to the optimum. We

note that there are other algorithms that fix the convergence of sum-TRBP, but these algorithms cannot be applied to high-order regions graphs [9, 11].

Convexity provides a well established framework to extend the spanning trees upper bounds to other combinatorial objects. [8] describe upper bounds based on planar decomposition. Since planarity is a property of pairwise regions this work does not extend to general region graphs. In contrast, our work presents efficient bounds and tightening for high-order region graphs. [7] provide a family of upper bounds that are based on conditional entropy decompositions. These upper bounds can be applied to high-order region graphs. However, this work differs from ours in important respects: First, to compute the bound they directly solved the primal program. Since the primal program consists of conditional entropies, which are not strictly concave, they used a conditional gradient solver which is suboptimal when addressing large scale region graphs. A subsequent work presented the corresponding message-passing solver, but it was restricted to pairwise regions [9]. In contrast, our work introduces the fractional covering entropy bounds which are strictly concave, thus providing an efficient dual decomposition solver through message-passing over the region graph. Second, their tightening approach considers all conditional entropies sequences, thus they are restricted to a small number of sequences. In contrast, our work provides an efficient dual decomposition algorithm to tighten the bound over all fractional covering numbers. [21] introduce a new approach for conditional entropy decomposition, which allows to use more sequences through a mini-bucket elimination method. However, this approach suffers from similar drawbacks as [7] when applied to region graphs.

In Fig. 1 we present a message-passing algorithm to compute upper bounds on the partition function in high-order graphical models. This algorithm can be applied to general programs, consisting of sums of linear terms and entropy terms. In the last decade many different message-passing algorithms were devised for similar programs. Assuming all regions intersect on at most one variable, our algorithm has two types of messages, $\lambda_{i\to\alpha}(x_i)$ and $\mu_{\alpha\to i}(x_i)$, and it reduces to the norm-product belief propagation, which includes as special cases both sum-product and max-product, tree-reweighted belief propagation, NMPLP and the asynchronous splitting algorithm for different values of c_i, c_α [26, 32, 33, 10, 28]. However, the main purpose of our algorithm is high-order region graphs. In this perspective, when the region graph is a bipartite graph that contains outer-inner regions, our algorithm reduces the parent-to-child generalized belief propagation algorithm [36, 16, 13], and its convex forms [12, 24]. These works differ from ours in an important respect, as they consider a bipartite region graph which is computationally demanding since it uses the maximal regions in every iterations. In contrast, our algorithm considers a general region graph, thus enables to pass messages between intermediate size regions to gain computational efficiency.

7 Conclusions and Discussion

In our work we describe methods to tighten upper bounds on the partition function over general region graphs. These upper bounds use the fractional covering bounds on the entropy function. We introduced two dual decomposition algorithms, for computing the upper bound and to minimize the upper bound. Both algorithms use the entropy barrier function to obtain a closed-form block dual steps that optimize their respective dual programs.

To compute the upper bound we solve a concave program, consisting of linear terms and entropy terms. Our solver passes messages along the edges of the region graph that describe these terms. The computational complexity of this solver depends on the structure of the region graph. It turns out that there are many different region graphs, such as the inner-outer graph or the Hesse diagram that describe the same program [12, 36]. Although some works reasoned about the optimal graph, this problem is largely open [25].

Interestingly, the message-passing algorithm for computing the upper bound sends partial log-partition functions $\mu_{p\to c}(x_c)$ weighted by the covering number c_r. In particular, when $c_r = 1$ it sends a sum-product based message and when $c_r = 0$ is sends a max-product based messages. The form of these messages is determined by the covering number, and their primal interpretation relates to the weight c_r of the corresponding entropy function. For example, whenever no entropy terms are used, i.e., we use the algorithm for solving linear program relaxations, we only use max-product based operations and our algorithm contains the max-product, max-TRBP, NMPLP and convex max-product as special cases. In this perspective we could have used this algorithm to tighten linear program relaxations, but this problem was solved by [29]. Surprisingly, using fractional covering upper bounds while some of the covering numbers equal zero, we get a mixture of sum-product and max-product rules. This is a result of having sum of entropy and non-entropy terms in the primal. This goes against the common practice that mixing max-product and sum-product rules relates to the marginal-MAP solution [18, 22, 15]. The relations and differences between these two problems are subject to further research.

References

[1] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

[2] D.P. Bertsekas. Nonlinear programming. 1999.

[3] S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] T.M. Cover, J.A. Thomas, J. Wiley, et al. *Elements of information theory*, volume 6. Wiley Online Library, 1991.

[5] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1627–1645, 2009.

[6] E. Friedgut. Hypergraphs, entropy, and inequalities. *The American Mathematical Monthly*, 111(9):749–760, 2004.

[7] A. Globerson and T. Jaakkola. Approximate inference using conditional entropy decompositions. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

[8] A. Globerson and T. Jaakkola. Approximate inference using planar graph decomposition. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 473. The MIT Press, 2007.

[9] A. Globerson and T. Jaakkola. Convergent propagation algorithms via oriented trees. In *UAI*, 2007.

[10] A. Globerson and T. S. Jaakkola. Fixing max-product: convergent message passing algorithms for MAP relaxations. In *NIPS*, 2007.

[11] T. Hazan and A. Shashua. Convergent message-passing algorithms for inference over general graphs with convex free energies. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, Helsinki, Finland, July 2008.

[12] T. Heskes. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Artificial Intelligence Research*, 26(1):153–190, 2006.

[13] T. Heskes and O. Zoeter. Generalized belief propagation for approximate inference in hybrid bayesian networks. In *Artificial Intelligence and Statistics*. Citeseer, 2003.

[14] A. Jaimovich, O. Meshi, I. McGraw, and G. Elidan. Fastinf: An efficient approximate inference library. *The Journal of Machine Learning Research*, 11:1733–1736, 2010.

[15] J. Jiang, P. Rai, and H. Daumé III. Message-passing for approximate map inference with latent variables. NIPS, 2011.

[16] H.J. Kappen and W. Wiegerinck. Novel iteration schemes for the cluster variation method. In *Advances in Neural Information Processing Systems 14*, 2001.

[17] D. Karger and N. Srebro. Learning markov networks: Maximum bounded tree-width graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 392–401. Society for Industrial and Applied Mathematics, 2001.

[18] D. Koller and N. Friedman. *Probabilistic graphical models*. MIT press, 2009.

[19] T. Koo, A.M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298. Association for Computational Linguistics, 2010.

[20] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference of Machine Learning*, pages 282–289, 2001.

[21] Q. Liu and A. Ihler. Bounding the partition function using hölders inequality. 2011.

[22] Qiang Liu and Alexander Ihler. Variational algorithms for marginal map. In *Uncertainty in Artificial Intelligence (UAI)*. Barcelona, Spain, 2011.

[23] M. Madiman and P. Tetali. Information inequalities for joint distributions, with interpretations and applications. *Information Theory, IEEE Transactions on*, 56(6):2699–2713, 2010.

[24] T. Meltzer, A. Globerson, and Y. Weiss. Convergent message passing algorithms-a unifying view. In *UAI*, 2009.

[25] P. Pakzad and V. Anantharam. Estimation and marginalization using the kikuchi approximation methods. *Neural Computation*, 17(8):1836–1873, 2005.

[26] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[27] M.J.D. Powell. On search directions for minimization algorithms. *Mathematical Programming*, 4(1):193–201, 1973.

[28] N. Ruozzi and S. Tatikonda. Convergent and correct message passing schemes for optimization problems over graphical models. *Arxiv preprint arXiv:1002.3239*, 2010.

[29] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Conf. Uncertainty in Artificial Intelligence (UAI)*. Citeseer, 2008.

[30] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1068–1080, 2007.

[31] L.G. Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

[32] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *Trans. on Information Theory*, 51(7):2313–2335, 2005.

[33] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *Trans. on Information Theory*, 51(11):3697–3717, 2005.

[34] M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[35] W. Wiegerinck. Approximations with reweighted generalized belief propagation. 2005.

[36] JS Yedidia, WT Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005.

Inferring Strategies from Limited Reconnaissance in Real-time Strategy Games

Jesse Hostetler and **Ethan Dereszynski** and **Tom Dietterich** and **Alan Fern**
{hostetje, dereszet, tgd, afern}@eecs.orst.edu
Department of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331

Abstract

In typical real-time strategy (RTS) games, enemy units are visible only when they are within sight range of a friendly unit. Knowledge of an opponent's disposition is limited to what can be observed through scouting. Information is costly, since units dedicated to scouting are unavailable for other purposes, and the enemy will resist scouting attempts. It is important to infer as much as possible about the opponent's current and future strategy from the available observations. We present a dynamic Bayes net model of strategies in the RTS game *Starcraft* that combines a generative model of how strategies relate to observable quantities with a principled framework for incorporating evidence gained via scouting. We demonstrate the model's ability to infer unobserved aspects of the game from realistic observations.

1 INTRODUCTION

Real-time strategy (RTS) games, which are video game simulations of both the economic and military aspects of warfare, present a rich variety of reasoning challenges. Players must manage resources, plan at multiple time scales, coordinate heterogeneous armies composed of as many as 100 individual units, and respond to the actions of adversaries, all with incomplete information about the game state and under real-time constraints. We are particularly interested in the problem of decision-making with incomplete information. Specifically, how can an agent predict the intentions of its opponent from a limited number of observations?

This work addresses the problem of *unit count inference* during the early stages of the game. In the opening minutes, players make choices about what kinds of units to build and what technologies to pursue. These choices dictate the overall "feel" of the game: aggressive versus defensive, economy versus technology, etc. The players' plans are reflected in their opening *build orders*. Because resources are limited in the early game, players must commit to a particular path through the technology tree, culminating in a certain composition of the initial army. The primary concern early in the game is to choose an opening strategy that will gain an advantage against the opponent's opening.

This task is complicated by the limited information available to the players. Enemy units are visible only if a friendly unit is nearby. A player's knowledge of her opponent's strategy is thus limited to what has been observed through reconnaissance. It is impractical to observe all important actions of the enemy, because attempting to do so would divert many units from other tasks, and the enemy will attempt to destroy the scouts before they learn anything useful. Instead, players must gain whatever information they can with reasonable effort, and then evaluate their limited knowledge to infer the likely disposition of the enemy. An effective model of opening strategy must be able to make useful inferences from realistic evidence.

We present a dynamic Bayesian network model of opening strategy and apply it to the RTS game *Starcraft*. We combine a latent variable model of the true game state with an observation model that describes how the latent state generates the observed state. The true state, which we call the *battle space*, can be described by the true counts of each type of unit controlled by the enemy. The evolving counts of the various unit types are represented as Markov processes that are conditionally independent given the history of a hidden *strategy* process. The strategy determines which units are produced in a given time step, and the produced units accumulate to form the true counts. The observation model then determines the likelihood of observations given the true state and a measure of observation "effort." Although the hidden state space is potentially very large due to the number and car-

dinality of the "count" variables, the structure of the model admits an efficient particle filtering algorithm for inference, taking advantage of the conditional independence of the unit counts.

1.1 REAL-TIME STRATEGY GAMES

We now briefly describe the mechanics of a typical RTS game. RTS games are simulations of total warfare. Players claim *bases*, which contain *resources*, by building *city centers*. They then use *workers* to harvest the resources, which they spend to construct *buildings* and *mobile units*. We will use the generic term *units* to refer both to buildings and to mobile units. *Production buildings* produce mobile units, including both workers and military units. In contrast, constructing *tech buildings* enables access to more powerful fighting units, and to further tech buildings. The game ends when one player controls no buildings, though players will typically resign a lost position much earlier.

Most RTS games contain a mechanism called the "fog of war". A metaphor for the limited intelligence available to military commanders, the fog of war makes enemy units invisible unless they are within visual range of a friendly unit. Players who lack accurate information about their opponents' activities are at risk of being surprised by attacks for which they are not prepared. Skilled players thus make scouting a high priority, and they are adept at inferring their opponents' intentions on the basis of that scouting. It is this inference process that we seek to model.

1.2 RELATED WORK

A complete description of a player's opening would specify how many units of each type existed at any moment. To our knowledge, no one has yet attempted to create a model of openings at this level of granularity. Existing work simplifies the task by adopting less-detailed descriptions of opening strategies.

A common simplification is to assume a small number of opening categories, such as "rush attack" or "strong economy." Weber and Mateas (2009) created a set of labels for *Starcraft* openings based on player-community vocabulary, and tried a variety of supervised classifiers for predicting the labels. Schadd et al. (2007) designed a two-level hierarchy of labels for the free RTS game *Spring*.

An alternative to categorical strategies is to model the dynamics of the game state itself. One can then represent an opening strategy as a path through the state space. Ponsen et al. (2007) used a finite state machine (FSM) model of building construction in *Wargus*. States represent which buildings have been built, and transitions are triggered by the construction of novel building types. Hsieh and Sun (2008) took a similar approach in *Starcraft* but used a stochastic FSM, with transition probabilities learned from gameplay data. Dereszynski et al. (2011) used a hidden Markov model (HMM). Transitions between hidden states occur at fixed time intervals, and the observations are Bernoulli random variables specifying the probability of building units of each type in the current state.

Except for Schadd et al. (2007), the work discussed so far assumes that the agent has access to information, such as the times at which particular units were constructed, that would be difficult to observe during actual gameplay. Another exception is Synnaeve and Bessière (2011), who used a directed graphical model to describe a joint distribution over strategy categories, game states, and observations in *Starcraft*. Their game states are Boolean vectors indicating whether each type of unit has been built, and game states are conditionally independent given strategy categories. Observations are incorporated by assigning 0 posterior probability to states that are inconsistent with observations.

The idea of an observation model (also known as a detection model) has been applied in many settings. Observation models are needed when we do not observe the true state of a system, but rather the results of some process parameterized by the latent true state. For example, in ecological modeling, we are often interested in the true number of individuals of a certain species present in the environment, but have access only to counts provided by field observers, who may not detect the species even if it is present (MacKenzie et al., 2002). The observation model describes how the hidden state determines the observed state, possibly as a function of covariates such as observer effort. One can then find the MAP estimate of the hidden state, accounting for the evidence. Our model is quite similar in spirit to the "occupancy-detection" models discussed by Hutchinson et al. (2011).

1.3 OUR CONTRIBUTIONS

This work improves upon existing models of RTS game opening strategy in several ways. First, we model the game state at the level of individual unit counts over time, enabling inference at the level of detail necessary for making gameplay decisions. Second, because individual unit counts are directly observable during gameplay, our model does not require evidence that is impossible to acquire in order to make inferences. Third, because our observation model includes predictors of scouting success, we can extract maximum value from limited observations, particularly from *failure* to observe a unit despite significant effort.

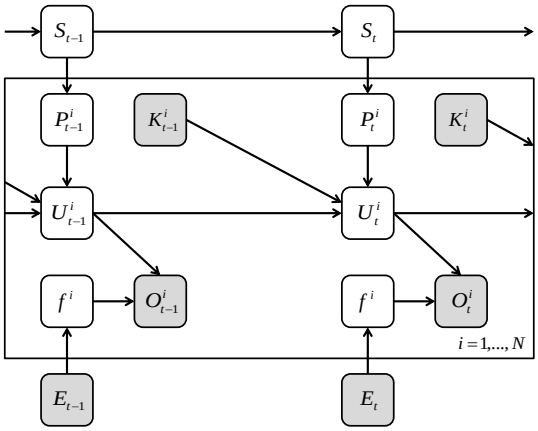


Figure 1: Two-slice representation of the reconnaissance model. We use plate notation to show that separate P, K, U, f, and O variables exist for each unit type i (but note that the variables are not exchangeable). There is a direct link from U^i_{t-1} to U^i_t for each unit type. Shaded rectangles denote variables that we can directly observe.

2 THE MODEL

The overall model (Figure 1) is a discrete-time dynamic Bayes net (Dean and Kanazawa, 1988), combining two distinct components, the *state model* and the *observation model*, which we describe separately.

2.1 STATE MODEL

The state model describes the process that generates the true state of the battle space. Each game corresponds to a path through a discrete state space beginning at a designated start state. Let S_t denote the state at time t. Time is discretized into 30-second intervals, and the player makes a state transition once every 30 seconds. We will say "epoch k" or "time $t = k$" to denote the time step begun by the kth transition. When the player visits state s, he or she decides how many units of each type to produce. Let P^i_t be the number of units of type i produced at time t. We model the decision to produce k units of type i as occurring in two steps. First, a decision is made about whether to produce any units of type i at all. Then, if the first decision is to produce one or more units, a decision is made about how many units to produce. This is modeled by the zero-inflated Poisson distribution

$$P(P^i_t = k | S_t = s) = \begin{cases} 1 - \nu^i_s & k = 0 \\ \nu^i_s \cdot Pois(k-1; \lambda^i_s) & k > 0 \end{cases}$$

Here, ν^i_s is the probability of producing one or more units of type i in state s, and hence, $1 - \nu^i_s$ is the probability of producing zero units. The number of units produced (beyond the first) is then determined by the Poisson rate parameter λ^i_s.

We model production as a two-step decision to make it easier to capture the decision not to produce any units of a particular type. This is important for modeling the production of *tech buildings*. In most cases, only one tech building of a given type is needed—it just serves as a prerequisite for producing higher-tech units. Decisions to produce tech buildings reflect strategic choices, so it is important to capture these 0/1 decisions. The alternative of just using a Poisson distribution does not handle this well, because there is no way to control the probability of producing 0 units without also changing the shape of the distribution. Similarly, modeling the unit production as Bernoulli random variables is not appropriate, because there do exist cases where two such buildings will be produced.

Let $\mathbf{P}_t = (P^i_t)_{i=1}^N$ denote the vector of unit production for all N unit types at time t. In our model, the P^i_t variables are independent conditioned on the current state S_t. Of course, there are important constraints among production rates of multiple unit types, since producing units of one type consumes resources that cannot be spent to produce other types. In principle, these constraints can be incorporated by using a sufficiently large number of states. However, that may conflict with the goal of capturing the player's overall strategy via the sequence of state transitions. Thus, the number of states must be selected appropriately to balance these objectives.

When scouting, we cannot directly observe the production of units; rather, a scout only observes (a subset of) the units that exist at a given time. To connect the individual unit production decisions to the scouting observations, we need to model the total number of units U^i_t of type i that exist at time t. This is equal to the number of units produced by time t minus the number that have died. Each U^i_t takes values in $\{0, 1, \ldots, U_{\max}\}$, where $U_{\max}$ is the maximum number of units that can exist. Because the production variable P^i_t has infinite support, the assignment $U^i_t = U_{\max}$ represents the situation where there are $U_{\max}$ *or more* units of type i.

Unit deaths can occur in two ways. Most deaths involve our units killing their units, and these deaths, $\mathbf{K}_t = (K^i_t)_{i=1}^N$, are observed. Each unit of type i also has a small probability ℓ^i of suffering an unobserved loss in each time step. For example, a building that was scouted while under construction may subsequently be canceled. Starting from initial unit counts c^i, the total count of unit type i at time t is defined recursively as

$$\begin{aligned} U^i_0 &= c^i \\ U^i_t &\sim Binomial(U^i_{t-1} - K^i_{t-1}, 1 - \ell^i) + P^i_t \end{aligned}$$

When calculating U^i_t in the case where $U^i_{t-1} = U_{\max}$,

we do not know the exact value represented by U^i_{t-1}, so we assume that it is *exactly* $U_{\max}$. Very rarely, this will lead us to underestimate U^i_t, but we set $U_{\max}$ to be large enough to avoid this problem in most cases.

Let $\mathbf{U}_t = (U^i_t)_{i=1}^N$ be the vector collecting the numbers of units of each type that exist at time t. We refer to $\mathbf{U}_t$ as the "battle space" at time t.

2.2 OBSERVATION MODEL

The *observation model* specifies the likelihood of an observation given a particular assignment to $\mathbf{U}_t$. We define an *observation* as the number of units of each type that we saw during a time interval, $\mathbf{O}_t = (O^i_t)_{i=1}^N$. We assume that we can distinguish between individual enemy units within an epoch, but not between epochs. That is, if we make two observations of a unit of a certain type during the same epoch, we know whether we have seen the same unit twice, or two different units. We can therefore model these observations as sampling without replacement from $\mathbf{U}_t$ within an epoch. For observations in different epochs, we assume that we cannot tell whether a unit observed in one epoch is the same as a unit observed in a previous epoch.

If we observe an enemy unit, we know that the unit exists. If, on the other hand, we do not observe a unit, there are two possible explanations. Either the unit does not exist, or it does exist but we did not look hard enough for it. Hence our observation model needs to incorporate some measure of scouting "effort". If we put little effort into scouting, failure to observe a unit does not tell us much about whether it exists. If we scout extensively and still do not see the unit, it probably does not exist.

We can exploit domain knowledge to come up with a measure of effort. In *Starcraft*, players construct most of their buildings in either their *main base* or their *natural expansion*. The main base is the area of the map where the player's starting units appear at the beginning of the game, while the natural expansion is the location where it is most "natural" to construct a second base. Because buildings must be defended, it is tactically advantageous to keep them close together. In the early game, the main base and the natural expansion are thus the most important areas to scout, since that is where the buildings will be located. A natural measure of scouting effort, then, is the proportion of these two areas that have been seen. We denote this proportion for slice t as E_t.

We must now decide how our scouting effort influences the number of units we observe. Our initial approach was to treat each observable unit as an independent Bernoulli trial, with probability of success E_t. An observation would then be a vector of Binomial random variables, $O^i_t \sim Binomial(U^i_t, E_t)$.

The Binomial model assumes that the locations of units are distributed uniformly and independently in space. However, this assumption is wrong. Units tend to cluster together. For example, the primary task of "worker" units is to gather resources, which they do by traveling back and forth between the city center and the resource. Thus, almost all worker units will be found in the area between the city center and the resources. If we see one worker, it is probably because we have seen part of this area, and we would expect to have seen most of the other workers, too. There is thus more variance in the observations than the Binomial model would predict; the data are *overdispersed* with respect to the Binomial distribution.

We account for overdispersion by placing a Beta prior on the success probability parameter of the Binomial, forming a Beta-Binomial model (Haseman and Kupper, 1979):

$$\begin{aligned} O^i_t &\sim BetaBinomial(U^i_t, \mu^i_t, \rho^i) \\ &= \binom{U^i_t}{O^i_t} \frac{B(O^i_t + \alpha^i_t, U^i_t - O^i_t + \beta^i_t)}{B(\alpha^i_t, \beta^i_t)} \end{aligned}$$

where $B(x, y)$ is the beta function, and

$$\alpha^i_t = \mu^i_t \frac{1-\rho^i}{\rho^i}, \quad \beta^i_t = (1-\mu^i_t)\frac{1-\rho^i}{\rho^i}.$$

We adopt the (μ, ρ) parameterization of the Beta distribution, where $\mu \in [0, 1]$ is the mean of the Beta and $\rho \in [0, 1]$ is the dispersion parameter, which can be thought of as the correlation between individual successes. The Beta distribution is a conjugate prior of the Binomial. When $\rho \to 0$, the Beta-Binomial approaches the Binomial, while as ρ increases, the density spreads out and eventually becomes bimodal. The bi-modality captures the "clumpiness" of the units: depending on whether the part of the area of interest we saw contains the units, the success probability is either high or low, but probably not in the middle.

For each unit type i, we learn a mapping $f^i(E_t)$ from the observation effort to the mean and dispersion of a Beta-Binomial distribution:

$$\begin{aligned} logit(\hat{\mu}^i_t) &= a^i_0 + a^i_1 E_t \\ logit(\hat{\rho}^i) &= b^i. \end{aligned}$$

This is then plugged into the Beta-Binomial to compute the likelihood of observing O^i_t given that U^i_t units exist. We learn a different mapping for each unit type, to allow for differences in dispersion and ease of observability between unit types. The regression coefficients are assumed constant in time, but $\hat{\mu}^i_t$ varies in time due to its dependence on E_t.

2.3 TRAINING

All model variables except $\mathbf{S}$ are observed during training. Because all other variables are conditionally independent of $\mathbf{S}$ given $\mathbf{P}$ and $\mathbf{P}$ is observed, we can factor the full model into a latent variable model composed of $\mathbf{S}$ and $\mathbf{P}$, and a fully observed model containing the rest of the parameters. We can learn the parameters of $\mathbf{S}$ and $\mathbf{P}$ separately from the rest, simplifying training.

The production process is a hidden Markov model (HMM) (Rabiner, 1990), where S_t is the latent state and $P_t^1, ..., P_t^N$ are the emissions. The parameters of the HMM are the initial state probabilities $P(S_0) = Multinomial(\eta_1, ..., \eta_M)$, the state transition probabilities $P(S_t|S_{t-1} = s) = Multinomial(\pi_1^s, ..., \pi_M^s)$, and the inflated Poisson parameters $P(P_t^i = k > 0|S_t = s) = \nu_s^i \cdot Pois(k - 1; \lambda_s^i)$. We denote the parameter set of the HMM $\Phi = (\eta_1, ..., \eta_M, \pi_1^1, ..., \pi_M^M, \lambda_1^1, ..., \lambda_M^N, \nu_1^1, ..., \nu_M^N)$.

At training time, we observe $\mathbf{P}_t$. We can then estimate Φ in the usual way using the Expectation Maximization (EM) algorithm. We initialized the EM algorithm as follows: the η and π parameters are set to $1/M$, values of ν are drawn from a Uniform$(0, 1)$, and λ parameters are drawn from a Uniform$(0, 10)$.

The "unobserved loss" probabilities ℓ^i are estimated as the number of unobserved losses of units of type i divided by the number of unit-epochs (analogous to human-years) during which a unit of that type existed. For unit types that were present in at least 100 unit-epochs, ℓ^i was estimated using additive smoothing as $\hat{\ell}^i = \frac{d^i+1}{D^i+2}$ where d^i is the number of unobserved losses of unit type i and D^i is the number of unit-epochs for type i. The smoothing ensures that all unit types have non-zero ℓ^i even if there were no unobserved losses in the training data. For types that were not present in at least 100 unit-epochs, the median estimate was used.

The functions f^i giving the parameters of the distributions of O_t^i are learned via logistic regression with a maximum likelihood objective using the R package `aod` (Lesnoff et al., 2010). We fit a $\hat{\mu}^i$ parameter for unit type i only if a unit of that type was observed on at least 100 occasions in the dataset. We fit a $\hat{\rho}^i$ parameter only if the unit type met the condition for $\hat{\mu}^i$ and there were at least two of the unit type present (but not necessarily observed) on at least 100 occasions. The reason for the condition on $\hat{\rho}$ is that if it is rare for more than one instance of the unit to exist, then there is little dispersion in the data, and the estimate of $\hat{\rho}^i$ will be near 0. In this case, $\hat{\rho}^i$ would merely be modeling the tendency not to build more than one unit, which is properly the job of $\mathbf{P}_t$. For types that did not have enough data for $\hat{\mu}$ or $\hat{\rho}$, the median of the estimated regression coefficients are used.

2.4 INFERENCE

We denote the subset of latent variables for a slice t as $X_t = \{S_t, P_t^1, \ldots, P_t^N, U_t^1, \ldots, U_t^N\}$, and the observed variables as $Y_t = \{E_t, K_t^1, \ldots, K_t^N, O_t^1, \ldots, O_t^N\}$. We use the lowercase y_t and x_t to denote instantiations of these variables (i.e, y_t refers to the evidence at time t). Because each U_t^i is conditioned on S_t, and S_t and U_t^i are Markovian, an exact filtering pass would require representing the forward message $\alpha_t = P(S_t, U_t^1, ..., U_t^N)$. This is intractable for even a modest number of types, since the size of the joint distribution is $MU_{\max}^N$. However, a key observation is that, given the history of the strategy state, $S_{0:t} = (S_0, \ldots, S_t)$, the model up to time t decomposes into N independent HMMs, each tracking the count of a single type. We leverage this structure by employing a Rao-Blackwellized particle filter (RBPF) for approximate inference (Doucet et al., 2000; Murphy, 2000).

In our application of RBPF, we draw particles of $S_{0:t}$, and compute $P(U_t^i|s_{0:t})$ analytically via standard HMM filtering. Following an importance sampling framework, particles are generated at each time step from a proposal distribution $Q(S_t)$. We use the state transition model for our proposal, $Q(S_t) = P(S_t|s_{t-1})$. While this choice ignores recent evidence at time t, it is computationally efficient to sample, and there are often periods of no evidence, anyway.

At $t = 0$ we draw R particles from the initial state prior $s_0^1, \ldots, s_0^R \sim P(S_0)$. Each particle has an importance weight $w_t^r = \phi(s_t^r)/Q(s_t^r)$, where $\phi(s_t^r)$ is the probability of the particle's value s_t^r given by the full model (up to normalization). At $t = 0$, $w_0^r = (P(y_0|s_r)P(s_r))/Q(s_r)$. For $t > 0$, each particle generates its next value of the state $s_t^i \sim Q(S_t) = P(S_t|s_{t-1}^i)$. We then update its weight using the ratio:

$$w_t^{r'} = \frac{P(Y_t = y_t|y_{0:t-1}, s_{0:t}^r)P(S_t = s_t^r|S_{t-1} = s_{t-1}^r)}{P(S_t = s_t^r|S_{t-1} = s_{t-1}^r)}.$$

The new weight for particle r is then $w_t^r = w_{t-1}^r w_t^{r'}$.

Because our proposal distribution is identical to $P(S_t|S_{t-1})$ (canceling out the denominator), we are only interested in the likelihood term of the numerator, $P(Y_t = y_t|y_{0:t-1}, s_{0:t}^r)$, which factors as

$$\prod_{i=1}^{N} \Big[P(U_t^i|U_{t-1}^i, P_t^i, K_{t-1}^i)P(U_{t-1}^{i,r}) \cdot P(P_t^i|S_t = s_t^r)P(O_t^i|U_t^i, E_t^i) \Big].$$

$P(U_{t-1}^{i,r})$ is a forward-pass message that captures the posterior marginal distribution over the counts of unit type i at time $t - 1$. After we weight a sample, we

compute $P(U_t^{i,r}|s_{0:t}^r, y_t)$ to update its belief about the unit count and to pass to slice $t+1$. Each particle computes N such messages, one for each unit type.

Particle filters often include a resampling step in which a new set of particles is sampled in proportion to the weights of the current set and given uniform weights. We do not include a resampling step because we expect that there will often be periods of no observations, particularly in the opening several epochs. Resampling would reduce the diversity of the particle population, with the result that if observations later come in that suggest *a priori* unlikely strategies, there may be no particles left that can represent them well.

3 EXPERIMENTS

We evaluated our model on its ability to infer the correct hidden unit counts, given the observations available to players during real games. We used two different metrics. For unit types that are usually present in numbers greater than 1, such as army units, we measured the model's ability to infer the correct *number* of units. For unit types that are either not built or built once, such as tech buildings, we measured accuracy in determining whether or not the unit is present. We also specifically tested the model's ability to infer the *absence* of units, a task that requires the observation model and the state model to work together.

Our experiments were conducted using gameplay data[1] from the RTS game *Starcraft*. We collected replays of 509 Protoss versus Terran games from the archives of the "Gosu Gamers" website.[2] *Starcraft* features three playable "races," Protoss, Terran, and Zerg, each with different units and abilities. We focused on a single match-up in this work. In our experiments, we take the perspective of the Terran player, and try to predict what the Protoss player is doing.

Data about unit counts and observations was extracted from the replays using the BWAPI library (BWAPI, 2012). We used the BWTA terrain analysis library (Perkins, 2010) to divide the maps into regions, and manually identified the regions corresponding to the Protoss player's main base and natural expansion.

The dataset was divided into 5 folds for cross-validation. To select M (number of strategy states) in the production model, we compared the average likelihood $P(\mathbf{P}|\Phi)$ on the held-out data for $M = 20, 25, 30, 35$, and 40. The likelihoods for $M = 25, 30$, and 35 were statistically identical, and we selected $M = 30$ based on examination of the learned parameters and our knowledge of *Starcraft*. After selecting M, we estimated the observation model parameters for each of the 5 folds using their respective training sets.

[1]Our dataset is available at `http://web.engr.oregonstate.edu/~tgd/rts/scouting/`

[2]`http://www.gosugamers.net/starcraft/replays/`

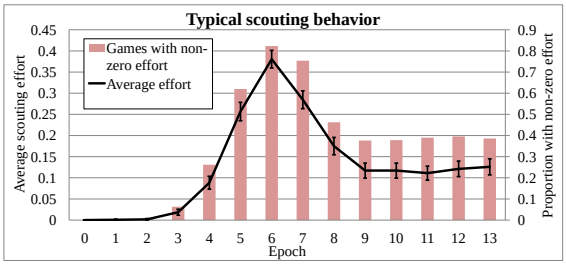


Figure 2: Average scouting effort, and proportion of games with non-zero effort, by epoch.

3.1 BASELINE METHOD

We implemented a simple baseline consisting of two sets of averages: the average number of units of each type at the end of each epoch for 1) all games, and 2) only games in which the unit was present in at least one epoch. At time t, if no units of type i have been scouted so far, the baseline predicts the average for type i at time t across all games. If at least one unit of type i has been scouted, the baseline predicts the average across games in which that unit type was present. The baseline should perform fairly well in the opening, since many openings are similar and thus unit counts often have low variance.

3.2 TYPICAL SCOUTING BEHAVIOR

Since our model uses observations obtained through scouting as its evidence, it is important to know what typical scouting behavior looks like. There is a pronounced peak in scouting effort from $t = 5$ to $t = 8$, after which effort falls back to a steady "background" level (Figure 2). The first significant scouting appears to begin at $t = 4$. Based on this pattern of scouting, we should expect our model to perform best during the peak scouting period, $t = 5, 6, 7, 8$.

3.3 MODEL ANALYSIS

The (logistic-transformed) regression coefficients of the observation model reveal varying levels of dispersion for different types of units. Typical values of $\hat{\rho}^i$ for mobile units were around 0.3, suggesting that mobile units often travel in groups. Values for buildings varied more widely. For Gateways (which produce military units), $\hat{\rho}^i$ was equal to 0.71, indicating that Gateways are very likely to be close to one another. At the other extreme, the value of $\hat{\rho}^i$ for the Nexus (the "city center" where resources are deposited by workers) was 10^{-8}, since Nexi are never built near one another. The regression coefficients of the scouting effort were al-

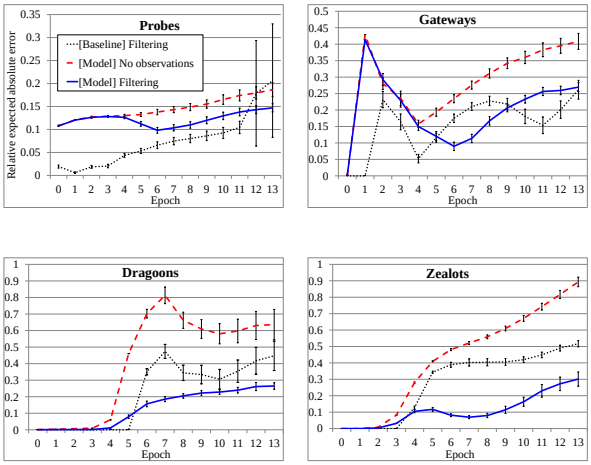


Figure 3: Relative expected absolute error for filtering predictions of common unit types, versus baseline. The error bars are 95% confidence intervals.

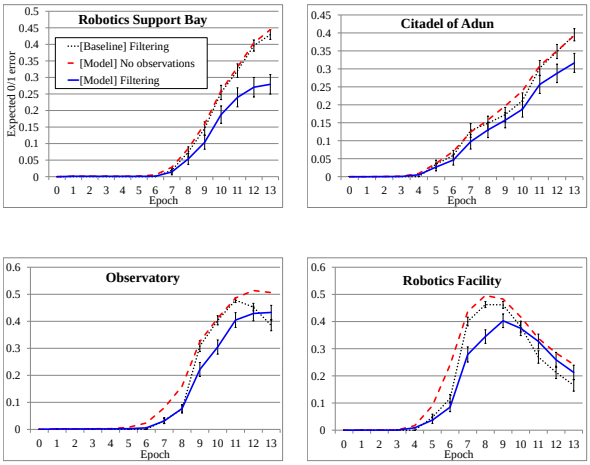


Figure 4: Expected 0/1 error for filtering predictions of tech buildings, versus baseline. The error bars are 95% confidence intervals. Error bars have been omitted for the model with no observations, for clarity.

most all near 1.0, indicating that our scouting effort measure is a good predictor of scouting success. The two effort coefficients that were substantially less than 1 corresponded to units with the *Cloak* ability, which can only be seen by particular kinds of units. For one of these units, the Observer, the model learned that greater scouting effort *decreases* the probability of detection. This seems incorrect; we believe it occurs because in addition to being hard to detect, Observers usually are not fielded until after the period of peak scouting, so effort is negatively correlated with Observer presence.

3.4 INFERRING UNIT QUANTITIES

For common units that are built in large numbers, we would like to be able to infer the true counts. In *Starcraft*, the most common units in the early game are the Probe (the Protoss worker), the Dragoon and Zealot (basic military units), and the Gateway (a building that produces Dragoons and Zealots). Figure 3 compares our model's performance to that of the baseline for the filtering task over these four unit types. The dotted lines show the baseline method, the solid lines show our model's accuracy at filtering, and the dashed lines show our model's accuracy with no observations. The error measure is relative expected absolute error. That is, for a true count u_t^i and the model's marginal distribution U_t^i, the error is given by

$$\varepsilon_t^i = E[|U_t^i - u_t^i|]/(u_t^i + 1).$$

We outperform the baseline from the onset of scouting for both Dragoons and Zealots. For Probes, we do worse for most of the opening, although both models have low error. We can attribute the good performance of the baseline to the low variance in Probe counts. Players almost always build Probes as quickly as possible in order to increase resource income, up to a "saturation" point that depends on the number of bases the player has secured. The large increase in the baseline's error at the end of the opening suggests that this is a point where some players have reached saturation while others have not. Our model does not experience a notable increase in error because it can represent multi-modal distributions. For Gateways, the baseline is notably better from $t = 10$ to $t = 12$. This is explained by the typical timing of Gateway construction. The first dip at $t = 4$ is a time when nearly everyone has built their first Gateway. Players will then build a second "batch" of Gateways, with the exact timing and number depending on their strategy. The second dip in baseline error around $t = 11$ is the end of the second period of Gateway production. Our model has trouble capturing this structure because it is stationary.

The shapes of the error curves for our model in the case of no observations are very similar to the baseline, although the baseline generally has better accuracy. Our model tracks the average-case behavior, but over-predicts production due to the Markov assumption on strategy states. For example, our model will give some probability of transitioning to a Dragoon-producing state earlier than a Dragoon could possibly have been produced.

3.5 INFERRING TECH BUILDINGS

While counts are important for units that are produced in numbers, for tech buildings we are primarily concerned with whether or not they exist. Thus, we compared our model to an appropriately modified baseline

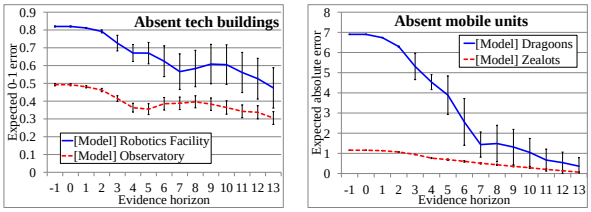


Figure 5: Error in predictions for $t = 13$ given evidence up to various horizons, on games where the target unit is *not* present. Error bars are 95% confidence intervals.

using expected 0/1 error for four tech buildings.

There are two qualitatively different shapes in the baseline error curves (Figure 4). For the Robotics Facility and the Observatory, baseline error peaked and then decreased, while for the Citadel of Adun ("Citadel") and Robotics Support Bay ("Support Bay"), it continued to increase for the entire opening. Our model with no observations tracks the baseline error closely, until the baseline error begins to drop.

We again seem to outperform the baseline from the onset of scouting, though the baseline sometimes makes a comeback at the end of the opening. As with the Gateway results in Section 3.4, the drop in error for the Robotics Facility at the end of the opening is explained by typical strategy. The great majority of openings involve building a single Robotics Facility before epoch 13. In constrast to the Gateway results, our model was able to capture this temporal structure for the Robotics Facility, perhaps because a Robotics Facility is built only once, whereas multiple Gateways are built over a period of time. On the other hand, a Support Bay will only be built in some openings, and will be skipped entirely in others. We would expect error to stabilize after the time period during which the Support Bay would be constructed passes, and we do see some evidence of this for our model with observations. It is also worth noting that the probability of scouting a Robotics Facility given that it exists is 0.37, while the probability of scouting a Support Bay is only 0.18.

3.6 INFERRING ABSENCE OF UNITS

Because we model the probability of observation success as a function of effort put into observing, we can make maximum use of *negative* observations to infer that units are *not* present. In Figure 5, we show our model's error for predictions of the *final* epoch, given evidence up to varying *horizons*, in games where the target unit was *not* present. For example, at horizon 6, the model is making predictions for $t = 13$ given evidence through $t = 6$. The true count is equal to 0, so all of the error comes from over-predicting. We see that as more negative observations arrive, the

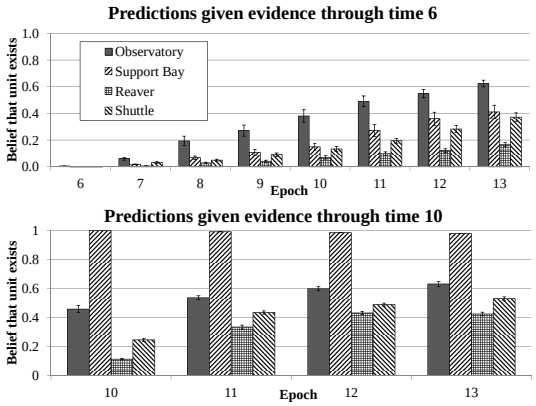


Figure 6: Belief that a unit of each of four different types will exist in future epochs in our Reaver drop case study game, given evidence up to $t = 6$ and $t = 10$. The Shuttle and Support Bay were actually built at $t = 9$, and the Reaver was built at $t = 11$. No Observatory was built. Error bars are 95% confidence intervals over 30 runs.

model revises its predictions downwards. The observation model uses the negative observations to infer that there are currently no units, and the rates of production given by the state model limit the number of units that could be built in the remaining time. The effect is strongest for units that are normally easy to scout. For example, Dragoons are present in all but 11 games, and they are easy to observe because they will be trying to attack the scout. Similarly, Robotics Facilities are both more common and more frequently scouted than Observatories. The baseline (not shown) would have a large, flat error on this task.

3.7 CASE STUDY

To demonstrate how our model would be applied in a game-playing agent, we examined a single game in detail. In this game, the Protoss player follows the *Reaver drop* strategy, which involves using a transport aircraft to carry a powerful unit called a Reaver behind enemy lines to attack the workers. This attack is potentially devastating, but easy to stop unless we are caught by surprise. We must note that one of the reasons that we chose this particular game is that the Terran player's scouting was particularly effective, giving our model a chance to make interesting inferences.

The key units in this strategy are the *Robotics Support Bay*, which is the tech building required for the Reaver; the *Shuttle*, which is the aircraft that transports the Reaver; and of course the Reaver itself. We can also contrast the Reaver drop opening to the "standard" opening, which involves building an *Observatory*.

This game features two distinct periods of scouting.

The Terran player scouts the Protoss at $t = 6$, leaves for a while, then returns to scout again at $t = 10$. Figure 6 shows, for each of the key units, the model's belief that at least one such unit will exist at each future time step, given evidence up to $t = 6$ and $t = 10$.

We first examine our model's predictions with evidence up to $t = 6$. The Terran player has just scouted the Robotics Facility, but the Support Bay and Reaver have not been built yet. Belief that an Observatory or a Support Bay exists begins to increase at $t = 8$. The Observatory is the "standard" continuation, so it receives more belief. Belief that a Shuttle exists increases in step with belief in the Support Bay because the Support Bay is strongly associated with the Reaver drop strategy, which always features a Shuttle. Belief that a Reaver exists increases more slowly than belief in the Support Bay because a Support Bay must be completed before a Reaver can be built. In the game, the Support Bay and Shuttle were actually built at $t = 9$, and the Reaver was built at $t = 11$.

We now examine how the model's predictions change when more evidence arrives. At $t = 10$, the Terran player scouts the Support Bay. This should unambiguously signal that a Reaver is going to be built, since there is no other reason to build a Support Bay. As expected, belief that a Reaver is coming increases considerably compared to the prediction with evidence through time 6. Belief that a Shuttle exists has also increased.

4 DISCUSSION

Our model generally performed well in comparison to the baseline for both count predictions and 0/1 predictions. The fact that our model is stationary while the baseline is non-stationary appears to account for most of the cases where the baseline was equal to or better than our model. Whereas the baseline can exploit the fact that certain patterns of production events are associated with particular time periods, in our model the Markov property of the hidden state erases information about how much time has elapsed. In the early game, our model begins predicting production events too early because it models the first several epochs, which all look the same, with a single state that has a high self-transition probability. Later in the game, uncertainty about the time means it has trouble capturing time-dependent "pauses" in production. On the other hand, our model can in principle be extended to full-length games, while our baseline method is too coarse to be useful much beyond the opening.

Another weakness of our model is that it incorporates no explicit prior knowledge about configurations of unit counts. For example, there will almost never be two Observatories, but our model can only account for this by designing the state transition matrix to visit the Observatory-producing state only once. Accuracy could be improved by making production of a unit at time t (P_t^i) dependent on the count of that unit at time $t-1$ (U_{t-1}^i), with a corresponding increase in the complexity of the latent variable portion of the model. The situation worsens if we want to incorporate prerequisite relationships, as these cause the counting processes for *different* unit types to become coupled, destroying the conditional independences that we leveraged for efficient inference.

5 FUTURE WORK

A significant future challenge is to devise a model that can perform well in a full-length game. The space of possible strategies expands greatly as the game runs longer and the actions of the opponent begin to influence one's own decisions. Naively extending the current model by adding states is likely to prove intractable. One possibility is to try to exploit hierarchical structure in strategies to reduce the strategy space. We suspect that strategic decisions take place on multiple time scales—broad objectives at the top level, and smaller steps necessary to achieve them at the bottom. A model of a full game will need to incorporate a model of resource flow in order to reason effectively about production rates. It will also need to account for changes in the opponent's strategy in response to our actions.

We are also interested in applying opponent models to optimize scouting policies. As we saw in Figure 2, human players have converged on at least one time period in which the tradeoff between probability of scouting success and expected information gain is at an optimum. An agent could use our model to determine *when* important information is likely to be available. Modeling the probability that a scouting action will succeed in acquiring information, assuming that the information is there, is a challenging problem in itself.

Acknowledgements

This work was made possible by the efforts of Mark Udarbe and Thao-Trang Hoang in assembling and labeling our dataset.

This work was partly funded by ARO grant W911NF-08-1-0242. The views and conclusions contained in this document are those of the authors and do not necessarily represent the official policies of ARO or the United States Government. Jesse Hostetler is partly supported by a scholarship from the ARCS foundation of Portland, OR.

References

BWAPI (2012). bwapi: An api for interacting with Starcraft: Broodwar (1.16.1). [Online] `https://code.google.com/p/bwapi/`.

Dean, T. and Kanazawa, K. (1988). Probabilistic temporal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 524–529, Cambridge, Massachusetts. MIT Press.

Dereszynski, E., Hostetler, J., Fern, A., Dietterich, T., Hoang, T., and Udarbe, M. (2011). Learning probabilistic behavior models in real-time strategy games. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

Doucet, A., de Freitas, N., Murphy, K., and Russell, S. (2000). Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 176–183, San Francisco, CA. Morgan Kaufmann.

Haseman, J. and Kupper, L. (1979). Analysis of dichotomous response data from certain toxicological experiments. *Biometrics*, pages 281–293.

Hsieh, J. and Sun, C. (2008). Building a player strategy model by analyzing replays of real-time strategy games. In *IEEE International Joint Conference on Neural Networks (IJCNN), 2008.*, pages 3106–3111. IEEE.

Hutchinson, R., Liu, L., and Dietterich, T. (2011). Incorporating boosted regression trees into ecological latent variable models. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Lesnoff, M., Lancelot, and R. (2010). *aod: Analysis of Overdispersed Data*. R package version 1.2.

MacKenzie, D., Nichols, J., Lachman, G., Droege, S., Andrew Royle, J., and Langtimm, C. (2002). Estimating site occupancy rates when detection probabilities are less than one. *Ecology*, 83(8):2248–2255.

Murphy, K. (2000). Bayesian map learning in dynamic environments. In *In Neural Info. Proc. Systems (NIPS)*, pages 1015–1021. MIT Press.

Perkins, L. (2010). Terrain analysis in real-time strategy games: An integrated approach to choke point detection and region decomposition. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Ponsen, M., Spronck, P., Muñoz-Avila, H., and Aha, D. (2007). Knowledge acquisition for adaptive game AI. *Science of Computer Programming*, 67(1):59–75.

Rabiner, L. R. (1990). A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Schadd, F., Bakkes, S., and Spronck, P. (2007). Opponent modeling in real-time strategy games. In *8th International Conference on Intelligent Games and Simulation (GAME-ON) 2007*, pages 61–68.

Synnaeve, G. and Bessière, P. (2011). A Bayesian model for opening prediction in RTS games with application to Starcraft. In *IEEE Conference on Computational Intelligence and Games (CIG), 2011*, pages 281–288. IEEE.

Weber, B. and Mateas, M. (2009). A data mining approach to strategy prediction. In *IEEE Symposium on Computational Intelligence and Games (CIG), 2009*, pages 140–147. IEEE.

Optimally-Weighted Herding is Bayesian Quadrature

Ferenc Huszár
Department of Engineering
Cambridge University
fh277@cam.ac.uk

David Duvenaud
Department of Engineering
Cambridge University
dkd23@cam.ac.uk

Abstract

Herding and kernel herding are deterministic methods of choosing samples which summarise a probability distribution. A related task is choosing samples for estimating integrals using Bayesian quadrature. We show that the criterion minimised when selecting samples in kernel herding is equivalent to the posterior variance in Bayesian quadrature. We then show that sequential Bayesian quadrature can be viewed as a weighted version of kernel herding which achieves performance superior to any other weighted herding method. We demonstrate empirically a rate of convergence faster than $\mathcal{O}(1/N)$. Our results also imply an upper bound on the empirical error of the Bayesian quadrature estimate.

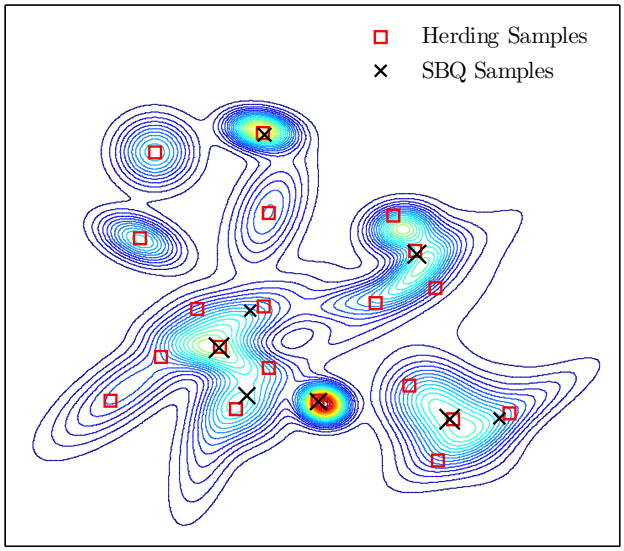


Figure 1: The first 8 samples from sequential Bayesian quadrature, versus the first 20 samples from herding. Only 8 weighted SBQ samples are needed to give an estimator with the same maximum mean discrepancy as using 20 herding samples with uniform weights. Relative sizes of samples indicate their relative weights.

1 INTRODUCTION

The problem: Integrals A common problem in statistical machine learning is to compute expectations of functions over probability distributions of the form:

$$Z_{f,p} = \int f(x)p(x)dx \tag{1}$$

Examples include computing marginal distributions, making predictions marginalizing over parameters, or computing the Bayes risk in a decision problem. In this paper we assume that the distribution $p(x)$ is known in analytic form, and $f(x)$ can be evaluated at arbitrary locations.

Monte Carlo methods produce random samples from the distribution p and then approximate the integral by taking the empirical mean $\hat{Z} = \frac{1}{N}\sum_{n=1}^{N} f_{x_n}$ of the function evaluated at those points. This non-deterministic estimate converges at a rate $\mathcal{O}(\frac{1}{\sqrt{N}})$.

When exact sampling from p is impossible or impractical, Markov chain Monte Carlo (MCMC) methods are often used. MCMC methods can be applied to almost any problem but convergence of the estimate depends on several factors and is hard to estimate (Cowles & Carlin, 1996). The focus of this paper is on quasi-Monte Carlo methods that – instead of sampling randomly – produce a set of pseudo-samples in a deterministic fashion. These methods operate by directly minimising some sort of discrepancy between the empirical distribution of pseudo-samples and the target distribution. Whenever these methods are applicable, they achieve convergence rates superior to the $\mathcal{O}(\frac{1}{\sqrt{N}})$ rate typical of random sampling.

In this paper we highlight and explore the connections between two deterministic sampling and integration methods: Bayesian quadrature (BQ) (O'Hagan, 1991; Rasmussen & Ghahramani, 2003) (also known as Bayesian Monte Carlo) and kernel herding (Chen et al., 2010). Bayesian quadrature estimates integral (1) by inferring a posterior distribution over f conditioned on the observed evaluations f_{x_n}, and then computing the posterior expectation of $Z_{f,p}$. The points where the function should be evaluated can be found via Bayesian experimental design, providing a deterministic procedure for selecting sample locations.

Herding, proposed recently by Chen et al. (2010), produces pseudosamples by minimising the discrepancy of moments between the sample set and the target distribution. Similarly to traditional Monte Carlo, an estimate is formed by taking the empirical mean over samples $\hat{Z} = \frac{1}{N}\sum_{n=1}^{N} f_{x_n}$. Under certain assumptions, herding has provably fast, $\mathcal{O}(\frac{1}{N})$ convergence rates in the parametric case, and has demonstrated strong empirical performance in a variety of tasks.

Summary of contributions In this paper, we make two main contributions. First, we show that the Maximum Mean Discrepancy (MMD) criterion used to choose samples in kernel herding is identical to the expected error in the estimate of the integral $Z_{f,p}$ under a Gaussian process prior for f. This expected error is the criterion being minimized when choosing samples for Bayesian quadrature. Because Bayesian quadrature assigns different weights to each of the observed function values $f(x)$, we can view Bayesian quadrature as a weighted version of kernel herding. We show that these weights are optimal in a minimax sense over all functions in the Hilbert space defined by our kernel. This implies that Bayesian quadrature dominates uniformly-weighted kernel herding and other non-optimally weighted herding in rate of convergence.

Second, we show that minimising the MMD, when using BQ weights is closely related to the sparse dictionary selection problem studied in (Krause & Cevher, 2010), and therefore is approximately submodular with respect to the samples chosen. This allows us to reason about the performance of greedy forward selection algorithms for Bayesian Quadrature. We call this greedy method Sequential Bayesian Quadrature (SBQ).

We then demonstrate empirically the relative performance of herding, i.i.d random sampling, and SBQ, and demonstrate that SBQ attains a rate of convergence faster than $\mathcal{O}(1/N)$.

2 HERDING

Herding was introduced by Welling (2009) as a method for generating pseudo-samples from a distribution in such a way that certain nonlinear moments of the sample set closely match those of the target distribution. The empirical mean $\frac{1}{N}\sum_{n=1}^{N} f_{x_n}$ over these pseudosamples is then used to estimate integral (1).

2.1 Maximum Mean Discrepancy

For selecting pseudosamples, herding relies on an objective based on the maximum mean discrepancy (MMD; Sriperumbudur et al., 2010). MMD measures the divergence between two distributions, p and q with respect to a class of integrand functions $\mathcal{F}$ as follows:

$$\mathrm{MMD}_{\mathcal{F}}(p,q) = \sup_{f\in\mathcal{F}} \left| \int f_x p(x)dx - \int f_x q(x)dx \right| \quad (2)$$

Intuitively, if two distributions are close in the MMD sense, then no matter which function f we choose from $\mathcal{F}$, the difference in its integral over p or q should be small. A particularly interesting case is when the function class $\mathcal{F}$ is functions of unit norm from a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. In this case, the MMD between two distributions can be conveniently expressed using expectations of the associated kernel $k(x,x')$ only (Sriperumbudur et al., 2010):

$$MMD^2_{\mathcal{H}}(p,q) = \sup_{\substack{f\in\mathcal{H}\\ \|f\|_{\mathcal{H}}=1}} \left| \int f_x p(x)dx - \int f_x q(x)dx \right|^2 \quad (3)$$

$$= \|\mu_p - \mu_q\|^2_{\mathcal{H}} \quad (4)$$

$$= \iint k(x,y)p(x)p(y)dxdy - 2\iint k(x,y)p(x)q(y)dxdy + \iint k(x,y)q(x)q(y)dxdy, \quad (5)$$

where in the above formula $\mu_p = \int \phi(x)p(x)dx \in \mathcal{H}$ denotes the *mean element* associated with the distribution p. For characteristic kernels, such as the Gaussian kernel, the mapping between a distribution and its mean element is bijective. As a consequence $\mathrm{MMD}_{\mathcal{H}}(p,q) = 0$ if and only if $p = q$, making it a powerful measure of divergence.

Herding uses maximum mean discrepancy to evaluate of how well the sample set $\{x_1,\ldots,x_N\}$ represents the target distribution p:

$$\epsilon_{herding}\left(\{x_1,\ldots,x_N\}\right) = \mathrm{MMD}_{\mathcal{H}}\left(p, \frac{1}{N}\sum_{n=1}^{N}\delta_{x_n}\right) \tag{6}$$

$$= \iint k(x,y)p(x)p(y)dxdy$$
$$-2\frac{1}{N}\sum_{n=1}^{N}\int k(x,x_n)p(x)dx + \frac{1}{N^2}\sum_{n,m=1}^{N}k(x_n,x_m) \tag{7}$$

The herding procedure greedily minimizes its objective $\epsilon_{herding}\left(\{x_1,\ldots,x_N\}\right)$, adding pseudosamples x_n one at a time. When selecting the $n+1$-st pseudosample:

$$x_{n+1} \leftarrow \operatorname*{argmin}_{x\in\mathcal{X}} \epsilon_{herding}\left(\{x_1,\ldots,x_n,x\}\right) \tag{8}$$
$$= \operatorname*{argmax}_{x\in\mathcal{X}} 2\mathbb{E}_{x'\sim p}\left[k(x,x')\right] - \frac{1}{n+1}\sum_{m=1}^{n}k(x,x_m),$$

assuming $k(x,x) = \text{const}$. The formula (8) admits an intuitive interpretation: the first term encourages sampling in areas with high mass under the target distribution $p(x)$. The second term discourages sampling at points close to existing samples.

Evaluating (8) requires us to compute $\mathbb{E}_{x'\sim p}\left[k(x,x')\right]$, that is to integrate the kernel against the target distribution. Throughout the paper we will assume that these integrals can be computed in closed form. Whilst the integration can indeed be carried out analytically in several cases (Song et al., 2008; Chen et al., 2010), this requirement is the most pertinent limitation on applications of kernel herding, Bayesian quadrature and related algorithms.

2.2 Complexity and Convergence Rates

Criterion (8) can be evaluated in only $\mathcal{O}(n)$ time. Adding these up for all subsequent samples, and assuming that optimisation in each step has $\mathcal{O}(1)$ complexity, producing N pseudosamples via kernel herding costs $\mathcal{O}(N^2)$ operations in total.

In finite dimensional Hilbert spaces, the herding algorithm has been shown to reduce MMD at a rate $\mathcal{O}(\frac{1}{N})$, which compares favourably with the $\mathcal{O}(\frac{1}{\sqrt{N}})$ rate obtained by non-deterministic Monte Carlo samplers. However, as pointed out by Bach et al. (2012), this fast convergence is not guaranteed in infinite dimensional Hilbert spaces, such as the RKHS corresponding to the Gaussian kernel.

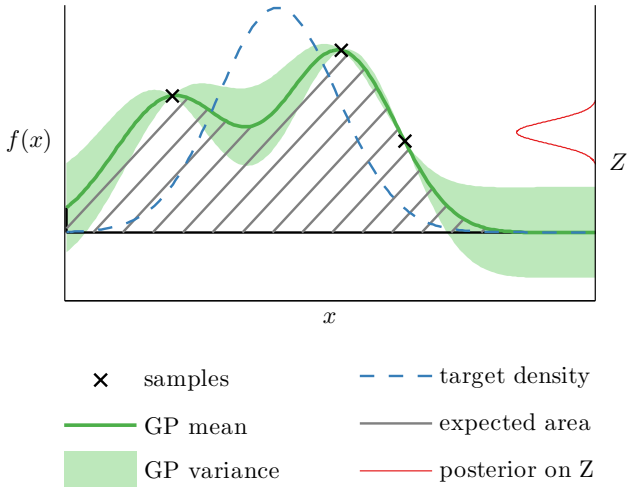


Figure 2: An illustration of Bayesian Quadrature. The function $f(x)$ is sampled at a set of input locations. This induces a Gaussian process posterior distribution on f, which is integrated in closed form against the target density, $p(x)$. Since the amount of volume under f is uncertain, this gives rise to a (Gaussian) posterior distribution over $Z_{f,p}$.

3 BAYESIAN QUADRATURE

So far, we have only considered integration methods in which the integral (1) is approximated by the empirical mean of the function evaluated at some set of samples, or pseudo-samples. Equivalently, we can say that Monte Carlo and herding both assign an equal $\frac{1}{N}$ weight to each of the samples.

In (Rasmussen & Ghahramani, 2003), an alternate method is proposed: Bayesian Monte Carlo, or Bayesian quadrature (BQ). BQ puts a prior distribution on f, then estimates integral (1) by inferring a posterior distribution over the function f, conditioned on the observations $f(x_n)$ at some query points x_n. The posterior distribution over f then implies a distribution over $Z_{f,p}$. This method allows us to choose sample locations x_n in any desired manner. See Figure 2 for an illustration of Bayesian Quadrature.

3.1 BQ Estimator

Here we derive the BQ estimate of (1), after conditioning on function evaluations $f(x_1)\ldots f(x_N)$, denoted as $f(X)$. The Bayesian solution implies a distribution over $Z_{f,p}$. The mean of this distribution, $\mathbb{E}\left[Z\right]$ is the optimal Bayesian estimator for a squared loss.

For simplicity, f is assigned a Gaussian process prior with kernel function k and mean 0. This assumption is very similar to the one made by kernel herding in Eqn. (7).

After conditioning on f_x, we obtain a closed-form posterior over f:

$$p(f(x_\star)|f(X)) = \mathcal{N}\big(f_{x_\star}|\bar{f}(x_\star), \mathrm{cov}(x_\star, x'_\star)\big) \quad (9)$$

where

$$\bar{f}(x_\star) = k(x_\star, X)K^{-1}f(X) \quad (10)$$

$$\mathrm{cov}(x_\star, x'_\star) = k(x_\star, x_\star) - k(x_\star, X)K^{-1}k(X, x_\star) \quad (11)$$

and $K = k(X, X)$. Conveniently, the GP posterior allows us to compute the expectation of (1) in closed form:

$$\mathbb{E}_{\mathrm{GP}}[Z] = \mathbb{E}_{\mathrm{GP}}\left[\int f(x)p(x)dx\right] \quad (12)$$

$$= \iint f(x)p(f(x)|f(X))p(x)dxdf \quad (13)$$

$$= \int \bar{f}(x)p(x)dx \quad (14)$$

$$= \left[\int k(x, X)p(x)dx\right] K^{-1}f(X) \quad (15)$$

$$= \mathbf{z}^T K^{-1} f(X) \quad (16)$$

where

$$z_n = \int k(x, x_n)p(x)dx = \mathbb{E}_{x'\sim p}[k(x_n, x')]. \quad (17)$$

Conveniently, as in kernel herding, the desired expectation of $Z_{f,p}$ is simply a linear combination of observed function values $f(x)$:

$$\mathbb{E}_{\mathrm{GP}}[Z] = \mathbf{z}^T K^{-1} f(X) \quad (18)$$

$$= \sum_n w_{\mathrm{BQ}}^{(n)} f_{x_n}) \quad (19)$$

where

$$w_{\mathrm{BQ}}^{(n)} = \sum_m \mathbf{z}_j^T K_{nm}^{-1} \quad (20)$$

Thus, we can view the BQ estimate as a weighted version of the herding estimate. Interestingly, the weights w_{BQ} do not need to sum to 1, and are not even necessarily positive.

3.1.1 Non-normalized and Negative Weights

When weighting samples, it is often assumed, or enforced (as in Bach et al., 2012; Song et al., 2008), that the weights w form a probability distribution. However, there is no technical reason for this requirement, and in fact, the optimal weights do not have this property. Figure 3 shows a representative set of 100 BQ weights chosen on samples representing the distribution in figure 1. There are several negative weights, and the sum of all weights is 0.93.

Figure 4 demonstrates that, in general, the sum of the Bayesian weights exhibits shrinkage when the number of samples is small.

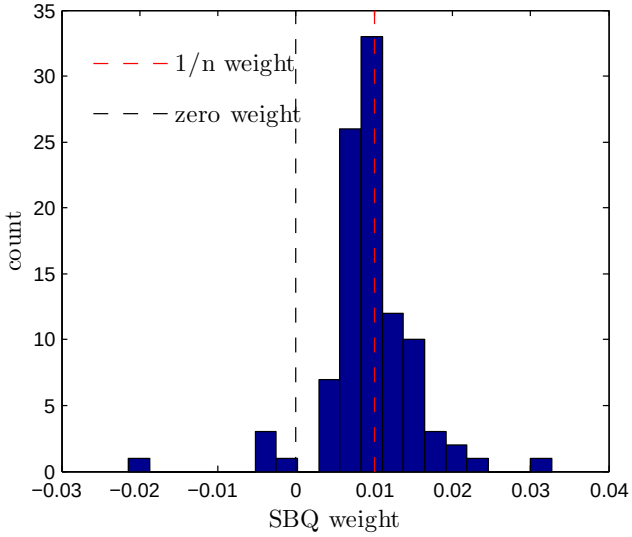


Figure 3: A set of optimal weights given by BQ, after 100 SBQ samples were selected on the distribution shown in Figure 1. Note that the optimal weights are spread away from the uniform weight ($\frac{1}{N}$), and that some weights are even negative. The sum of these weights is 0.93.

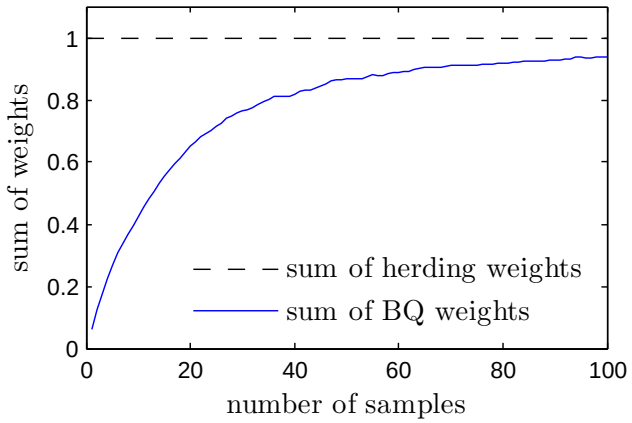


Figure 4: An example of Bayesian shrinkage in the sample weights. In this example, the kernel width is approximately 1/20 the width of the distribution being considered. Because the prior over functions is zero mean, in the small sample case the weights are shrunk towards zero. The weights given by simple Monte Carlo and herding do not exhibit shrinkage.

3.2 Optimal sampling for BQ

Bayesian quadrature provides not only a mean estimate of $Z_{f,p}$, but a full Gaussian posterior distribution. The variance of this distribution $\mathbb{V}\left[Z_{f,p}|f_{x_1},\ldots,f_{x_N}\right]$ quantifies our uncertainty in the estimate. When selecting locations to evaluate the function f, minimising the posterior variance is a sensible strategy. Below, we give a closed form formula for the posterior variance of $Z_{f,p}$, conditioned on the observations $f_{x_1}\ldots f_{x_N}$, which we will denote by ϵ^2_{BQ}. For a longer derivation, see Rasmussen & Ghahramani (2003).

$$\epsilon^2_{\text{BQ}}(x_1,\ldots,x_N) = \mathbb{V}\left[Z_{f,p}|f_{x_1},\ldots,f_{x_N}\right] \tag{21}$$
$$= \mathbb{E}_{x,x'\sim p}\left[k(x,x')\right] - \mathbf{z}^T K^{-1}\mathbf{z}, \tag{22}$$

where $\mathbf{z}_n = \mathbb{E}_{x'\sim p}\left[k(x_n,x')\right]$ as before. Perhaps surprisingly, the posterior variance of $Z_{f,p}$ does not depend on the observed function values, only on the location x_n of samples. A similar independence is observed in other optimal experimental design problems involving Gaussian processes (Krause et al., 2006). This allows the optimal samples to be computed ahead of time, before observing any values of f at all (Minka, 2000).

We can contrast the BQ objective ϵ^2_{BQ} in (22) to the objective being minimized in herding, $\epsilon^2_{herding}$ of equation (7). Just like $\epsilon^2_{herding}$, ϵ^2_{BQ} expresses a trade-off between accuracy and diversity of samples. On the one hand, as samples get close to high density regions under p, the values in $\mathbf{z}$ increase, which results in decreasing variance. On the other hand, as samples get closer to each other, eigenvalues of K increase, resulting in an increase in variance.

In a similar fashion to herding, we may use a greedy method to minimise ϵ^2_{BQ}, adding one sample at a time. We will call this algorithm *Sequential Bayesian Quadrature* (SBQ):

$$x_{n+1} \leftarrow \operatorname*{argmin}_{x\in\mathcal{X}} \epsilon_{\text{BQ}}\left(\{x_1,\ldots,x_n,x\}\right) \tag{23}$$

Using incremental updates to the Cholesky factor, the criterion can be evaluated in $\mathcal{O}(n^2)$ time. Iteratively selecting N samples thus takes $\mathcal{O}(N^3)$ time, assuming optimisation can be done on $\mathcal{O}(1)$ time.

4 RELATING $\mathbb{V}\left[Z_{f,p}\right]$ TO MMD

The similarity in the behaviour of $\epsilon^2_{herding}$ and ϵ^2_{BQ} is not a coincidence, the two quantities are closely related to each other, and to MMD.

Proposition 1. *The expected variance in the Bayesian quadrature ϵ^2_{BQ} is the maximum mean discrepancy between the target distribution p and $q_{\text{BQ}}(x) = \sum_{n=1}^N w^{(n)}_{\text{BQ}}\delta_{x_n}(x)$*

Proof. The proof involves invoking the representer theorem, using bilinearity of scalar products and the fact that if f is a standard Gaussian process then $\forall g\in\mathcal{H}: \langle f,g\rangle \sim \mathcal{N}(0,\|g\|_{\mathcal{H}})$:

$$\mathbb{V}\left[Z_{f,p}|f_{x_1},\ldots,f_{x_N}\right] = \tag{24}$$
$$= \mathbb{E}_{f\sim GP}\left(\int f(x)p(x)dx - \sum_{n=1}^N w^{(n)}_{\text{BQ}} f(x_n)\right)^2 \tag{25}$$
$$= \mathbb{E}_{f\sim GP}\left(\int \langle f,\phi(x)\rangle\, p(x)dx - \sum_{n=1}^N w^{(n)}_{\text{BQ}}\,\langle f,\phi(x_n)\rangle\right)^2 \tag{26}$$
$$= \mathbb{E}_{f\sim GP}\left\langle f, \int \phi(x)p(x)dx - \sum_{n=1}^N w^{(n)}_{\text{BQ}}\phi(x_n)\right\rangle^2 \tag{27}$$
$$= \left\|\mu_p - \mu_{q_{\text{BQ}}}\right\|^2_{\mathcal{H}} \tag{28}$$
$$= \text{MMD}^2(p,q_{\text{BQ}}) \tag{29}$$

□

We know that the the posterior mean $\mathbb{E}_{\text{GP}}\left[Z_{f,p}|f_1,\ldots,f_N\right]$ is a Bayes estimator and has therefore the minimal expected squared error amongst all estimators. This allows us to further rewrite ϵ^2_{BQ} into the following minimax forms:

$$\epsilon^2_{\text{BQ}} = \sup_{\substack{f\in\mathcal{H}\\ \|f\|_{\mathcal{H}}\mathcal{H}=1}} \left|\int f_x p(x)dx - \sum_{n=1}^N w^{(n)}_{\text{BQ}} f_{x_n}\right|^2 \tag{30}$$
$$= \inf_{\hat{Z}:\mathcal{X}^N\mapsto\mathbb{R}} \sup_{\substack{f\in\mathcal{H}\\ \|f\|_{\mathcal{H}}\mathcal{H}=1}} \left|Z - \hat{Z}\left(f_{x_1},\ldots,f_{x_N}\right)\right|^2 \tag{31}$$
$$= \inf_{\boldsymbol{w}\in\mathbb{R}^N} \sup_{\substack{f\in\mathcal{H}\\ \|f\|_{\mathcal{H}}\mathcal{H}=1}} \left|\int f_x p(x)dx - \sum_{n=1}^N w_n f_{x_n}\right|^2 \tag{32}$$

Looking at ϵ^2_{BQ} this way, we may discover the deep similarity to the criterion $\epsilon^2_{herding}$ that kernel herding minimises. Optimal sampling for Bayesian quadrature minimises the same objective as kernel herding, but with the uniform $\frac{1}{N}$ weights replaced by the optimal weights. As a corollary

$$\epsilon^2_{\text{BQ}}(x_1,\ldots,x_N) \le \epsilon^2_{KH}(x_1,\ldots,x_N) \tag{33}$$

It is interesting that ϵ^2_{BQ} has both a Bayesian interpretation as posterior variance under a Gaussian process prior, and a frequentist interpretation as a minimax bound on estimation error with respect to an RKHS.

5 SUBMODULARITY

In this section, we use the concept of approximate submodularity (Krause & Cevher, 2010), in order to study convergence properties of SBQ.

A set function $s : 2^{\mathcal{X}} \mapsto \mathbb{R}$ is *submodular* if, for all $A \subseteq B \subseteq \mathcal{X}$ and $\forall x \in \mathcal{X}$

$$s(A \cup \{x\}) - s(A) \geq s(B \cup \{x\}) - s(B) \tag{34}$$

Intuitively, submodularity is a diminishing returns property: adding an element to a smaller set has larger relative effect than adding it to a larger set. A key result (see e. g. Krause & Cevher, 2010, and references therein) is that greedily maximising a submodular function is guaranteed not to differ from the optimal strategy by more than a constant factor of $(1 - \frac{1}{e})$.

Herding and SBQ are examples of greedy algorithms optimising set functions: they add each pseudosamples in such a way as to minimize the instantaneous reduction in MMD. So it is intuitive to check whether the objective functions these methods minimise are submodular. Unfortunately, neither $\epsilon_{herding}$, nor ϵ_{BQ} satisfies all conditions necessary for submodularity. However, noting that SBQ is identical to the sparse dictionary selection problem studied in detail by Krause & Cevher (2010), we can conclude that SBQ satisfies a weaker condition called *approximate submodularity*.

A set function $s : 2^{\mathcal{X}} \mapsto \mathbb{R}$ is *approximately submodular* with constant $\epsilon > 0$, if for all $A \subseteq B \subseteq \mathcal{X}$ and $\forall x \in \mathcal{X}$

$$s(A \cup \{x\}) - s(A) \geq s(B \cup \{x\}) - s(B) - \epsilon \tag{35}$$

Proposition 2. *$\epsilon^2_{\text{BQ}}(\emptyset) - \epsilon^2_{\text{BQ}}(\cdot)$ is weakly a weakly submodular set function with constant $\epsilon < 4r$, where r is the incoherency*

$$r = \max_{x, x' \in \mathcal{P} \subseteq \mathcal{X}} \frac{k(x, x')}{\sqrt{k(x, x) k(x', x')}} \tag{36}$$

Proof. By the definition of MMD we can see that $-\epsilon^2_{\text{BQ}} = \inf_{w \in \mathbb{R}^N} \left\| \mu_p - \sum_{n=1}^N w^{(n)}_{\text{BQ}} k(\cdot, x_n) \right\|^2_{\mathcal{H}}$ is the negative squared distance between the mean element μ_p and its projection onto the subspace spanned by the elements $k(\cdot, x_n)$. Substituting $k = 1$ into Theorem 1 of Krause & Cevher (2010) concludes the proof. □

Unfortunately, weak submodularity does not provide the strong near-optimality guarantees as submodularity does . If $s : 2^{\mathcal{X}} \mapsto \mathbb{R}$ is a weakly submodular function with constant ϵ, and $|\mathcal{A}_n| = n$ is the result of greedy optimisation of s, then

$$s(\mathcal{A}_n) \geq \left(1 - \frac{1}{e}\right) \max_{|\mathcal{A}| \leq n} s(\mathcal{A}) - n\epsilon \tag{37}$$

As pointed out by Krause & Cevher (2010), this guarantee is very weak, as in our case the objective function $\epsilon^2_{\text{BQ}}(\emptyset) - \epsilon^2_{\text{BQ}}(\cdot)$ is upper bounded by a constant. However, establishing a connection between SBQ and sparse dictionary selection problem opens up interesting directions for future research, and it may be possible to apply algorithms and theory developed for sparse dictionary selection to kernel-based quasi-Monte Carlo methods.

6 EXPERIMENTS

In this section, we examine empirically the rates of convergence of sequential Bayesian quadrature and herding. We examine both the expected error rates, and the empirical error rates.

In all experiments, the target distribution p is chosen a 2D mixture of 20 Gaussians, whose equiprobability contours are shown in Figure 1. To ensure a comparison fair to herding, the target distribution, and the kernel used by both methods, correspond exactly to the one used in (Chen et al., 2010, Fig. 1). For experimental simplicity, each of the sequential sampling algorithms minimizes the next sample location from a pool of 10000 locations randomly drawn from the base distribution. In practice, one would run a local optimizer from each of these candidate locations, however in our experiments we found that this did not make a significant difference in the sample locations chosen.

6.1 Matching a distribution

We first extend an experiment from (Chen et al., 2010) designed to illustrate the mode-seeking behavior of herding in comparison to random samples. In that experiment, it is shown that a small number of i. i. d. samples drawn from a multimodal distribution will tend to, by chance, assign too many samples to some modes, and too few to some other modes. In contrast, herding places 'super-samples' in such a way as to avoid regions already well-represented, and seeks modes that are under-represented.

We demonstrate that although herding improves upon i. i. d. sampling, the uniform weighting of super-samples leads to sub-optimal performance. Figure 1 shows the first 20 samples chosen by kernel herding, in comparison with the first 8 samples chosen by SBQ. By weighting the 8 SBQ samples by the quadrature weights in (20), we can obtain the same expected loss as by using the 20 uniformly-weighted herding samples. Figure 5 shows MMD versus the number of samples added, on the distribution shown in Figure 1. We can see that in all cases, SBQ dominates herding. It appears that SBQ converges at a faster rate than $\mathcal{O}(1/N)$, although the

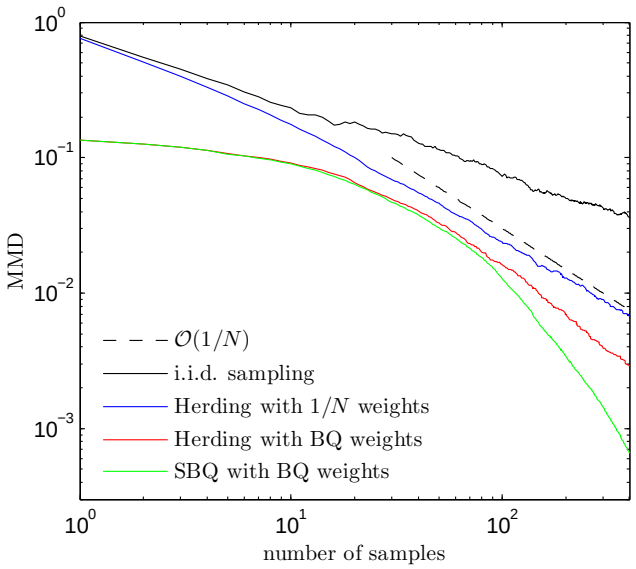


Figure 5: The maximum mean discrepancy, or expected error of several different quadrature methods. Herding appears to approach a rate close to $\mathcal{O}(1/N)$. SBQ appears to attain a faster, but unknown rate.

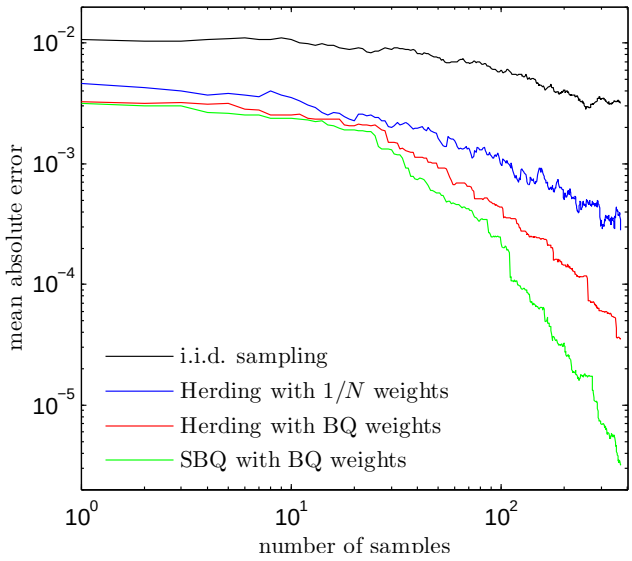


Figure 6: Within-model error: The empirical error rate in estimating $Z_{f,p}$, for several different sampling methods, averaged over 250 functions randomly drawn from the RKHS corresponding to the kernel used.

form of this rate is unknown.

There are two differences between herding and SBQ: SBQ chooses samples according to a different criterion, and also weights those samples differently. We may ask whether the sample locations or the weights are contributing more to the faster convergence of SBQ. Indeed, in Figure 1 we observe that the samples selected by SBQ are quite similar to the samples selected by kernel herding. To answer this question, we also plot in Figure 5 the performance of a fourth method, which selects samples using herding, but later re-weights the herding samples with BQ weights. Initially, this method attains similar performance to SBQ, but as the number of samples increases, SBQ attains a better rate of convergence. This result indicates that the different sample locations chosen by SBQ, and not only the optimal weights, are responsible for the increased convergence rate of SBQ.

6.2 Estimating Integrals

We then examined the empirical performance of the different estimators at estimating integrals of real functions. To begin with, we looked at performance on 100 randomly drawn functions, of the form:

$$f(x) = \sum_{i=1}^{10} \alpha_i k(x, c_i) \tag{38}$$

where

$$\|f\|_{\mathcal{H}}^2 = \sum_{i=1}^{10}\sum_{j=1}^{10} \alpha_i \alpha_j k(c_i, c_j) = 1 \tag{39}$$

That is, these functions belonged exactly to the unit ball of the RKHS defined by the kernel $k(x, x')$ used to model them. Figure 6 shows the empirical error versus the number of samples, on the distribution shown in Figure 1. The empirical rates attained by the method appear to be similar to the MMD rates in Figure 5.

By definition, MMD provides a upper bound on the estimation error in the integral of any function in the unit ball of the RKHS (Eqn. (3)), including the Bayesian estimator, SBQ. Figure 7 demonstrates this quickly decreasing bound on the SBQ empirical error.

6.3 Out-of-model performance

A central assumption underlying SBQ is that the integrand function belongs to the RKHS specified by the kernel. To see how performance is effected if this assumption is violated, we performed empirical tests with functions chosen from outside the RKHS. We drew 100 functions of the form:

$$f(x) = \sum_{i=1}^{10} \alpha_i \exp(-\frac{1}{2}(x - c_i)^T \Sigma_i^{-1}(x - c_i) \tag{40}$$

where each α_i c_i Σ_i were drawn from broad distributions. This ensured that the drawn functions had features such as narrow bumps and ridges which would

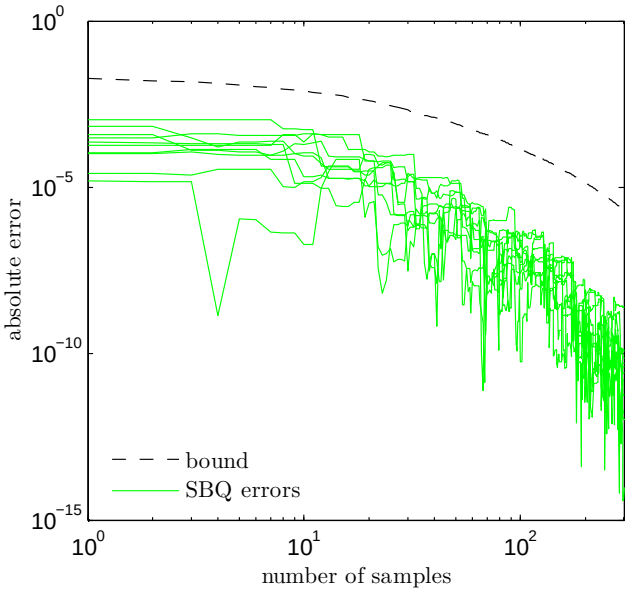


Figure 7: The empirical error rate in estimating $Z_{f,p}$, for the SBQ estimator, on 10 random functions drawn from the RKHS corresponding to the kernel used. Also shown is the upper bound on the error rate implied by the MMD.

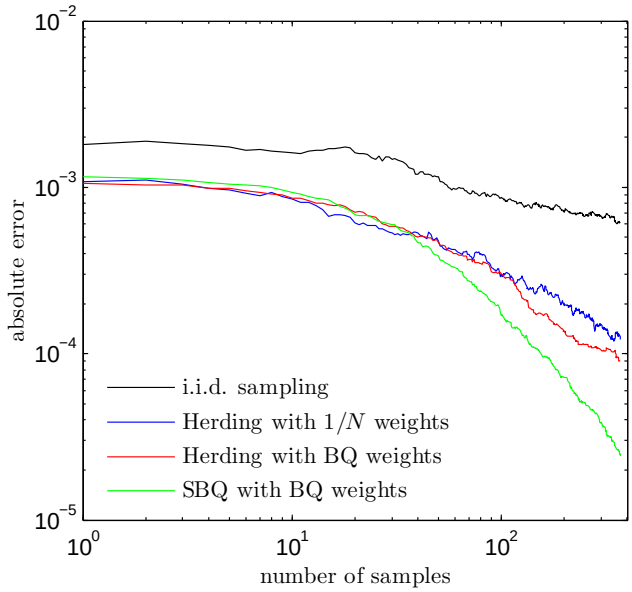


Figure 8: Out-of-model error: The empirical error rates in estimating $Z_{f,p}$, for several different sampling methods, averaged over 250 functions drawn from outside the RKHS corresponding to the kernels used.

not be well modelled by functions belonging to the isotropic kernel defined by k. Figure 8 shows that, on functions drawn from outside the assumed RKHS, relative performance of all methods remains similar.

Code to reproduce all results is available at `http://mlg.eng.cam.ac.uk/duvenaud/`

7 DISCUSSION

7.1 Choice of Kernel

Using herding techniques, we are able to achieve fast convergence on a Hilbert space of *well-behaved* functions, but this fast convergence is at the expense of the estimate not necessarily converging for functions outside this space. If we use a characteristic kernel (Sriperumbudur et al., 2010), such as the exponentiated-quadratic or Laplacian kernels, then convergence in MMD implies weak convergence of q_N to the target distribution. This means that the estimate converges for any bounded measurable function f. The speed of convergence, however, may not be as fast.

Therefore it is crucial that the kernel we choose is representative of the function or functions f we will integrate. For example, in our experiments, the convergence of herding was sensitive to the width of the Gaussian kernel. One of the major weaknesses of kernel methods in general is the difficulty of setting kernel parameters. A key benefit of the Bayesian interpretation of herding and MMD presented in this paper is that it provides a recipe for adapting the Hilbert space to the observations $f(x_n)$. To be precise, we can fit the kernel parameters by maximizing the marginal likelihood of Gaussian process conditioned on the observations. Details can be found in (Rasmussen & Williams, 2006).

7.2 Computational Complexity

While we have shown that Bayesian Quadrature provides the optimal re-weighting of samples, computing the optimal weights comes at an increased computational cost relative to herding. The computational complexity of computing Bayesian quadrature weights for N samples is $\mathcal{O}(N^3)$, due to the necessity of inverting the Gram matrix $K(x, x)$. Using the Woodbury identity, the cost of adding a new sample to an existing set is $\mathcal{O}(N^2)$. For herding, the computational complexity of evaluating a new sample is only $\mathcal{O}(N)$, making the cost of choosing N herding samples $\mathcal{O}(N^2)$. For Monte Carlo sampling, the cost of adding an i.i.d. sample from the target distribution is only $\mathcal{O}(1)$.

The relative computational cost of computing samples and weights using BQ, herding, and sampling must be weighed against the cost of evaluating f at the sample locations. Depending on this trade-off, the three sampling methods form a Pareto frontier over computational speed and estimator accuracy. When computing

method	complexity	rate	guarantee
MCMC	$\mathcal{O}(N)$	variable	ergodic theorem
i.i.d. MC	$\mathcal{O}(N)$	$\frac{1}{\sqrt{N}}$	law of large numbers
herding	$\mathcal{O}(N^2)$	$\frac{1}{\sqrt{N}} \geq \cdot \geq \frac{1}{N}$	(Chen et al., 2010; Bach et al., 2012)
SBQ	$\mathcal{O}(N^3)$	unknown	approximate submodularity

Table 1: A comparison of the rates of convergence and computational complexity of several integration methods.

f is cheap, we may wish to use Monte Carlo methods. In cases where f is computationally costly, we would expect to choose the SBQ method. When f is relatively expensive, but a very large number of samples are required, we may choose to use kernel herding instead. However, because the rate of convergence of SBQ is faster, there may be situations in which the $\mathcal{O}(N^3)$ cost is relatively inexpensive, due to the smaller N required by SBQ to achieve the same accuracy as compared to using other methods.

There also exists the possibility to switch to a less costly sampling algorithm as the number of samples increases. Table 1 summarizes the rates of convergence of all the methods considered here.

8 CONCLUSIONS

In this paper, we have shown three main results: First, we proved that the loss minimized by kernel herding is closely related to the loss minimized by Bayesian quadrature, when selecting sample locations. This implies that sequential Bayesian quadrature can viewed as an optimally-weighted version of kernel herding.

Second, we showed that the loss minimized by the Bayesian method is approximately submodular with respect to the samples chosen, and established connections to the submodular dictionary selection problem studied in (Krause & Cevher, 2010).

Finally, we empirically demonstrated a superior rate of convergence of SBQ over herding, and demonstrated a bound on the empirical error of the Bayesian quadrature estimate.

8.1 Future Work

In section 5, we showed that SBQ is approximately submodular, which provides only weak sub-optimality guarantees of its performance. It would be of interest to further explore the connection between Bayesian Quadrature and the dictionary selection problem to see if algorithms developed for dictionary selection can provide further practical or theoretical developments. The results in section 6, specifically Figure 5, suggest that the convergence rate of SBQ is faster than $\mathcal{O}(1/N)$. However, we are not aware of any work showing what the theoretically optimal rate is. It would be of great interest to determine this optimal rate of convergence for particular classes of kernels.

Acknowledgements

The authors would like to thank Carl Rasmussen and Francis Bach for helpful discussions, and Yutian Chen for his help in reproducing experiments.

References

Bach, F., Lacoste-Julien, S., and Obozinski, G. On the equivalence between herding and conditional gradient algorithms. *Arxiv preprint arXiv:1203.4523*, 2012.

Chen, Y., Welling, M., and Smola, A. Super-samples from kernel herding. UAI, 2010.

Cowles, Mary K. and Carlin, Bradley P. Markov chain monte carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996. ISSN 01621459. doi: 10.2307/2291683. URL http://dx.doi.org/10.2307/2291683.

Krause, A., Guestrin, C., Gupta, A., and Kleinberg, J. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 2–10, Nashville, Tennessee, USA, 2006.

Krause, Andreas and Cevher, Volkan. Submodular dictionary selection for sparse representation. In Fürnkranz, Johannes and Joachims, Thorsten (eds.), *ICML*, pp. 567–574. Omnipress, 2010. ISBN 978-1-60558-907-7.

Minka, T. P. Deriving quadrature rules from Gaussian processes. Technical report, Statistics Department, Carnegie Mellon University, 2000.

O'Hagan, A. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, 29:245–260, 1991.

Rasmussen, C. E. and Ghahramani, Z. Bayesian monte carlo. In Becker, S. and Obermayer, K. (eds.), *Ad-*

vances in Neural Information Processing Systems, volume 15. MIT Press, Cambridge, MA, 2003.

Rasmussen, C.E. and Williams, CKI. Gaussian Processes for Machine Learning. *The MIT Press, Cambridge, MA, USA*, 2006.

Song, Le, Zhang, Xinhua, Smola, Alex, Gretton, Arthur, and Schölkopf, Bernhard. Tailoring density estimation via reproducing kernel moment matching. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pp. 992–999, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390281. URL http://doi.acm.org/10.1145/1390156.1390281.

Sriperumbudur, Bharath K., Gretton, Arthur, Fukumizu, Kenji, Schölkopf, Bernhard, and Lanckriet, Gert R.G. Hilbert space embeddings and metrics on probability measures. *J. Mach. Learn. Res.*, 99:1517–1561, August 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1859890.1859901.

Welling, M. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1121–1128. ACM, 2009.

Causal Discovery of Linear Cyclic Models from Multiple Experimental Data Sets with Overlapping Variables

Antti Hyttinen
HIIT & Dept. of Computer Science
University of Helsinki
Finland

Frederick Eberhardt
Dept. of Philosophy
Carnegie Mellon University
Pittsburgh, PA, USA

Patrik O. Hoyer
HIIT & Dept. of Computer Science
University of Helsinki
Finland

Abstract

Much of scientific data is collected as randomized experiments intervening on some and observing other variables of interest. Quite often, a given phenomenon is investigated in several studies, and different sets of variables are involved in each study. In this article we consider the problem of integrating such knowledge, inferring as much as possible concerning the underlying causal structure with respect to the union of observed variables from such experimental or passive observational overlapping data sets. We do not assume acyclicity or joint causal sufficiency of the underlying data generating model, but we do restrict the causal relationships to be linear and use only second order statistics of the data. We derive conditions for full model identifiability in the most generic case, and provide novel techniques for incorporating an assumption of faithfulness to aid in inference. In each case we seek to establish what is and what is not determined by the data at hand.

1 INTRODUCTION

For many scientific domains different research groups study the same or closely related phenomena. In general the resulting data sets may contain different sets of variables with only partial overlap, and some of the data sets may be passive observational, while others have resulted from experiments intervening on a subset of the variables. In the context of discovering causal pathways, it would be desirable to be able to integrate all the findings of these 'overlapping' experimental and observational data sets, not only to gain an overall view, but, where possible, to also detect causal relations that were not identified in any individual study.

In the literature on causal discovery methods this problem has been addressed in two complementary, yet so far, independent ways. On the one hand a variety of algorithms have been developed that integrate data from *overlapping* but purely *passive observational* data sets (Tillman et al., 2009; Triantafillou et al., 2010; Tillman and Spirtes, 2011). On the other hand there are several procedures that combine multiple *experimental* studies on the *same* set of variables (Cooper and Yoo, 1999; Tong and Koller, 2001; Murphy, 2001; Eaton and Murphy, 2007; He and Geng, 2008; Schmidt and Murphy, 2009; Eberhardt et al., 2010; Hyttinen et al., 2010).

In this article we present methods that combine the two approaches: we consider data sets that contain passive observational or experimental measures of overlapping sets of variables. We do *not* assume that the set of variables is jointly causally sufficient; in other words we allow for the possible existence of common causes of the measured variables that are not measured in *any* of the available data sets. Moreover, unlike any of the previous work on overlapping data sets, we do *not* assume that the underlying causal structure is acyclic. To ensure that we have a well-defined representation of the *cyclic* relations, we assume that the causal relations are *linear* (see Section 2). The assumption of linearity also enables the use of quantitative constraints that can lead to the identifiability of the underlying causal structure in cases which are non-identifiable without such a parametric assumption (Hyttinen et al., 2011). We provide necessary and sufficient conditions under which the true causal model can be identified, but given the weak set of background assumptions, these are inevitably very demanding. In general there may only be a very limited number of experimental data sets, and so the remaining underdetermination can still be quite substantial. We then provide a novel approach for incorporating the assumption of *faithfulness* (Spirtes et al., 1993) to aid in model search for this family of linear models. We show that, for sparse graphs, a significant amount of structure can be inferred using this framework.

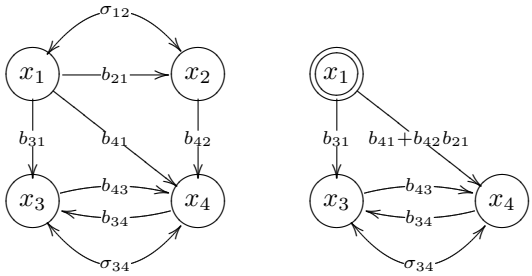


Figure 1: An example of a linear cyclic model with latent variables (left) and the corresponding manipulated model in an experiment where x_1 is intervened, x_3, x_4 are observed and x_2 is hidden (right). Single-headed arrows denote non-zero direct effects while double-headed arrows represent the existence of latent confounders. The disturbance variables are omitted to improve readability.

This paper is structured as follows. We formalize the problem in Section 2, by defining the model family we are considering, specifying the representation used for overlapping data sets, and formulating the inference problem. Then, in Section 3, we derive necessary and sufficient conditions for the identifiability of the model, in the absence of extra assumptions. In Section 4, we consider an approach to integrate simple faithfulness constraints, which we then extend to a more general and powerful method in Sections 5-6. Simulations are provided in Section 7, and some conclusions in Section 8.

2 MODEL AND PROBLEM DEFINITION

We use the following formal representation of the setting with overlapping data sets: Given data sets $\mathcal{D}_1, \ldots, \mathcal{D}_K$ with measurements on the sets of variables $\mathcal{V}_1, \ldots, \mathcal{V}_K$, respectively, let the joint set of variables be $\mathcal{V} = \{x_1, \ldots, x_n\} = \bigcup_{k=1,\ldots,K} \mathcal{V}_k$. Obviously, the individual sets $\mathcal{V}_k$ of variables in $\mathcal{D}_k$ will generally not be causally sufficient, but we also allow the *joint* set of variables $\mathcal{V}$ to be causally insufficient, i.e. there can be confounders that are not measured in any of the available data sets. Each data set $\mathcal{D}_k$ is considered to be the result of an experiment $\mathcal{E}_k$ that partitions the variables $\mathcal{V}$ into $\mathcal{J}_k$ (intervened), $\mathcal{U}_k$ (observed), and $\mathcal{L}_k$ (hidden) variables. We assume that all variables in $\mathcal{V}$ are measured in at least one of the experiments, i.e. for any $x \in \mathcal{V}$ there exists some experiment $\mathcal{E}_k$ with $1 \leq k \leq K$ such that $x \in \mathcal{J}_k \cup \mathcal{U}_k$. Passive observational studies are trivially represented by 'null'-experiments where $\mathcal{J}_k = \emptyset$.

We assume that there is a single generating model over the variables in $\mathcal{V}$ that, suitably manipulated for each experiment, gives rise to all the data sets. We consider the model class of linear cyclic models with latent variables (linear non-recursive SEMs with correlated disturbances) (Bollen, 1989). The model can be described by the equation

$$\mathbf{x} := \mathbf{B}\mathbf{x} + \mathbf{e} \tag{1}$$

where $\mathbf{x}$ is a vector that denotes the variables in $\mathcal{V}$ while $\mathbf{e}$ represents the corresponding exogenous disturbances. The model is parameterized by a direct effects matrix $\mathbf{B}$ and a covariance matrix of the disturbances $\mathrm{cov}(\mathbf{e}) = \mathbf{\Sigma_e}$. Confounding due to variables *external* to the joint set of variables $\mathcal{V}$ is modeled implicitly by non-zero off-diagonal entries in $\mathbf{\Sigma_e}$ that correlate the disturbances. Figure 1 shows how the structure of such a model can be described by a mixed graph. Note that we assume throughout that the diagonal of $\mathbf{B}$ is zero; this is not a substantial assumption.[1]

We use the standard notion of "surgical" interventions (Pearl, 2000), where an intervention makes the intervened variable independent of its normal causes. Formally, the intervened model $(\tilde{\mathbf{B}}, \tilde{\mathbf{\Sigma}}_\mathbf{e})$ is the same as the unintervened model $(\mathbf{B}, \mathbf{\Sigma_e})$ except that each row in $\tilde{\mathbf{B}}$ corresponding to edges incident on some $x_i \in \mathcal{J}$ is set to zero (thereby breaking the edges into any such variables); similarly, for the ith row and column of $\tilde{\mathbf{\Sigma}}_\mathbf{e}$. An intervention vector $\mathbf{c}$ is added to the model Equation (1) that determines the values of (only) the intervened variables according to an intervention distribution with $\boldsymbol{\mu_c}$ and $\mathbf{\Sigma_c}$. For notational simplicity we assume that the intervened variables are randomized independently with zero mean and unit variance. This does not seriously limit the applicability of the methods.[2] For any individual (experimental) data set $\mathcal{D}_k$ we then assume that only the values of $\mathbf{x}$ corresponding to variables in $\mathcal{J}_k \cup \mathcal{U}_k$ are revealed (see Figure 1, right).

To ensure that the cyclic model has a well-defined equilibrium distribution, Hyttinen et al. (2012) examined models that are *weakly stable*, that is, models for which the $\mathbf{I} - \mathbf{B}$ matrix of the model (where $\mathbf{I}$ is the identity matrix) and all corresponding manipulated versions are invertible. Weak stability implies that the observed distribution follows a covariance matrix $\mathrm{cov}(\mathbf{x}) = (\mathbf{I}-\mathbf{B})^{-1}\mathbf{\Sigma_e}(\mathbf{I}-\mathbf{B})^{-T}$. However, in the

[1] There is inherent underdetermination of the diagonal elements of $\mathbf{B}$, related to self-loops (edges from a node to itself) in the model. These do not affect the equilibrium distribution in linear cyclic models, and can be handled by a suitable normalization (Hyttinen et al., 2012).

[2] A detailed formal account of the intervened model is given by Hyttinen et al. (2012), see their Equation 7 and Lemma 5.

case of overlapping data sets we have to make a slightly stronger assumption to obtain the desired identifiability results:

Assumption 1 (No unit cycles) *The sum-product of edge-coefficients on any subset of paths from a variable back to itself cannot sum up to exactly 1.*

Weak stability follows from this assumption. The proof of this claim, and all subsequent results in this paper, can be found in the Appendix in the Supplementary Material.

We can now state our learning problem formally: Given data sets $\mathcal{D}_1, \ldots, \mathcal{D}_K$ containing measurements of the marginalized distribution of a (experimentally manipulated) linear cyclic model with latent variables $(\mathbf{B}, \mathbf{\Sigma_e})$ that satisfies Assumption 1, identify as many causal relations among the joint set of variables, i.e. entries in $\mathbf{B}$, as possible.

3 IDENTIFIABILITY

One of the advantages of linear cyclic models is that the correlation measured between an intervened variable x_i and a non-intervened variable x_j corresponds to the sum-product of all the directed paths from x_i to x_j in a graph where all arcs into the intervened variables in $\mathcal{J}_k$ are cut. Eberhardt et al. (2010) refer to these terms as *experimental effects* and use the notation $\text{cov}_k(x_i, x_j) = t(x_i \rightsquigarrow x_j || \mathcal{J}_k)$. When the intervention set of an experimental effect contains all but one variable, then it corresponds to the *direct effect*: $t(x_i \rightsquigarrow x_j || \mathcal{V} \setminus \{x_j\}) = b(x_i \rightarrow x_j) = \mathbf{B}[j, i]$. Similarly, when only one variable is subject to intervention, we have the *total effect*: $t(x_i \rightsquigarrow x_j || \{x_i\}) = t(x_i \rightsquigarrow x_j)$.

Eberhardt et al. (2010) describe a learning algorithm which uses experimental effects estimated from the data to form linear equations on the unknown total effects. In particular, for an experiment with a variable $x_i \in \mathcal{J}_k$ and a variable $x_u \in \mathcal{U}_k$, one can obtain the constraint

$$t(x_i \rightsquigarrow x_u) = \sum_{x_j \in \mathcal{J}_k \setminus \{x_i\}} t(x_i \rightsquigarrow x_j) t(x_j \rightsquigarrow x_u || \mathcal{J}_k) + t(x_i \rightsquigarrow x_u || \mathcal{J}_k). \quad (2)$$

Note that for all $x_j \in \mathcal{J}_k$ the experimental effects $t(x_j \rightsquigarrow x_u || \mathcal{J}_k)$ are directly given by the relevant covariances (see above) in the data from this experiment, so the equation is linear in the unknown total effects. The intuition behind the equation is that the set of all directed paths from x_i to x_u in the original (unmanipulated) model can be separated into those going through each of the other intervened variables (the ones in $\mathcal{J}_k \setminus \{x_i\}$) and the remaining ones. Collecting a sufficient number of such linear constraints makes it possible to identify all total effects, from which it is straightforward to obtain the direct effects, as described in their paper.

While their learning procedure was constructed for the fully observed case, we note that it is directly applicable to our overlapping data sets case, since for any $x_i \in \mathcal{J}_k$ and any $x_u \in \mathcal{U}_k$, all required experimental effects are covariances between observed variables (and do not involve any variables in $\mathcal{L}_k$). Moreover, the identifiability results described in Eberhardt et al. (2010) and Hyttinen et al. (2012) can be generalized to the current setting once it is clarified how their conditions apply when the set of variables $\mathcal{V}$ is split into three sets $\mathcal{J}_k, \mathcal{U}_k$ and $\mathcal{L}_k$, rather than just $\mathcal{J}_k$ and $\mathcal{U}_k$. For this purpose, we say that a given set of experiments $\mathcal{E}_1, \ldots, \mathcal{E}_K$ satisfies the *pair condition* for the ordered pair of variables (x_j, x_u) if there is an experiment $\mathcal{E}_k$, with $1 \leq k \leq K$, such that $x_j \in \mathcal{J}_k$ and $x_u \in \mathcal{U}_k$. Note that an experiment with $x_j \in \mathcal{J}_k$ and $x_l \in \mathcal{L}_k$ does not satisfy the pair condition for the pair (x_j, x_l) even though x_l is not subject to intervention. The identifiability conditions for overlapping data sets are then as follows:

Theorem 1 (Sufficiency) *Given some set of experiments, a linear cyclic model with latent variables satisfying Assumption 1 is fully identified if the pair condition is satisfied for all ordered pairs* $(x_i, x_j) \in \mathcal{V} \times \mathcal{V},\ x_i \neq x_j$.

Note that unlike in Eberhardt et al. (2010), our sufficiency result here requires the slightly stronger Assumption 1. We further show that the sufficient condition is also in the worst case necessary.

Theorem 2 (Worst Case Necessity) *Given any set of experiments that does* not *satisfy the pair condition for all ordered pairs of variables* $(x_i, x_j) \in \mathcal{V} \times \mathcal{V},\ x_i \neq x_j$, *there exist two distinct linear cyclic models with latent variables satisfying Assumption 1 that are indistinguishable given those experiments.*

So with minor adjustments to the conditions and assumptions, we obtain for the overlapping data sets case very similar identifiability results as were given for the case when all data sets shared the same set of variables. In fact, as indicated in Appendix C, the learning algorithm of Eberhardt et al. (2010) is also *complete* given the background assumptions we have been considering here, in the sense that the information gained from the experimental effects fully exhausts what can be inferred about the underlying causal model. That is, correlations between non-intervened variables in this case provide no extra benefit. In Section 7 we use this algorithm ('EHS') as a baseline comparison of what a

complete procedure can infer without making stronger search space assumptions. Together, Theorems 1 and 2 provide *a priori* guidelines to identify which experimental results are needed to complement any available ones. Note that K specifically chosen experiments are enough to the satisfy the identifiability condition for models with up to $\binom{K}{\lfloor K/2 \rfloor}$ variables (Spencer, 1970).

4 FAITHFULNESS & BILINEARITY

In most realistic contexts the set of available overlapping data sets will not contain a sufficient variety of experimental interventions to satisfy the demanding identifiability condition given in the previous section. Hence, the full model will typically not be fully identified by the 'EHS' procedure. Furthermore, in most cases the overwhelming majority of the direct effects will remain underdetermined, as is evident from the simulations in Section 7.

Thus, a useful approach might be to strengthen the assumptions regarding the underlying model. The natural assumption, prevalent throughout causal discovery research, is the assumption of *faithfulness* (Spirtes et al., 1993), which can be seen as a simplicity or minimality assumption for causal discovery procedures. Most commonly used for inference in passive observational data, we consider the natural extension of the faithfulness assumption to experimental contexts: A data-generating model is said to be *faithful* if *all independences true in any (manipulated) observed distribution are consequences of the structure of the (manipulated) graph, and not due to the specific parameter values of the model*. In short, faithfulness enables us to infer from a statistical independence that there is an absence of a causal connection between the variables in question, in the particular context under which the statistical independence occurs. It is well known that heavy confounding often results in causal models that are effectively unfaithful given the limited sample size of the data. Nevertheless, if one has reason to believe that models are sparse and confounding limited, then faithfulness may constitute a reasonable assumption that can provide further insights.

Hyttinen et al. (2010) considered a set of simple faithfulness rules by which marginal or conditional independencies in the observed variables were used to derive constraints of the form $b(x_i \to x_j) = \mathbf{B}[j,i] = 0$, i.e. based on statistical independencies in the experimental data, some direct effects were inferred to equal zero. While these constraints are (trivially) linear in the direct effects, they are highly *non-linear* in the total effects, and thus cannot easily be integrated with linear constraints on the total effects of the form of Equation 2. To solve this problem Hyttinen et al. (2010) noted that one could, in fact, use the measured experimental effects to put linear constraints on the *direct effects*, as opposed to the total effects.

$$t(x_i \rightsquigarrow x_u || \mathcal{J}_k) = b(x_i \rightsquigarrow x_u) + \sum_{x_j \in \mathcal{V} \setminus (\mathcal{J} \cup \{x_u\})} t(x_i \rightsquigarrow x_j || \mathcal{J}_k) b(x_j \rightsquigarrow x_u) \quad (3)$$

In this way, the constraints from the experimental effects could be straightforwardly combined with faithfulness constraints, resulting in a fully linear system of equations constraining the direct effects (Hyttinen et al., 2010). The resulting linear system can be analyzed to determine which direct effects are identified and which are underdetermined; this procedure is empirically evaluated in Section 7 under the abbreviation 'HEH'.

Unfortunately, in the overlapping variables scenario, in most cases one cannot obtain many linear constraints on the direct effects. This is because one or more of the non-intervened variables may be latent, and one needs the experimental effects to *all* non-intervened variables in a given experiment to construct the linear constraints on the direct effects. In general, the experimental effects $t(x_i \rightsquigarrow x_j || \mathcal{J}_k)$ are not available when $x_j \in \mathcal{L}_k$, unless they can be inferred from the combination of results from all data sets. On the other hand, as we discussed in Section 3, we *can* obtain linear constraints on the *total* effects, because latents do not form a problem in this case. The remaining problem is thus that, in general, we may have a large set of linear constraints on the elements of the total effects matrix $\mathbf{T} = (\mathbf{I} - \mathbf{B})^{-1}$, where $\mathbf{T}[j,i] = t(x_i \rightsquigarrow x_j)$, combined with the knowledge that certain elements of $\mathbf{B}$ are zero, but finding a solution to this combined set of constraints is no longer trivial.

One approach to solving this problem is to take the *combination* of the direct effects (elements of $\mathbf{B}$) *and* the total effects (elements of $\mathbf{T}$) as the unknown free parameters of the problem. In this way, we now have a *bilinear* system of equations: There are equations linear in the elements of $\mathbf{B}$, equations linear in the elements of $\mathbf{T}$, and the constraint $\mathbf{T}(\mathbf{I} - \mathbf{B}) = \mathbf{I}$ yields equations that are bilinear; they are linear in one parameter vector given the other. To determine the underdetermination of the system under this combined set of constraints thus involves characterizing the solution set for such a bilinear equation system. Unfortunately, no efficient solution methods for large bilinear equation systems are known, except in certain special cases (Cohen and Tomasi, 1997; Johnson and Link, 2009).

Nevertheless, one can attempt to solve the system by

minimizing the objective function

$$C(\mathbf{B}, \mathbf{T}) = \|\mathbf{K}_1 \text{vec}(\mathbf{B}) - \mathbf{k}_1\|^2 + \|\mathbf{K}_2 \text{vec}(\mathbf{T}) - \mathbf{k}_2\|^2 + \|\mathbf{T}(\mathbf{I} - \mathbf{B}) - \mathbf{I}\|^2, \quad (4)$$

where the $\|\cdot\|^2$ denotes squared Euclidean norm (for a vector) and squared Frobenius norm (for a matrix), i.e. in both cases the sum of squares of the elements. In this objective function, the first term (involving $\mathbf{K}_1$ and $\mathbf{k}_1$) represents all linear constraints on the elements of $\mathbf{B}$ that we have been able to construct, the second term (involving $\mathbf{K}_2$ and $\mathbf{k}_2$) similarly represents any available linear constraints on the elements of $\mathbf{T}$, while the last term derives from the constraint $\mathbf{T} = (\mathbf{I} - \mathbf{B})^{-1}$. This objective function is quadratic in $\mathbf{B}$ (for fixed $\mathbf{T}$), and quadratic in $\mathbf{T}$ (for fixed $\mathbf{B}$), so it is easy to set up an alternating variables approach which minimizes with respect to each one at a time, keeping the other fixed. Since the objective is not convex with respect to the joint parameter vector, one cannot be guaranteed to find the global minimum using such a procedure. In our experience, however, this approach is generally quite effective in finding solutions to the underlying system of equations, when such solutions exist. We use this procedure, starting from a sample of random initial points, to generate a sample of solutions to the available constraints. For each of the direct effects (elements of $\mathbf{B}$) we then, on the basis of its variance in the sample, classify it as determined or underdetermined. For determined coefficients we further infer whether the coefficient is zero (edge is absent) or non-zero (edge is present), based on the mean value of the coefficient in the sample. Note that any constraints on the model parameters that follow the above bilinear form can easily be added to this inference procedure. In fact, it turns out that several of the faithfulness constraints derived in the following section can be added. With these additions, we empirically evaluate this method (under the abbreviation 'BILIN') in the simulations in Section 7.

5 FAITHFULNESS CONSTRAINTS ON SETS OF PATHS

While constraints of the form $b(x_i \to x_j) = 0$ are the most obvious consequences of the faithfulness assumption, they by no means exhaust the inferences that are possible in linear models. In this section, we provide a more general framework for deriving and representing faithfulness constraints, and in the following section we make use of these constraints in an inference algorithm for the overlapping data sets setting.

We build on the work of Claassen and Heskes (2011), who developed a logical inference procedure based on *minimal* conditioning sets to derive graphical consequences of the faithfulness assumption. A minimal conditioning set, indicated by the brackets, identifies exactly the set of variables that make a pair of variables (in)dependent. Formally, for variables x and y and disjoint sets of variables C and D not containing x and y, we denote a minimal independence by

$$x \perp\!\!\!\perp y \,|\, D \cup [C] \text{ whenever we have } x \perp\!\!\!\perp y \,|\, D \cup C \text{ and } \forall C' \subsetneq C, x \not\perp\!\!\!\perp y \,|\, D \cup C',$$

and a minimal dependence by

$$x \not\perp\!\!\!\perp y \,|\, D \cup [C] \text{ whenever we have } x \not\perp\!\!\!\perp y \,|\, D \cup C \text{ and } \forall C' \subsetneq C, x \perp\!\!\!\perp y \,|\, D \cup C'.$$

In both cases D and C can be empty, although when C is empty, the statements become trivial. Under the assumption that the generating model is acyclic, Claassen and Heskes (2011) provide causal inference rules based on such minimal dependencies and independencies, that identify all consequences of faithfulness given passive observational data on a set of (potentially) causally insufficient variables. We drop their acyclicity axiom and change and expand their rules to apply to overlapping data sets, experimental data, and cyclic generating models. We use them to derive constraints on the experimental effects of hypothetical experiments, which, in turn, are used to identify additional features of the underlying causal model.

For example, if we find that $x \perp\!\!\!\perp y \,|\, [z]$ in the manipulated distribution where only z is subject to an intervention, then we infer that $t(x \rightsquigarrow y || \{x, z\}) = 0$, i.e. for an experiment in which x and z are subject to intervention, there would be no correlation between x and y, since, by faithfulness, there is no directed path from x to y that does not pass through z. Moreover, we can generalize this constraint to all supersets of the intervention set:

$$t(x \rightsquigarrow y || \mathcal{J}) = 0 \quad \Rightarrow \quad t(x \rightsquigarrow y || \mathcal{J} \cup C) = 0 \quad (5)$$

for all $C \subseteq \mathcal{V} \setminus \{y\}$. Note, in particular, that C may contain variables in $\mathcal{L}$. As a special case of this rule we thus obtain the direct effect constraint $t(x \rightsquigarrow y || \mathcal{V} \setminus \{y\}) = b(x \to y) = 0$. Similarly, x and y can be reversed in all the previous constraints. If we additionally found for some non-intervened w, that $x \not\perp\!\!\!\perp y \,|\, \{z\} \cup [w]$, then w is a so-called "unshielded collider", which implies that $t(w \rightsquigarrow x || \{w, z\}) = t(w \rightsquigarrow y || \{w, z\}) = 0$. Again we can also generate constraints for all the experimental effects with supersets of the intervention sets.

Since faithfulness in general implies the absence of particular causal pathways given a set of discovered independence constraints, it implies for the linear models that we have been considering, that the sum-products

of edge coefficients representing a (set of) path(s) connecting two variables are zero. An independence between x and y in a passive observational data set can thus give rise to a set of polynomial equations representing the fact that there is no directed pathway from x to y or vice versa, and that there is no (latent) common cause of x and y. Quite apart from the large number of equations such a general approach produces, there are no known efficient solution methods for such a set of equations. We thus have to determine which subset of the consequences of faithfulness we can handle.

We found that for consequences of faithfulness that can be represented as bilinear equations on the experimental effects, such as Equation 6, computationally feasible solution methods could still be developed.

$$t(x \rightsquigarrow u || \mathcal{J} \cup \{x\}) \cdot t(u \rightsquigarrow y || \mathcal{J} \cup \{u\}) \quad = \quad 0 \quad (6)$$

The bilinear approach of Section 4 can include such equations as (6) if one of the experimental effects is a total effect and the other is a direct effect. In Section 6 we present a method that handles versions of Equation 6 in its more general form. In Appendix D we describe the full set of faithfulness rules we apply and the equations we use. These include the skeleton rules of the PC-algorithm, as well as many other standard faithfulness inferences. We also note the rules' limitations by giving an example of an inference that we cannot include in the current bilinear representation.

The independence constraints due to faithfulness thus allow us to obtain zero-constraints on a variety of experimental effects and their products without ever having to perform the corresponding experiment or having a data set that includes all the variables present in the constraint. We note that in practice inferences due to faithfulness are very sensitive to the correctness of the independence test judgments. In particular, once the inference depends on more than a simple unconditional independence test, reliability rapidly decreases and the theoretical gains one obtains in the infinite sample limit evaporate (see Section 7 for more discussion and comparisons). So while a complete method for inferences based on faithfulness given the background assumptions here is still of theoretical interest, its practical usefulness may be limited.

6 INFERENCE ALGORITHM

In Section 4 we took an approach that overparameterized the search problem by treating both the total and the direct effects as unknown. The proposed method searched for solutions using the available constraints that could be represented in the bilinear system relating total and direct effects. But in the previous section we showed that the assumption of faithfulness implies zero-constraints on the experimental effects that cannot, in general, be used by such a bilinear method (see Equations 5 & 6). In this section we propose a method that includes these more general constraints on experimental effects that are neither total nor direct effects. We consider *all* experimental effects as unknown parameters of the model. This can be seen as taking the overparametrization one step further, but given that there are now a total of $n(n-1)2^{n-2}$ different experimental effects for a model with n variables in $\mathcal{V}$, the iterative approach of Section 4 becomes unfeasible; something simpler is needed.

Algorithm 1 Linear Inference algorithm: LININF

Record all observed experimental effects.

Record any experimental effects directly identified as zero by the faithfulness rules.

While TRUE,

- Initialize matrix $\mathbf{A}$ and vector $\mathbf{b}$ as empty. Let $\mathbf{x}$ be the vector of the remaining unknown experimental effects.
- For all experimental effects $t(x_i \rightsquigarrow x_u || \mathcal{J}_k)$,
  - For all sets $\mathcal{J}_l$ such that $\mathcal{J}_k \subsetneq \mathcal{J}_l \subsetneq \mathcal{V} \setminus \{x_u\}$,
    - If Equation 7 is *linear* with respect to the unknown experimental effects, add it to the system $\mathbf{Ax} = \mathbf{b}$.
- For all equations induced by the faithfulness rules,
  - If the equation is *linear* with respect to the unknown experimental effects, add it to the system $\mathbf{Ax} = \mathbf{b}$.
- Solve $\mathbf{x}$ from $\mathbf{Ax} = \mathbf{b}$ and determine the uniquely identified elements of $\mathbf{x}$.
- Record all experimental effects corresponding to uniquely identified elements of $\mathbf{x}$, and derive implied experimental effects based on Equation 5.
- Exit if no new experimental effects were recorded.

Output the recorded experimental effects.

Some experimental effects are readily observed in the overlapping experiments. We use these observed experimental effects to determine some of the remaining unknown ones. Hyttinen et al. (2012) generalized Equation 2 to show how some experimental effects can be used to constrain others. If two experiments with intervention sets $\mathcal{J}_k$ and $\mathcal{J}_l$ satisfy $\mathcal{J}_k \subsetneq \mathcal{J}_l \subsetneq \mathcal{V} \setminus \{x_u\}$, then the experimental effects of the two experiments are related by the following equation:

$$t(x_i \rightsquigarrow x_u || \mathcal{J}_k) = t(x_i \rightsquigarrow x_u || \mathcal{J}_l) + \sum_{x_j \in \mathcal{J}_l \setminus \mathcal{J}_k} t(x_i \rightsquigarrow x_j || \mathcal{J}_k) t(x_j \rightsquigarrow x_u || \mathcal{J}_l) \quad (7)$$

Generally such equations are non-linear in the unknown experimental effects. However, the equation is *linear* if for all $x_j \in \mathcal{J}_l \setminus \mathcal{J}_k$ either $t(x_i \rightsquigarrow x_j || \mathcal{J}_k)$ or

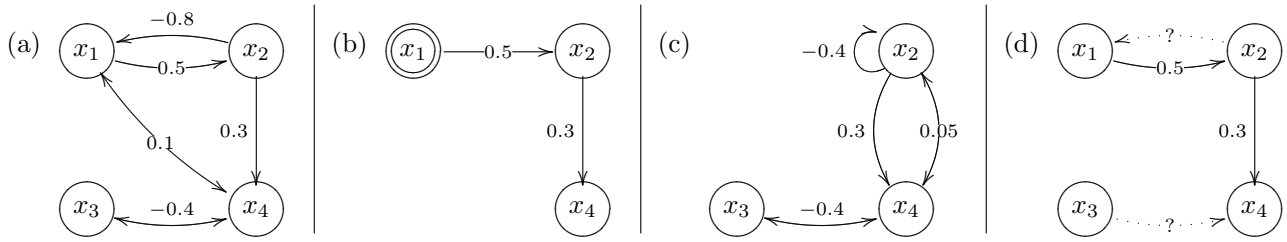


Figure 2: A case study. (a) True data generating model. (b) The manipulated and marginalized model corresponding to an experiment with $\mathcal{J}_1 = \{x_1\}$, $\mathcal{U}_1 = \{x_2, x_4\}$, and $\mathcal{L}_1 = \{x_3\}$. (c) The marginalized model corresponding to a 'null'-experiment with $\mathcal{J}_2 = \emptyset$, $\mathcal{U}_2 = \{x_2, x_3, x_4\}$, and $\mathcal{L}_2 = \{x_1\}$. (d) The causal structure over the union of observed variables learned by LININF (omitting any double-headed arcs). Notice that all causal arcs between x_1 and x_3 are discovered to be absent, although variables x_1 and x_3 are not observed together in either of the observed data sets. Dotted arrows indicate underdetermined parameters.

$t(x_j \rightsquigarrow x_u || \mathcal{J}_l)$ is already known. In addition, the full set of implications from the faithfulness rules of Section 5 (see Appendix D) may require that a particular experimental effect is zero, or – as in Equation 6 – that $t(x_i \rightsquigarrow x_j || \mathcal{J}_k) t(x_j \rightsquigarrow x_u || \mathcal{J}_l)$ is zero. Combining all the available information can render several equations linear. We ignore any equations that remain non-linear in the unknown experimental effects in order to keep the solvability of the system feasible.

There are a total of $n(n-1)\sum_{i=0}^{n-2}\binom{n-2}{i}(2^{n-i-2}-1)$ equations similar to Equation 7 for a model with n variables in $\mathcal{V}$. Thus, if even a small share of these equations are linear with respect to the unknown experimental effects, the model space compatible with the constraints at hand will be substantially restricted. Furthermore, equations that were initially non-linear in the unknown experimental effects, can be rendered linear as we iterate the inference and more experimental effects are determined. Since we are only using linear equations, we can trivially see which of the unknown experimental effects are uniquely determined by the system of equations. Similarly, since the direct effects are just a special case of the experimental effects, we can find which direct effects are underdetermined.

The Linear Inference algorithm using these ideas is outlined in Algorithm 1. A powerful part of the algorithm is the application of the inference in Equation 5 throughout the several rounds of inference. For finite samples this inference depends on judging whether an inferred experimental effect that is not measured in any of the available data sets, is zero. One option is to run the same algorithm in parallel on several resampled data sets, and see whether all estimates for a single experimental effect are effectively zero or not.

Unlike the bilinear procedure of Section 4, this linear inference procedure is consistent. It is incomplete as it does not exploit the available non-linear equations, but it is able to extract more information from the overlapping experiments by benefiting from the vast number of different faithfulness constraints produced by sparse data generating models.

7 SIMULATIONS

In the overlapping data set setting the general idea is that the experiments have already been run, with little regard to how the data sets and interventions may fit together. So it is likely that the identifiability condition of Section 3 is not satisfied and that there still is a large amount of underdetermination. In particular, one might expect substantial underdetermination with regard to the causal relations between variables that are never measured together in one data set. Figure 2 offers a concrete example of how the findings from multiple experimental data sets can be synthesized to yield knowledge not available from any of the individual data sets. Panel (a) shows the true underlying data generating model. This model is subject to one experiment where x_1 is intervened, x_2 and x_4 are observed, while x_3 is latent (b), and one 'null'-experiment where all variables except x_1 are passively observed (c). Note that x_1 and x_3 are never measured together in the same data set. One can think of the two experiments as resulting from two different research groups investigating two overlapping sets of variables. Panel (b) represents the manipulated and marginalized version of the true model corresponding to the first experiment, while (c) represents the marginalized model corresponding to the 'null'-experiment. Panel (d) shows the output of the Linear Inference algorithm described in Section 6 applied to the overlapping data sets of the two experiments. Here, solid edges denote identified edges, absences of edges denotes cases where the algorithm has inferred that there is no edge, and what remains undetermined is shown by dotted edges. Note that faithfulness has been effectively used to detect the absence of several edges in the underlying model. In

particular, x_4 is determined *not* to be a direct cause of any other variable, without there being any experiment intervening on x_4. Also, similarly to the results of Tillman et al. (2009) for passive observational data from acyclic models, we can detect the absence of edges between x_1 and x_3, *even though they were never measured together in any of the data sets.* Only the existence of the arcs $x_2 \to x_1$ and $x_3 \to x_4$ (indicated by dotted arrows) is left underdetermined. Finally, note that we have not discussed the detection of confounding here, so the absence of bi-directed edges in panel (d) is not indicative of any prediction on such edges.

We assessed the performance of our methods more generally in simulations. To test the effect of the faithfulness assumption we ran the method of Eberhardt et al. (2010) (EHS), that was shown in Section 3 to be complete for our circumstances, as a baseline for comparison. It does not assume faithfulness. We compared this baseline with the performance of the new BILINear and LINear-INFerence methods, presented in Sections 4 and 6, respectively, that were specifically designed for the overlapping data set case, and that assume faithfulness. Lastly, we ran the method of Hyttinen et al. (2010) (HEH), that only uses faithfulness constraints on the *direct effects* obtained *within* each data set.

The methods[3] were modified to produce one of three answers: 'present', ' absent', or 'unknown', for each possible edge, i.e. each entry of $\mathbf{B}$. An algorithm could only return 'present' if it had actually identified a specific value for the edge parameter. We compared how much of the true model the algorithms identified, and what percentage of that was correct.

To ensure that assuming faithfulness could actually make a difference, we simulated the methods on sparse models. We generated 100 6-variable models, with around 20% of the possible edges present, and 15% of the possible non-zero covariances, representing exogenous latent confounders. We drew the connection strengths randomly, though we avoided generating coefficients that were very close to zero, in order to make the predictions of absence/presence of edges meaningful. For each model, we constructed 5 overlapping experiments by selecting uniformly among partitions of the set of variables into intervened, observed and hidden variables, with the only constraint that for each experiment, neither $\mathcal{J}$ nor $\mathcal{U}$ were empty.

Figure 3 reports the percentage correct among the edges reported absent (left) and present (right) for

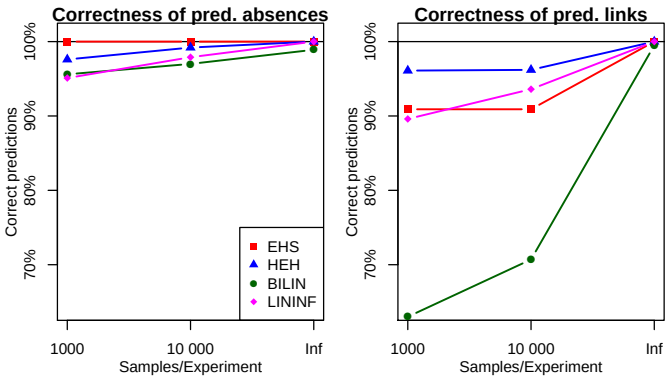


Figure 3: Accuracy of the learning algorithms. Each point on the lines is the average over 100 models.

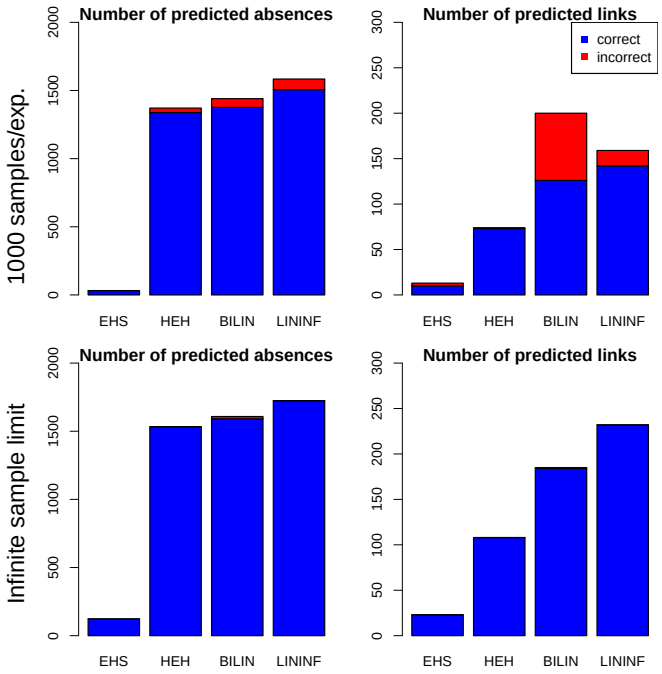


Figure 4: Predictions of the learning algorithms. Each bar represents the predictions over 100 models.

each algorithm at 1,000 and 10,000 samples per experiment, as well as for the infinite sample limit. Note that for the finite sample cases we restricted the maximum size of the conditioning set, in the search for faithfulness rules, to 1; this tends to improve the accuracy of the predictions. Similarly, in the finite sample cases, we restricted the number of inference rounds in the LININF algorithm to 1. With the exception of BILIN, the algorithms have a similar performance across the board. BILIN offers an undesirable trade-off: for finite samples it offers a high correctness of predicted absences, but a poor performance for predicted links. And since it is not consistent, it does not guarantee the absences of errors in the infinite sample limit, as

[3]Code implementing all compared methods and reproducing the simulations is available at `http://www.cs.helsinki.fi/u/ajhyttin/exp/`

can be seen in the presence of errors for the predicted absences.

Figure 4 shows the actual numbers of correct and incorrect predicted absences (left column) and links (right column) for each algorithm at 1,000 samples per experiment (top row) and the infinite sample limit (bottom row). In the 100 true models of the simulation there were a *total* of 624 edges and 2376 absences (confounding is not included in either of these counts). The first thing to notice is the scale of the plots: Substantial underdetermination remains for all algorithms, especially with respect to the identified links: More than half the links remain underdetermined. With respect to the individual algorithms we notice that EHS produces very few predictions, most edges are simply marked as 'unknown', since faithfulness is not assumed. HEH produces a remarkable number of predicted absences with good accuracy, so that the improvement of the methods tailored to the setting of overlapping experiments is limited for finite samples. However, the improvement in the number of predicted links is clearly quite significant (top and bottom right plots).

It would be desirable to have a comparison of how much underdetermination is inevitable given the overlapping data sets. A brute force check is not feasible even for six variable models, and we are not aware of any general results on equivalence classes for (manipulated) linear cyclic models with latent confounders. Developing an extension of LININF that is complete with regard to faithfulness may thus be the more promising route for such a comparison.

8 CONCLUSION

We have presented two new algorithms (BILIN and LININF) designed for causal learning from overlapping experimental or passive observational data sets. The first algorithm relies on a bilinear iterative optimization, while the second is based on solving linear constraints on general experimental effects. Both algorithms assume faithfulness, but apply to the general model class of linear cyclic models with latent confounding. We have also formulated necessary and sufficient conditions for model identifiability when faithfulness is not assumed.

The approach we have taken brings together several different strands of related work: We have connected the results on combining different *experimental* results on the *same* set of variables with the techniques of integrating *overlapping* data sets of *passive observational* data. To do so, we have relied on the inferences based on faithfulness developed for *non-parametric, acyclic* causal models with latent confounding, but adapted them to *linear, cyclic* models with latent confounding.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments and valuable suggestions. F.E. was supported by a grant from the James S. McDonnell Foundation on 'Experimental Planning and the Unification of Causal Knowledge'. A.H. and P.O.H. were supported by the Academy of Finland.

References

Bollen, K. A. (1989). *Structural Equations with Latent Variables*. John Wiley & Sons.

Claassen, T. and Heskes, T. (2011). A logical characterization of constraint-based causal discovery. In *UAI 2011*.

Cohen, S. and Tomasi, C. (1997). Systems of bilinear equations. Technical report, Stanford University.

Cooper, G. and Yoo, C. (1999). Causal discovery from a mixture of experimental and observational data. In *UAI 1999*.

Eaton, D. and Murphy, K. (2007). Exact Bayesian structure learning from uncertain interventions. In *AISTATS 2007*.

Eberhardt, F., Hoyer, P. O., and Scheines, R. (2010). Combining experiments to discover linear cyclic models with latent variables. In *AISTATS 2010*.

He, Y. and Geng, Z. (2008). Active learning of causal networks with intervention experiments and optimal designs. *Journal of Machine Learning Research*, 9:2523–2547.

Hyttinen, A., Eberhardt, F., and Hoyer, P. O. (2010). Causal discovery for linear cyclic models. In *PGM 2010*.

Hyttinen, A., Eberhardt, F., and Hoyer, P. O. (2011). Noisy-or models with latent confounding. In *UAI 2011*.

Hyttinen, A., Eberhardt, F., and Hoyer, P. O. (2012). Learning linear cyclic causal models with latent variables. Submitted. Available online from the authors' homepages.

Johnson, C. R. and Link, J. A. (2009). Solution theory for complete bilinear systems of equations. *Numerical Linear Algebra with Applications*, 16(11-12):929–934.

Murphy, K. P. (2001). Active learning of causal Bayes net structure. Technical report, U.C. Berkeley.

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Schmidt, M. and Murphy, K. (2009). Modeling discrete interventional data using directed cyclic graphical models. In *UAI 2009*.

Spencer, J. (1970). Minimal completely separating systems. *Journal of Combinatorial Theory*, 8(4):446 – 447.

Spirtes, P., Glymour, C., and Scheines, R. (1993). *Causation, Prediction, and Search*. Springer-Verlag.

Tillman, R. E., Danks, D., and Glymour, C. (2009). Integrating locally learned causal structures with overlapping variables. In *NIPS 2008*.

Tillman, R. E. and Spirtes, P. (2011). Learning equivalence classes of acyclic models with latent and selection variables from multiple datasets with overlapping variables. In *AISTATS 2011*.

Tong, S. and Koller, D. (2001). Active learning for structure in Bayesian networks. In *UAI 2001*.

Triantafillou, S., Tsamardinos, I., and Tollis, I. G. (2010). Learning causal structure from overlapping variable sets. In *AISTATS 2010*.

Join-graph based cost-shifting schemes

Alexander Ihler, Natalia Flerova, Rina Dechter and Lars Otten
University of California Irvine
[ihler, nflerova, dechter, lotten]@ics.uci.edu

Abstract

We develop several algorithms taking advantage of two common approaches for bounding MPE queries in graphical models: mini-bucket elimination and message-passing updates for linear programming relaxations. Both methods are quite similar, and offer useful perspectives for the other; our hybrid approaches attempt to balance the advantages of each. We demonstrate the power of our hybrid algorithms through extensive empirical evaluation. Most notably, a Branch and Bound search guided by the heuristic function calculated by one of our new algorithms has recently won first place in the PASCAL2 inference challenge.

1 INTRODUCTION

Combinatorial optimization tasks such as finding the most likely variable assignment of a probability model, the *most probable explanation* (MPE) or maximum *a posteriori* (MAP) problem, arise in many applications. These problems are typically NP-hard; graphical models provide a popular framework for reasoning about such tasks and organizing computations (Pearl, 1988).

Mini-Bucket Elimination (MBE) (Dechter & Rish, 2003) is a popular bounding scheme that generates upper and lower bounds by applying the exact Bucket Elimination (BE) algorithm (Dechter, 1999) to a simplified problem obtained by duplicating variables. The relaxation view of MBE is closely related to a family of iterative approximation techniques based on linear programming (LP). These include "reweighted" max-product (Wainwright *et al.*, 2005), max-product linear programming (MPLP) (Globerson & Jaakkola, 2007), dual decomposition (Komodakis *et al.*, 2007), and soft arc consistency (Schiex, 2000; Bistarelli *et al.*, 2000). These algorithms simplify the graphical model into independent components and tighten the resulting bound via iterative cost-shifting updates. They can be thought of as "re-parameterizing" the original functions without changing the global model. Most of the schemes operate on the original factors of the model, although some works tighten the approximations by introducing larger clusters (Sontag *et al.*, 2008).

In this paper we combine these ideas to define two new hybrid schemes. One algorithm, called mini-buckets with max-marginal matching (MBE-MM), is a non-iterative algorithm that applies a single pass of cost-shifting during the mini-bucket construction. The second approach, Join Graph Linear Programming (JGLP) applies cost-shifting updates to the full mini-bucket join-graph, iteratively. Our empirical evaluation demonstrates the increased power of these hybrid approximations over their individual components.

Finally, one of the primary uses of bounding schemes is in generating heuristics for Best-First and Branch and Bound search (Marinescu & Dechter, 2009a,b). Our hybrid schemes drastically decrease the search space explored by Branch and Bound to significantly increase its power, evidenced by both our empirical evaluation and the results of the current Probabilistic Inference Challenge[1], where our algorithm won first place in all optimization categories.

From an LP perspective, JGLP can be viewed as a specific algorithm within the class of generalized EMPLP (Sontag & Jaakkola, 2009). Its main novelty is in showing how the systematic join-graph structures of mini-bucket can facilitate effective clique choices. JGLP works "top down", creating large clusters immediately, compared to other methods (e.g., Sontag & Jaakkola, 2009; Batra *et al.*, 2011) that work "bottom up" (gradually including triplets, etc.) and may be less effective for large-width problems. Our non-iterative MBE-MM algorithm is an MBE scheme with some cost-shifting done locally in each bucket. From

[1] http://www.cs.huji.ac.il/project/PASCAL/

this perspective it is closely related to (Rollon & Larrosa, 2006); the two methods differ primarily in the form of the cost-shifting update within each bucket. However, our update is motivated by its connection to a globally applicable tightening algorithm.

2 PRELIMINARIES

We consider combinatorial optimization problems expressed as graphical models, including Markov and Bayesian networks (Pearl, 1988) and constraint networks (Dechter, 2003). A *graphical model* is a tuple $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F}, \bigotimes)$, where $\mathbf{X} = \{X_i : i \in V\}$ is a set of variables indexed by set V and $\mathbf{D} = \{D_i : i \in V\}$ is the set of their finite domains of values. $\mathbf{F} = \{f_\alpha : \alpha \in F\}$ is a set of discrete functions, where we use $\alpha \subseteq V$ and $X_\alpha \subseteq \mathbf{X}$ to indicate the scope of function f_α, i.e., $X_\alpha = \text{var}(f_\alpha) = \{X_i : i \in \alpha\}$. The set of function scopes implies a *primal graph* whose vertices are the variables and which includes an edge connecting any two variables that appear in the scope of the same function. The combination operator $\bigotimes \in \{\prod, \sum, \bowtie\}$ defines the complete function represented by the graphical model $\mathcal{M}$ as $C(\mathbf{X}) = \bigotimes_{\alpha \in F} f_\alpha(X_\alpha)$. In this work, we focus on *max-sum problems*, in which we would like to compute the optimal value C^* and/or its optimizing configuration x^*:

$$C^* = C(x^*) = \max_{\mathbf{X}} \sum_{\alpha \in F} f_\alpha(X_\alpha) \tag{1}$$

2.1 MINI-BUCKET ELIMINATION

Bucket elimination (BE) (Dechter, 1999) is a popular algorithm for solving reasoning tasks over graphical models. It is a special case of cluster tree elimination in which the tree-structure upon which messages are passed is determined by the variable elimination order used. In BE terminology, the nodes of the tree-structure are referred to as buckets and each bucket is associated with a variable to be eliminated.

Each bucket is processed by BE in two steps. First, all functions in the bucket are combined (by summation in the case of max-sum problem). Then the variable associated with the bucket is eliminated from the combined function (by maximization in case of max-sum task). The function resulting from the combination and elimination steps can be viewed as a "message" λ_i and is passed to the parent of the current bucket. Processing occurs in this fashion, from the leaves of the tree to the root, one node (bucket) at a time. The time and space complexity of BE are exponential in the graph parameter called the induced width w along the ordering o (Dechter, 1999).

Mini-bucket elimination (MBE) (Dechter & Rish, 2003) is an approximation scheme designed to avoid the space and time complexity of BE. Consider a bucket $\mathbf{B}_i$ and an integer bounding parameter z. MBE creates a z-partition $Q_i = \{q_i^1, \ldots, q_i^p\}$ of $\mathbf{B}_i$, where each set of functions $q_i^j \in Q_i$, called a *mini-bucket*, includes no more than $z + 1$ variables. Then each mini-bucket is processed separately, just as in BE. The scheme generates an upper bound on the exact optimal solution and its time and space complexity of MBE is exponential in z, which is chosen to be less than w, when w is too large. In general, increasing z tightens the upper bound, until $z = w$, in which case MBE finds the exact solution. The form of mini-bucket makes it easy to estimate and control both the computational and storage complexity through a single parameter z.

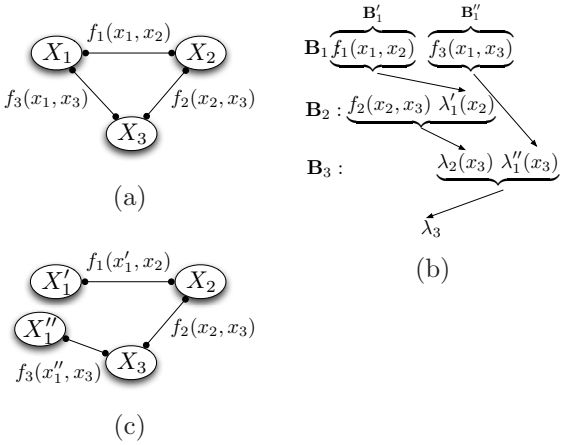


Figure 1: The mini-bucket procedure for a simple graph. (a) Original graph; (b) the buckets and messages computed in MBE; (c) interpreting MBE as variable duplication, and message passing on the resulting junction tree. X_1 is duplicated in each of two mini-buckets $q_1^1 = \{f_1(X_1, X_2)\}$ and $q_1^2 = \{f_3(X_1, X_3)\}$.

MBE's value is not only as a stand-alone bounding scheme, but also as a mechanism for generating powerful heuristic evaluation functions for informed search algorithms. Exploration of its potential as a heuristic generation scheme has yielded some of the most powerful Best-First and Branch and Bound search engines for graphical models, as summarized in (Kask & Dechter, 2001; Marinescu & Dechter, 2009a,b).

Mini-bucket elimination can also be interpreted using a junction tree view. The MBE bound corresponds to a problem relaxation in which a copy X_i^j of variable X_i is made for each mini-bucket $q_i^j \in Q_i$, and the resulting messages λ_i^j correspond to the messages in a junction tree defined on the augmented model over the variable copies; this junction tree is guaranteed to have induced-width z or less. Figure 1 shows a simple example on three variables. The problems are equivalent if all copies of X_i are constrained to be equal;

Algorithm 1 Join-Graph Structuring(z)

Input: Graphical model $\langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \sum \rangle$, parameter z
Output: Join-graph with cluster size $\leq z+1$

1: Order the variables from X_1 to X_n minimizing (heuristically) induced-width
2: Generate an ordered partition of functions $\mathbf{F} = \{f_\alpha\}$ into buckets $\mathbf{B}_1, \ldots, \mathbf{B}_n$, where $\mathbf{B}_i$ is a bucket of variable X_i
3: **for** $i \leftarrow n$ down to 1 (Processing bucket $\mathbf{B}_i$) **do**
4: Partition functions in $\mathbf{B}_i$ into $Q_i = \{q_i^1, \ldots, q_i^p\}$; each q_i^k has no more than $z+1$ variables.
5: For each mini-bucket q_i^k create a new set of variables $S_i^k = \{X | X \in q_i^k\} - \{X_i\}$ and place it in the bucket of its highest variable in the ordering
6: Maintain an arc between q_i^k and the mini-bucket (created later) that includes S_i^k
7: **end for**
8: Associate each resulting mini-bucket with a node in the join-graph
9: **Creating arcs:** keep the arcs created in step 6 and also connect the mini-bucket clusters belonging to the same bucket (for example, in a chain).

otherwise, the additional degrees of freedom lead to a relaxed problem and thus an upper bound.

The mini-bucket tree has several desirable properties (Mateescu *et al.*, 2010), including that it is a join-graph (satisfies the running intersection property) and each cluster has at most $z+1$ variables. The join-graph construction process is given in Algorithm 1.

2.2 LINEAR PROGRAMMING METHODS

Recently various iterative re-parameterization approaches have been introduced that are derived by solving an Linear Programming (LP) relaxation of the graphical model. Wainwright et al. (Wainwright *et al.*, 2005) established the connections between LP relaxations of integer programming problems and (approximate) dynamic programming methods using message-passing in the max-product algebra; subsequent improvements in algorithms such as max-product linear programming (MPLP) include coordinate-descent updates that ensure convergence (Globerson & Jaakkola, 2007; Sontag & Jaakkola, 2009). These methods are closely related to the mini-bucket bounds; since we build upon these approaches we introduce some of the ideas in more detail here.

To match the bulk of the graphical model LP relaxation literature, for this section we assume that the network consists of only pairwise functions $f_{ij}(X_i, X_j)$. A simple bound on the max-sum objective is then given by maxima of the individual functions:

$$C^* = \max_{\mathbf{X}} \sum_{(ij)\in F} f_{ij}(X_i, X_j) \leq \sum_{(ij)\in F} \max_{\mathbf{X}} f_{ij}(X_i, X_j), \tag{2}$$

exchanging the sum and max operators. One can interpret this operation as making an individual copy of each variable for each function, and optimizing over them separately.

However, we can also introduce a collection of functions $\{\lambda_{ij}(X_i), \lambda_{ji}(X_j)\}$ for each edge (ij), and require

$$\lambda \in \Lambda \quad \Leftrightarrow \quad \forall i, \quad \sum_j \lambda_{ij}(X_i) = 0$$

Then, we have

$$\begin{aligned} C^* &= \max_{\mathbf{X}} \sum_{(ij)\in F} f_{ij}(X_i, X_j) \\ &= \max_{\mathbf{X}} \sum_{(ij)\in F} f_{ij}(X_i, X_j) + \sum_i \sum_j \lambda_{ij}(X_i) \\ &\leq \min_{\lambda\in\Lambda} \sum_{(ij)\in F} \max_{\mathbf{X}} \left(f_{ij}(X_i, X_j) + \lambda_{ij}(X_i) + \lambda_{ji}(X_j) \right) \end{aligned} \tag{3}$$

by distributing each λ_{ij} to its associated factor and applying the inequality (2).

The new functions $\tilde{f}_{ij} = f_{ij}(X_i, X_j) + \lambda_{ij}(X_i) + \lambda_{ji}(X_j)$ define a *re-parameterization* of the original distribution, i.e., they change the individual functions without modifying the complete function $C(\mathbf{X})$. Depending on the literature, the λ_{ij} are interpreted as "cost-shifting" operations that transfer cost from one function to another while preserving the overall cost function (Schiex, 2000; Rollon & Larrosa, 2006), or as Lagrange multipliers enforcing consistency among the copies of X_i (Yedidia *et al.*, 2004; Wainwright *et al.*, 2005). In the former interpretation, the updates are called "soft arc-consistency" due to their similarity to arc-consistency for constraint satisfaction (Cooper & Schiex, 2004). Under the latter view, the bound corresponds to a *dual decomposition* solver for a linear programming (LP) relaxation of the original problem (Komodakis & Paragios, 2008; Sontag *et al.*, 2010).

The main distinguishing feature among such dual decomposition approaches is the way in which the bound is tightened by updating[2] the functions λ, generally either sub-gradient or gradient approaches (Komodakis & Paragios, 2008; Jojic *et al.*, 2010) or coordinate descent updates that can be interpreted as "message passing" (Globerson & Jaakkola, 2007; Sontag & Jaakkola, 2009). In practice, coordinate descent updates tend to tighten the bound more quickly, but can become caught in sub-optimal minima caused by

[2]We refer to these iterative bound improvement updates as "LP-tightening" updates, although technically we are tightening the decomposition bound (3) which is the dual of the LP. This is in contrast to literature that uses "tightening" to mean the inclusion of additional constraints (higher-order consistency), e.g., (Sontag *et al.*, 2008).

Algorithm 2 LP-tightening

Input: Graphical model $\langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \sum \rangle$, where f_α is a potential defined on variables X_α.
Output: Upper bound on the optimum value

1: Iterate until convergence:
2: **for** any pair α, β with $X_{\alpha\beta} = X_\alpha \cap X_\beta \neq \emptyset$ **do**
3: Compute max-marginals:
$\gamma_\alpha(X_{\alpha\beta}) = \max_{X_\alpha \setminus X_{\alpha\beta}} f_\alpha(X_\alpha)$
$\gamma_\beta(X_{\alpha\beta}) = \max_{X_\beta \setminus X_{\alpha\beta}} f_\beta(X_\beta)$
Update parameterization:
$f_\alpha(X_\alpha) \leftarrow f_\alpha(X_\alpha) + \frac{1}{2}\big(\gamma_\beta(X_{\alpha\beta}) - \gamma_\alpha(X_{\alpha\beta})\big)$
$f_\beta(X_\beta) \leftarrow f_\beta(X_\beta) + \frac{1}{2}\big(\gamma_\alpha(X_{\alpha\beta}) - \gamma_\beta(X_{\alpha\beta})\big)$
4: **end for**

the limited number of coordinate directions considered. Here we give an extremely simple update, most closely related to the "tree-block" coordinate descent updates derived in (Sontag & Jaakkola, 2009).

The algorithm is initialized with all $\lambda_{ij}(X_i) = 0$. Let us re-arrange the terms on right hand side of inequality 3, grouping together all functions that include a particular variable X_i in their scope. Consider minimizing over a single pair $\lambda_{ij}(X_i), \lambda_{ik}(X_i)$; since all other terms are fixed, we minimize

$$\begin{aligned}
&\big[\max_{\mathbf{X}} f_{ij} + \lambda_{ij}\big] + \big[\max_{\mathbf{X}} f_{ik} + \lambda_{ik}\big] \\
&= \max_{X_i} \big[\gamma_{ij}(X_i) + \lambda_{ij}(X_i)\big] + \max_{X_i} \big[\gamma_{ik}(X_i) + \lambda_{ik}(X_i)\big] \\
&\geq \max_{X_i} \big[\gamma_{ij}(X_i) + \lambda_{ij}(X_i) + \gamma_{ik}(X_i) + \lambda_{ik}(X_i)\big]
\end{aligned}$$

where we have defined the "max-marginals" $\gamma_{ij}(X_i) = \max_{X_j} f_{ij}(X_i, X_j)$, and we require that $\lambda_{ij}(X_i) + \lambda_{ik}(X_i) = 0$ to preserve $\lambda \in \Lambda$. Many choices of λ_{ij} achieve this minimum; a useful one is

$$\lambda_{ij} = \frac{1}{2}\big(\gamma_{ik}(X_i) - \gamma_{ij}(X_i)\big)$$

We can then set $f_{ij} \leftarrow f_{ij} + \lambda_{ij}$ and $\lambda_{ij} = 0$ (preserving $\lambda_{ij}(X_i) = 0$ for all i,j) and repeat, giving a simple coordinate descent update that can be interpreted as a max-marginal or "moment"-matching procedure on the functions f_{ij}. The update is easy to extend to higher-order functions $f_\alpha(X_\alpha)$; see Algorithm 2.

Another useful intuition comes from the alternative choice, $\lambda_{ij} = \gamma_{ik}(X_i) = \max_{X_k} f_{ik}(X_i, X_k)$. In this case, the contribution of edge (ik)'s term to the bound becomes zero. The "message" λ_{ij} is equivalent to the dynamic program computed on the chain j–i–k. It is thus easy to see that dynamic programming (or variable elimination) on a tree can also be interpreted as coordinate descent on the same bound, using the same set of coordinates, and that this procedure will exactly optimize any tree-structured graph (Johnson *et al.*, 2007; Yarkony *et al.*, 2010).

Algorithm 3 Factor graph LP (FGLP)

Input: Graphical model $\langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \sum \rangle$, where f_α is a potential defined on variables X_α.
Output: Upper bound on the optimum value

1: Iterate until convergence:
2: **for** each variable X_i **do**
3: Get factors $F_i = \{\alpha : i \in \alpha\}$ with X_i in their scope
4: $\forall \alpha$, compute max-marginals:
$\gamma_\alpha(X_i) = \max_{X_\alpha \setminus X_i} f_\alpha(X_\alpha)$
5: $\forall \alpha$, update parameterization:
$f_\alpha(X_\alpha) \leftarrow f_\alpha(X_\alpha) - \gamma_\alpha(X_i) + \frac{1}{|F_i|} \sum_{\beta \in F_i} \gamma_\beta(X_i)$
6: **end for**

A well-known LP-tightening algorithm is message-passing linear programming (MPLP) (Globerson & Jaakkola, 2007); its generalized version for higher-order cliques tightens the same bound as Algorithm 2 under descent updates along the same coordinates. The main distinction is that MPLP is formulated as sending messages between variable "beliefs" and the f_{ij}. Message-passing has the advantage that scheduled updates (e.g., Elidan *et al.*, 2006; Sutton & McCallum, 2007) are easily applied. However, for cliques with large separator sets these messages can consume a noticable fraction of available memory. In contrast, reparameterization updates like Algorithm 2 never store the λ, giving a memory advantage for large cliques. Although some scheduling approaches, such as those based on the decoding gap (Tarlow *et al.*, 2011), are still applicable to pure reparameterization updates, our implementation used a fixed update ordering.

The original MPLP, soft-arc consistency, and many other LP algorithms operate directly on the original functions f_α, updating coordinates that consist of single variable functions $\lambda(X_i)$ at each step; this corresponds to message passing on the "factor graph" representation of the model. Algorithm 3 (FGLP) gives a simple extension of the matching update that operates on this factor graph LP and tightens all f_{ij} involved with some X_i simultaneously. FGLP is similar to the "star-shaped" tree block coordinate descent derived in (Sontag & Jaakkola, 2009), but centered on separators (variables) rather than cliques (factors); in practice on our problems we found it faster than other update methods for this LP.

3 JOIN GRAPH LINEAR PROGRAMMING

From the perspective of Section 2.2, mini-bucket can be viewed within the LP-tightening framework. The mini-bucket procedure defines a join graph represented as a collection of maximal cliques and separators. The mini-bucket computations define a downward sweep of dynamic programming on the join graph with du-

plicate variables; it is equivalent to running an LP-tightening procedure to convergence, *only* messages along edges of the mini-bucket spanning tree.

Given this view, it is straightforward to consider iterative updates on the same set of maximal cliques. We can construct a join graph using the mini-bucket relaxation, again assigning the original functions f to their earliest clique and defining for any mini-bucket q_i^k a function associated with its corresponding clique,

$$F_{q_i^k}(X_{q_i^k}) = \sum_{f_\alpha \in q_i^k} f_\alpha(X_\alpha) \tag{4}$$

We then perform re-parameterization updates to the functions F_q as in Algorithm 2. The resulting join-graph linear programming (JGLP) algorithm is shown in Algorithm 4.

Note that once JGLP converges, re-processing these new functions using mini-bucket elimination using the same value of z will not change the choice of cliques (since no two cliques q, q' could be joined without violating the clique size bound, $|X_q| \leq z+1$). Additionally, the MBE pass will not change the bound value, since the MBE dynamic program can be viewed as a sequence of coordinate-descent updates along a subset of the edges, all of which must already be tight if the algorithm has converged. In the sequel, we will use this property of JGLP when developing heuristic evaluation functions for search; see Section 5.

MBE vs. LP perspectives. From the LP perspective, the MBE process can be thought of as a heuristic for selecting the cliques used to define the LP bound. This heuristic has the advantage of being very fast and controlled by a single integer value that is easy to search over, and it is easy to estimate the memory requirements of the approximation. In practice this results in a "top-down" construction, in which z is set to the induced width and reduced until the computational resource constraints are met. Existing heuristics for selecting variable orderings with low induced width (Robertson & Seymour, 1983; Kask *et al.*, 2011; Bodlaender, 2007; Gogate & Dechter, 2004) can be easily applied as well. In contrast, most existing generalized LP solvers work in a "bottom-up" fashion, running the LP to convergence on the original graph, then proposing slightly larger cliques (for example, from among fully connected triplets of variables (Sontag *et al.*, 2008)) based on some greedy heuristic. However, for problems with small variables and reasonable resource constraints, it is easy to represent very large cliques ($z = 15$ to 25), in which case the top-down approach can be far more effective.

From the MBE perspective, the LP tightening updates provide an iterative version of mini-bucket that attempts to compensate for the variable copying relaxation. From this view it is most closely related to non-iterative cost-shifting procedures during the mini-bucket construction phase (Rollon & Larrosa, 2006). Another iterative "relax and compensate" approach was recently proposed in (Choi & Darwiche, 2009), but did not explicitly maintain a re-parameterization or use coordinate descent (thus could fail to converge).

Algorithm 4 Algorithm JGLP

Input: Graphical model $\langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \sum \rangle$; variable order $o = \{X_1, \ldots, X_n\}$; parameter z.
Output: Upper bound on the optimum value

```
Place each function f_α in its latest bucket in o
Build mini-bucket join graph (Algorithm 1)
Find the function of each mini-bucket as in (4)
Iterate to convergence / time-limit:
for all pairs q^i, q^j connected by an edge do
    Find common variables, S = var(q^i) ∩ var(q^j)
    Find the max-marginals of each mini-bucket
    q^k: γ_k = max_{var(q^k)\S}(F_k), k = i, j
    Update functions in both mini-buckets
    q^i: F_i ← F_i − ½(γ_i − γ_j)
    q^j: F_j ← F_j + ½(γ_i − γ_j)
end for
Return: Ĉ* = Σ_i max_{X_i} F_i(X_i)
```

4 MBE-MM

While the iterative nature of JGLP is appealing, it can have significant additional time and space overhead compared to MBE. We would thus also like to consider a single-pass MBE-like algorithm in which LP tightening is performed only within each bucket. We call the resulting algorithm *mini-bucket with max-marginal matching*, or MBE-MM.

MBE-MM proceeds by following the standard mini-bucket downward pass. However, when each mini-bucket $q_i^j \in Q_i$ is processed, eliminating variable X_i, we first perform an LP-tightening update to the mini-bucket functions f_q. For storage and computational efficiency reasons, we perform a single update on all buckets simultaneously, matching their max-marginals on their joint intersection. See Algorithm 5.

Viewing the matching update as a cost-shifting procedure, our MBE-MM algorithm is closely related to the work of (Rollon & Larrosa, 2006), who also proposed several dynamic cost-shifting updates for mini-bucket, with different update patterns. Their updates correspond to a "dynamic programming"-like heuristic that shifts all cost into a single function. In our case, we mimic the "balanced" cost shifting used by optimal iterative tightening but restrict our updates to a single bucket rather than to the whole global problem.

It is worth noting that, although any max-marginal

Algorithm 5 Algorithm MBE-MM

Input: Graphical model $\langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \sum \rangle$; variable order $o = \{X_1, \ldots, X_n\}$; parameter z.
Output: Upper bound on the optimum value

1: Place each function f_α in its latest bucket in o
2: **for** $i \leftarrow n$ down to 1 (processing bucket $\mathbf{B}_i$) **do**
3: Partition functions in $\mathbf{B}_i$ into $Q_i = \{q_i^1, \ldots, q_i^p\}$, where each q_i^k has no more than $z+1$ variables.
4: Find the set of variables common to all the mini-buckets: $S_i = S_i^1 \cap \cdots \cap S_i^p$, where $S_i^k = var(q_i^k)$
5: Find the function of each mini-bucket q_i^k: $F_{ik} \leftarrow \prod_{f \in q_i^k} f$
6: Find the max-marginals of each mini-bucket q_i^k: $\gamma_{ik} = \max_{var(q_i^k) \setminus S_i}(F_{ik})$
7: Update functions of each mini-bucket q_i^k: $F_{ik} \leftarrow F_{ik} - \gamma_{ik} + \frac{1}{p}\sum_l \gamma_{il}$
8: Generate messages $\lambda_i^k = \max_{X_i} F_{ik}$ and place each in the latest variable in $var(q_i^k)$'s bucket.
9: **end for**
10: **Return:** The buckets and cost bound from $\mathbf{B}_1$

matching step strictly tightens the fully decomposed bound (3) within each bucket, it does not necessarily tighten the overall solution found by MBE (which corresponds to fully optimizing the MBE subtree's edges); thus MBE-MM is not guaranteed to be tighter than ordinary MBE. However, it is reasonable to expect that the update will help, and in practice we find that the bounds are almost always significantly improved (see the experiments, Section 6).

Like "soft" arc-consistency, our algorithms are also related to known methods in constraint satisfaction. MBE(z) and JGLP(z), parameterized by a z-bound, are analogous to directional z-consistency and full z-consistency, respectively. Our algorithm MBE-MM represents an intermediate step between these two, and is analogous to an improvement of directional i-consistency with full iterative relational consistency schemes within each bucket (Dechter, 2003).

5 HEURISTICS FOR SEARCH

Mini-bucket is also a powerful mechanism for generating heuristics for informed search algorithms (Kask & Dechter, 2001). The intermediate functions λ recorded by MBE (see Figure 1(b)) are used to express upper bounds on the best extension of any partial assignment, and so can be used as admissible heuristics guiding Best-First or Branch and Bound search.

Algorithms JGLP and FGLP do not produce heuristic functions directly; we obtain one by applying MBE to the modified (re-parameterized) functions output by the iterative algorithms. As noted in Section 3, for JGLP constructed with the same clique size bound z, this additional pass does not change the value of the bound. In contrast, applying MBE to the (much smaller) functions re-parameterized by FGLP forms new clusters and typically tightens the bound, yielding a hybrid heuristic generator "FGLP+MBE". Being a straightforward extension of "ordinary" MBE, the MBE-MM algorithm also directly yields a heuristic function suitable for informed search.

In our experiments (Section 6.2) we apply the output bounds as admissible heuristics for one of the most effective informed search approaches: the AND/OR Branch and Bound (AOBB) algorithm (Marinescu & Dechter, 2009a). This algorithm explores in a depth-first manner an AND/OR search space that is defined using a pseudo-tree arrangement of the problem's primal graph and takes advantage of the problem decomposition. The AND/OR search space is usually much smaller than the corresponding standard OR search space. AOBB keeps track of the value of the best solution found so far (a lower bound on the optimal cost) and uses this value and the heuristic function to prune away portions of the search space that are guaranteed not to contain the optimal solution in a typical branch-and-bound manner. For details on AOBB guided by MBE see (Marinescu & Dechter, 2009a).

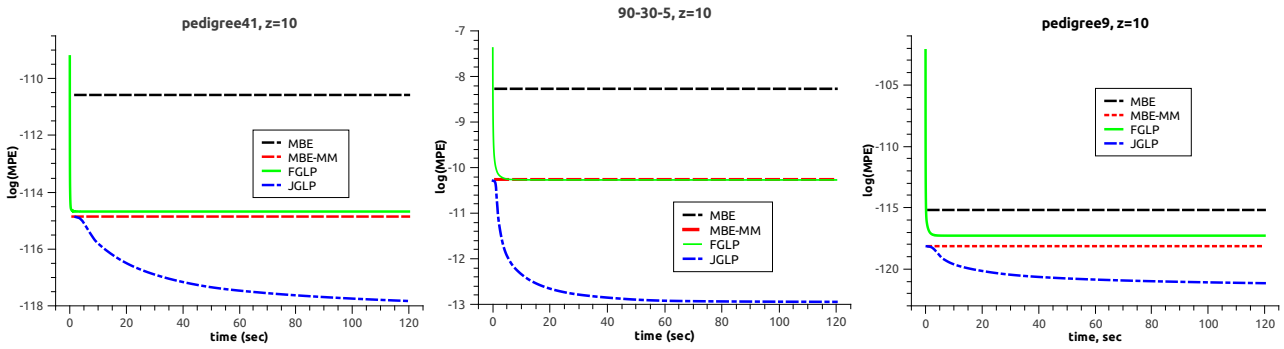
6 EMPIRICAL EVALUATION

We investigate the impact of single-pass and iterative LP-tightening in conjunction with the MBE scheme for the task of finding the most probable explanation (MPE) over Bayesian networks. Specifically, we evaluate the performance of MBE-MM, Factor Graph LP-Tightening (denoted FGLP) applied to the original functions, and Join Graph LP-Tightening (JGLP) which is applied over the mini-bucket-based join graph. We compare these three algorithms against each other and against "pure" MBE, both as stand-alone bounding algorithms (Section 6.1) and as generator of heuristic evaluation functions that guides search algorithms such as Branch and Bound (Section 6.2).

Our benchmark problems include three sets of instances from genetic linkage analysis networks (Fishelson & Geiger, 2002) (denoted `pedigrees`, `type4b` and `LargeFam`) and grid networks from the UAI 2008 competition (Darwiche *et al.*, 2008). In total we evaluated 10 `pedigrees`, 10 `type4` instances, 40 `LargeFam` instances and 32 `grid` networks. The algorithms were implemented in C++ (64-bit).

6.1 LP-TIGHTENING ALGORITHMS AS BOUNDING SCHEMES

We compare the upper bounds' accuracy obtained by the non-iterative MBE and MBE-MM schemes and against the iterative FGLP and JGLP schemes. The iterative schemes ran for 5, 300 and 3600 seconds.



(a) (b) (c)

Figure 2: Upper bound on optimum (log scale) as a function of time (sec). Non-iterative bounds MBE, MBE-MM are shown for comparison.

Table 1: Upper bound (log scale) and runtime (# seconds) for a typical set of instances, $z = 10$ and $z = 20$. FGLP is not affected by z. Lower values are better. OOM shows that the algorithm ran out of memory (4 Gb). We report the number of variables n, largest domain size k, and the induced width w along the ordering used.

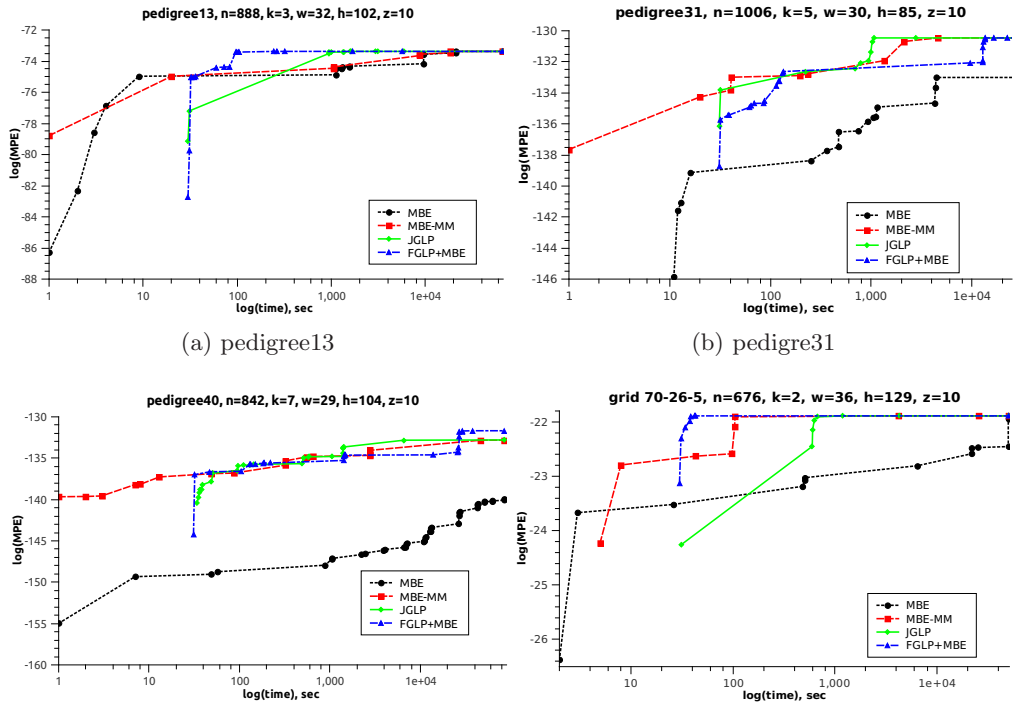
instance name	n	k	w	z	MBE	MBE-MM	FGLP time cut-offs			JGLP time cut-offs		
					UB/time	UB/time	5 UB	300 UB	3600 UB	5 UB	300 UB	3600 UB
75-25-5	625	2	34	10	-15.4553/1	-18.4089/1	-16.6853	-16.6854	-16.6854	-20.0289	-20.8364	-20.8364
				20	-17.4417/4	-20.0576/4				-20.0576	-20.1278	-20.7067
90-30-5	900	2	42	10	-8.2481/1	-10.2597/1	-10.2450	-10.2705	-10.2705	-11.8469	-12.9594	-13.015
				20	-9.7424/7	-11.6004/7				-11.6004	-11.6942	-12.5259
90-34-5	1156	2	48	10	-8.42007/1	-10.3708/1	-9.65003	-9.69458	-9.69458	-12.3469	-13.2262	-13.2883
				20	-9.58332/8	-12.3670/9				-12.3670	-12.5621	-13.1538
90-42-5	1764	2	60	10	-12.7401/1	-15.9680/1	-15.2480	-15.3653	-15.3653	-18.4100	-20.7714	-20.8136
				20	-14.6136/13	-18.5487/14				-18.5487	-18.7679	-19.9705
largeFam4_11_51	1002	4	40	10	-201.136/1	-211.656/1	-201.582	-201.673	-201.673	-211.671	-216.500	-217.176
				20	OOM	OOM				OOM	OOM	OOM
largeFam4_11_55	1114	4	38	10	-229.43/1	-242.489/1	-226.075	-226.328	-226.328	-242.657	-249.551	-250.453
				20	OOM	OOM				OOM	OOM	OOM
largeFam4_12_51	1461	4	56	10	-218.229/2	-239.896/3	-217.564	-217.740	-217.740	-239.896	-245.900	-253.153
				20	OOM	OOM				OOM	OOM	OOM
pedigree7	867	4	32	10	-105.854/1	-109.569/1	-110.179	-110.187	-110.187	-109.960	-110.810	-111.293
				20	-108.011/33	-111.120/42				OOM	OOM	OOM
pedigree13	888	3	32	10	-69.0973/1	-70.0999/1	-71.8561	-71.8591	-71.8591	-70.4581	-71.9869	-72.0374
				20	-69.8890/8	-71.1071/11				-71.1071	-71.1071	-71.3658
pedigree31	1006	5	30	10	-125.032/1	-126.629/1	-126.667	-126.678	-126.678	-126.644	-129.158	-129.277
				20	OOM	OOM				OOM	OOM	OOM
pedigree41	885	5	33	10	-110.156/1	-114.858/1	-114.681	-114.681	-114.681	-115.050	-118.133	-118.419
				20	-112.153/29	-117.638/37				OOM	OOM	OOM
type4_120_17	4302	5	23	10	-1128.22/1	-1203.08/1	-1049.34	-1049.85	-1049.86	-1203.21	-1221.21	-1223.69
				20	-1235.94/18	-1237.95/21				-1237.95	OOM	OOM
type4_170_23	6933	5	21	10	-1682.9/1	-1747.18/1	-1509.96	-1511.61	-1511.65	-1747.22	-1769.96	-1772.16
				20	-1783.18/7	-1783.76/7				-1783.76	-1783.76	-1783.76

In Table 1 we present a subset of the results obtained from all 4 instance sets for the z-bound values of $z = 10$ and $z = 20$. Results for $z = 15$ are similar and are omitted for lack of space. Note, that the value of z does not influence the results of FGLP, that runs on the original functions. For every problem instance described via its parameters (i.e., number of variables n, largest domain size k, and induced width w), for each algorithm we report the upper bound obtained and the CPU time (or time-bound) in seconds.

We observe clearly that for all instances MBE-MM is superior to pure MBE, since it produces considerably more accurate bounds in a comparable time. For most instances the MBE-MM bounds are also better than pure FGLP (at the comparable point in time), especially for $z = 20$, but only if the memory required by MBE-MM is not too high. As example behavior see instances `75-25-5` and `largeFam4_11_51`.

Figure 2 illustrates the anytime performance of the iterative schemes FGLP and JGLP. (Note that the single-pass algorithms MBE and MBE-MM do not change as a function of time.) As we expect, FGLP improves the bound rapidly, but converges to a suboptimal solution, while JGLP improves at a slower pace but eventually produces the most accurate results.

From both the table and the figure we see that given enough time and memory, JGLP produces the most accurate bounds eventually. However, when time and memory are bounded, MBE-MM can present a cost-effective hybrid of bounded inference (as indicated by



(a) pedigree13 (b) pedigre31

(c) pedigree40 (d) grid-75-26-5

Figure 3: Lower bounds by the AOBB as a function of time (sec) for z=10, memory limit 3 Gb, timelimit 24h.

the cluster size) and bounded LP-tightening. Note that the memory requirement of MBE-MM is slightly less than the comparable JGLP, since some memory can be freed by the single-pass algorithm; this behavior is evidenced in `pedigree7` and `pedigree41` for $z = 20$.

6.2 LP-TIGHTENING ALGORITHMS AS SEARCH GUIDING HEURISTICS

We also evaluated the impact of each of the bounding schemes as a generator of heuristic evaluation functions for the AOBB algorithm, as described in Section 5. We tested four schemes: AOBB guided by heuristics generated by pure MBE (denoted "AOBB-MBE"), AOBB guided by MBE and max-marginal matching heuristics ("AOBB-MBE-MM"), AOBB whose heuristics are generated from FGLP followed by MBE ("AOBB-FGLP+MBE") and AOBB guided by JGLP-produced heuristics ("AOBB-JGLP"). All the heuristic functions were generated in a preprocessing phase, prior to search. The iterative FGLP and JGLP algorithms were run for 30 seconds each. The total time bound for AOBB with each of the heuristics was set to 24 hours (including the preprocessing), memory limit was 3 Gb, and the mini-bucket z-bound parameters for generating the heuristics was set to $z \in \{10, 15, 20\}$.

In Table 2 we show some of the results. For space reasons and clarity we pick a representative set from the full 92 instances. The table reports the total runtime in seconds and the number of nodes expanded by each of the AOBB schemes.

We see that, as expected, the heuristic generated by MBE-MM are more cost-effective compared with "pure" MBE, both in runtime and nodes expanded. We see that the two iterative schemes are quite powerful as heuristic generators. For most instances FGLP+MBE presents a good balance[3]. But on some hard instances and as long as memory is available JGLP is the overall best-performing scheme.

We also observe the impact of the z bounds. As expected, with larger z we obtain more accurate heuristics, but this increases the memory requirements (see, for example, AOBB-JGLP on `pedigree13`).

Figure 3 shows the anytime behaviour of AOBB with each heuristic, plotting the solution value found over time (both on log scale) on four typical instances. Higher values are better. As expected, AOBB-MBE-MM has a more precise heuristic and consistently outputs better solutions faster compared with AOBB-MBE. Algorithms AOBB-FGLP-MBE and AOBB-JGLP perform very well as anytime schemes. We see

[3]We did not test a "pure" FGLP heuristic, since its performance should be inferior to FGLP-MBE (FGLP followed by pure MBE).

Table 2: Search time (seconds) / # nodes expanded for selected instances. FGLP and JGLP ran for 30 seconds. "OOM" indicates that search ran out of memory (3Gb) and "— / —" that it ran out of time (24h). In **bold** we highlight the best runtime for each instance, *italics* indicate the smallest search space explored.

Instances (n,k,w,h)	AOBB-MBE(z) AOBB-MBE-MM(z) AOBB-FGLP+MBE(z) AOBB-JGLP(z) z-bound=10	AOBB-MBE(z) AOBB-MBE-MM(z) AOBB-FGLP+MBE(z) AOBB-JGLP(z) z-bound=15	AOBB-MBE(z) AOBB-MBE-MM(z) AOBB-FGLP+MBE(z) AOBB-JGLP(z) z-bound=20
	time / # nodes	time / # nodes	time / # nodes
pedigree instances			
pedigree7	— / —	9404 / 1876188145	OOM
(867, 4, 32, 90)	2171 / 348425451	428 / 78953096	OOM
	805 / 140665826	**227** / *36619862*	OOM
	530 / 80597149	286 / 38350755	OOM
pedigree13	— / —	22799 / 5614980160	7229 / 1522450313
(888, 3, 32, 102)	66156 / 11726505961	8150 / 1441111422	704 / 164319080
	5658 / 905160506	926 / 182970673	**357** / *73658489*
	5911 / 1015334227	1939 / 366168237	OOM
pedigree31	— / —	— / —	OOM
(1006, 5, 30, 85)	61382 / 10617627744	3856 / 750931932	OOM
	24896 / 3695993630	**1033** / *188749113*	OOM
	2775 / 497649324	2337 / 435238548	OOM
type4 linkage instances			
type4b_120_17	— / —	— / —	— / —
(4072, 5, 24, 319)	— / —	— / —	**33** / *720778*
	— / —	— / —	71 / 1168656
	— / —	— / —	OOM
LargeFam linkage instances			
largeFam3_11_53	— / —	— / —	OOM
(1094, 3, 39, 71)	— / —	— / —	OOM
	— / —	44663 / 8080262337	OOM
	— / —	**10292** / *1878168857*	OOM
largeFam3_11_59	— / —	— / —	OOM
(1119, 3, 33, 73)	— / —	— / —	OOM
	— / —	59012 / 8098379409	OOM
	— / —	**22538** / *3025470612*	OOM
binary grid instances			
90-30-5	— / —	54415 / 10603123693	5853 / 1299094138
(900, 2, 42, 151)	8601 / 1790747055	423 / 97620783	**12** / 1125656
	5928 / 1084067942	337 / 67303699	47 / 2101919
	350 / 62930133	31 / 28688	48 / *7493*
90-42-5	— / —	— / —	— / —
(1764, 2, 60, 229)	— / —	62051 / 8399774202	2471 / 340122171
	— / —	17628 / 2349582057	651 / 93715978
	40 / 1411953	134 / *13038792*	OOM
90-50-5	— / —	— / —	— / —
(2500, 2, 74, 312)	— / —	— / —	— / —
	— / —	— / —	— / —
	— / —	**48781** / *4187198638*	OOM

that they output the first solution later than the other two schemes due to initial 30 seconds pre-processing step. However, if desired, a shorter time bound for computing heuristic can be used.

In summary, based on our empirical evaluation we can conclude that some level of LP-tightening can significantly improve the power of the MBE heuristics, yielding an improved search, sometimes by orders of magnitudes. The question of instance-based balance, namely tailoring the right level of z-bound and LP-tightening to the problem instance is clearly a central issue and a direction of future research. Overall, in this study we observed that MBE-MM always improves upon MBE, using comparable time and memory, while FGLP quickly converges and is less memory-consuming than the other schemes. On the other hand, given sufficient time and memory JGLP produces the tightest bound.

7 CONCLUSION

The paper presents the first systematic combination of iterative cost-shifting updates with elimination-order based clustering algorithms and provides extensive empirical evaluation demonstrating its effectiveness. Specifically, we present Join Graph Linear Programming, a new bounding scheme for optimization tasks in graphical models that combines MBE bounds with LP-based cost-shifting or soft arc-consistency. Empirically, larger clusters improved the iterative updates' performance in all instances. We also demonstrated that JGLP can often find better bounds faster than LP-tightening on the original model, even for relatively small z. Most importantly, we showed the algorithms' ability to improve informed search algorithms; without requiring significantly more computational power (for fixed z) than classical MBE they can drastically reduce the search space. Notably, the algorithm that used as a heuristic generator all three cost-shifting schemes in a sequence (FGLP+JGLP+MBE-MM) won the first place in all optimization categories in this year's Probabilistic Inference Challenge.

Acknowledgements. Work supported in part by NSF Grant IIS-1065618 and NIH grant 5R01HG004175-03.

References

Batra, D., Nowozin, S., & Kohli, P. 2011. Tighter relaxations for MAP-MRF inference: A local primal-dual gap based separation algorithm. *In: Conference on Uncertainty in Artificial Intelligence (AISTATS)*.

Bistarelli, S., Gennari, R., & Rossi, F. 2000. Constraint propagation for soft constraints: Generalization and termination conditions. *Principles and Practice of Constraint Programming–CP 2000*, 83–97.

Bodlaender, H. 2007. Treewidth: Structure and algorithms. *Structural Information and Communication Complexity*, 11–25.

Choi, Arthur, & Darwiche, Adnan. 2009. Approximating MAP by Compensating for Structural Relaxations. *Pages 351–359 of: Advances in Neural Information Processing Systems 22*.

Cooper, M., & Schiex, T. 2004. Arc consistency for soft constraints. *Artificial Intelligence*, **154**(1), 199–227.

Darwiche, A., Dechter, R., Choi, A., Gogate, V., & Otten, L. 2008. Results from the probablistic inference evaluation of UAI08, a web-report in http://graphmod.ics.uci.edu/uai08/Evaluation/Report. *In: UAI applications workshop*.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, **113**(1), 41–85.

Dechter, R. 2003. *Constraint processing*. Morgan Kaufmann.

Dechter, R., & Rish, I. 2003. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, **50**(2), 107–153.

Elidan, G., McGraw, I., & Koller, D. 2006. Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing. *In: Proc. UAI*.

Fishelson, M., & Geiger, D. 2002. Exact genetic linkage computations for general pedigrees. *Pages 189–198 of: International Conference on Intelligent Systems for Molecular Biology (ISMB)*.

Globerson, A., & Jaakkola, T. 2007. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *Advances in Neural Information Processing Systems*, **21**(1.6).

Gogate, V., & Dechter, R. 2004. A complete anytime algorithm for treewidth. *Pages 201–208 of: Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press.

Johnson, J., Malioutov, D., & Willsky, A. 2007. Lagrangian Relaxation for MAP Estimation in Graphical Models. *In: Allerton Conf. Comm. Control and Comput.*

Jojic, V., Gould, S., & Koller, D. 2010. Accelerated dual decomposition for MAP inference. *In: ICML*.

Kask, K., & Dechter, R. 2001. A general scheme for automatic generation of search heuristics from specification dependencies. *Artificial Intelligence*, **129**(1), 91–131.

Kask, K., Gelfand, A., Otten, L., & Dechter, R. 2011. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. *AAAI 2011*.

Komodakis, N., & Paragios, N. 2008. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. *Computer Vision–ECCV 2008*, 806–820.

Komodakis, N., Paragios, N., & Tziritas, G. 2007 (Oct.). MRF Optimization via Dual Decomposition: Message-Passing Revisited. *In: ICCV*.

Marinescu, Radu, & Dechter, Rina. 2009a. AND/OR Branch-and-Bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, **173**(16-17), 1457–1491.

Marinescu, Radu, & Dechter, Rina. 2009b. Memory intensive AND/OR search for combinatorial optimization in graphical models. *Artificial Intelligence*, **173**(16-17), 1492–1524.

Mateescu, Robert, Kask, Kalev, Gogate, Vibhav, & Dechter, Rina. 2010. Join-Graph Propagation Algorithms. *Journal of Artificial Intelligence Research (JAIR)*, **37**, 279–328.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Robertson, N., & Seymour, P.D. 1983. Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, **35**(1), 39–61.

Rollon, E., & Larrosa, J. 2006. Mini-bucket Elimination with Bucket Propagation. *Principles and Practice of Constraint Programming-CP 2006*, 484–498.

Schiex, T. 2000. Arc consistency for soft constraints. *Principles and Practice of Constraint Programming (CP2000)*, 411–424.

Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T., & Weiss, Y. 2008. Tightening LP relaxations for MAP using message passing. *UAI*.

Sontag, D., Globerson, A., & Jaakkola, T. 2010. Introduction to dual decomposition for inference. *Optimization for Machine Learning*.

Sontag, David, & Jaakkola, Tommi. 2009. Tree Block Coordinate Descent for MAP in Graphical Models. *Pages 544–551 of: AI & Statistics*. JMLR: W&CP 5.

Sutton, C., & McCallum, A. 2007. Improved Dynamic Schedules for Belief Propagation. *Pages 376–383 of: Proc. UAI*.

Tarlow, Daniel, Batra, Dhruv, Kohli, Pushmeet, & Kolmogorov, Vladimir. 2011 (June). Dynamic Tree Block Coordinate Ascent. *Pages 113–120 of: ICML*.

Wainwright, M.J., Jaakkola, T.S., & Willsky, A.S. 2005. MAP estimation via agreement on trees: message-passing and linear programming. *Information Theory, IEEE Transactions on*, **51**(11), 3697–3717.

Yarkony, J., Fowlkes, C., & Ihler, A. 2010. Covering Trees and Lower Bounds on Quadratic Assignment. *In: CVPR*.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. 2004 (May). *Constructing Free Energy Approximations and Generalized Belief Propagation Algorithms*. Tech. rept. 2004-040. MERL.

Algorithms for Approximate Minimization of the Difference Between Submodular Functions, with Applications

Rishabh Iyer
Dept. of Electrical Engineering
University of Washington
Seattle, WA-98175, USA

Jeff Bilmes
Dept. of Electrical Engineering
University of Washington
Seattle, WA-98175, USA

Abstract

We extend the work of Narasimhan and Bilmes [30] for minimizing set functions representable as a difference between submodular functions. Similar to [30], our new algorithms are guaranteed to monotonically reduce the objective function at every step. We empirically and theoretically show that the per-iteration cost of our algorithms is much less than [30], and our algorithms can be used to efficiently minimize a difference between submodular functions under various combinatorial constraints, a problem not previously addressed. We provide computational bounds and a hardness result on the multiplicative inapproximability of minimizing the difference between submodular functions. We show, however, that it is possible to give worst-case additive bounds by providing a polynomial time computable lower-bound on the minima. Finally we show how a number of machine learning problems can be modeled as minimizing the difference between submodular functions. We experimentally show the validity of our algorithms by testing them on the problem of feature selection with submodular cost features.

1 Introduction

Discrete optimization is important to many areas of machine learning and recently an ever growing number of problems have been shown to be expressible as submodular function minimization or maximization (e.g., [19, 23, 25, 28, 27, 29]). The class of submodular functions is indeed special since submodular function minimization is known to be polynomial time, while submodular maximization, although NP complete, admits constant factor approximation algorithms. Let $V = \{1, 2, \cdots, n\}$ refer a ground set, then $f : 2^V \rightarrow \mathbb{R}$ is said to be submodular if for sets $S, T \subseteq V$, $f(S)+f(T) \geq f(S \cup T)+f(S \cap T)$ (see [10] for details on submodular, supermodular, and modular functions). Submodular functions have a diminishing returns property, wherein the gain of an element in the context of bigger set is lesser than the gain of that element in the context of a smaller subset. This property occurs naturally in many applications in machine learning, computer vision, economics, operations research, etc.

In this paper, we address the following problem. Given two submodular functions f and g, and define $v(X) \triangleq f(X) - g(X)$, solve the following optimization problem:

$$\min_{X \subseteq V} [f(X) - g(X)] \equiv \min_{X \subseteq V} [v(X)]. \tag{1}$$

A number of machine learning problems involve minimization over a difference between submodular functions. The following are some examples:

- **Sensor placement with submodular costs:** The problem of choosing sensor locations A from a given set of possible locations V can be modeled [23, 24] by maximizing the mutual information between the chosen variables A and the unchosen set $V \backslash A$ (i.e., $f(A) = I(X_A; X_{V \backslash A})$). Alternatively, we may wish to maximize the mutual information between a set of chosen sensors X_A and a fixed quantity of interest C (i.e., $f(A) = I(X_A; C)$) under the assumption that the set of features X_A are conditionally independent given C [23]. These objectives are submodular and thus the problem becomes maximizing a submodular function subject to a cardinality constraint. Often, however, there are costs $c(A)$ associated with the locations that naturally have a diminishing returns property. For example, there is typically a discount when purchasing sensors in bulk. Moreover, there may be diminished cost for placing a sensor in a particular location given placement in certain other locations (e.g., the additional equipment needed to install a sensor in, say, a precarious environment

could be re-used for multiple sensor installations in like environments). Hence, along with maximizing mutual information, we also want to simultaneously minimize the cost and this problem can be addressed by minimizing the difference between submodular functions $f(A) - \lambda c(A)$ for tradeoff parameter λ.

- **Discriminatively structured graphical models and neural computation:** An application suggested in [30] and the initial motivation for this problem is to optimize the EAR criterion to produce a discriminatively structured graphical model. EAR is basically a difference between two mutual information functions (i.e., a difference between submodular functions). [30] shows how classifiers based on discriminative structure using EAR can significantly outperform classifiers based on generative graphical models. Note also that the EAR measure is the same as "synergy" in a neural code [2], widely used in neuroscience.

- **Feature selection:** Given a set of features $X_1, X_2, \cdots, X_{|V|}$, the feature selection problem is to find a small subset of features X_A that work well when used in a pattern classifier. This problem can be modeled as maximizing the mutual information $I(X_A; C)$ where C is the class. Note that $I(X_A; C) = H(X_A) - H(X_A|C)$ is always a difference between submodular functions. Under the naïve Bayes model, this function is submodular [23]. It is not submodular under general classifier models such as support vector machines (SVMs) or neural networks. Certain features, moreover, might be cheaper to use given that others are already being computed. For example, if a subset $S_i \subseteq V$ of the features for a particular information source i are spectral in nature, then once a particular $v \in S_i$ is chosen, the remaining features $S_i \setminus \{v\}$ may be relatively inexpensive to compute, due to grouped computational strategies such as the fast Fourier transform. Therefore, it might be more appropriate to use a submodular cost model $c(A)$. One such cost model might be $c(A) = \sum_i \sqrt{m(A \cap S_i)}$ where $m(j)$ would be the cost of computing feature j. Another might be $c(A) = \sum_i c_i \min(|A \cap S_i|, 1)$ where c_i is the cost of source i. Both offer diminishing cost for choosing features from the same information source. Such a cost model could be useful even under the naïve Bayes model, where $I(X_A; C)$ is submodular. Feature selection becomes a problem of maximizing $I(X_A; C) - \lambda c(A) = H(X_A) - [H(X_A|C) + \lambda c(A)]$, the difference between two submodular functions.

- **Probabilistic Inference:** We are given a distribution $p(x) \propto \exp(-v(x))$ where $x \in \{0,1\}^n$ and v is a pseudo-Boolean function [1]. It is desirable to compute $\text{argmax}_{x \in \{0,1\}^n} p(x)$ which means minimizing $v(x)$ over x, the most-probable explanation (MPE) problem [33]. If p factors with respect to a graphical model of tree-width k, then $v(x) = \sum_i v_i(x_C)$ where C_i is a bundle of indices such that $|C| \leq k+1$ and the sets $\{C_i\}_i$ form a junction tree, and it might be possible to solve inference using dynamic programming. If k is large and/or if hypertree factorization does not hold, then approximate inference is typically used [38]. On the other hand, defining $x(X) = \{x \in \{0,1\}^n : x_i = 1 \text{ whenever } i \in X\}$, if the set function $\bar{v}(X) = v(x(X))$ is submodular, then even if p has large tree-width, the MPE problem can be solved exactly in polynomial time [17]. This, in fact, is the basis behind inference in many computer vision models where v is often not only submodular but also has limited sized $|C_i|$. For example, for submodular v and if $|C_i| \leq 2$ then graph-cuts can solve the MPE problem extremely rapidly [22] and even some cases with v non-submodular [21]. An important challenge is to consider non-submodular v that can be minimized efficiently and for which there are approximation guarantees, a problem recently addressed in [18]. On the other hand, if v can be expressed as a difference between two submodular functions (which it can, see Lemma 3.1), or if such a decomposition can be computed (which it sometimes can, see Lemma 3.2), then a procedure to minimize the difference between two submodular functions offers new ways to solve probabilistic inference.

Previously, Narasimhan and Bilmes [30] proposed an algorithm inspired by the convex-concave procedure [39] to address Equation (1). This algorithm iteratively minimizes a submodular function by replacing the second submodular function g by it's modular lower bound. They also show that any set function can be expressed as a difference between two submodular functions and hence every set function optimization problem can be reduced to minimizing a difference between submodular functions. They show that this process converges to a local minima, however the convergence rate is left as an open question.

In this paper, we first describe tight modular bounds on submodular functions in Section 2, including lower bounds based on points in the base polytope as used in [30], and recent upper bounds first described in a result in [15]. In section 2.2, we describe the submodular-supermodular procedure proposed in [30]. We further provide a constructive procedure for finding the submodular functions f and g for any arbitrary set function v. Although our construction is NP hard in general, we show how for certain classes of set functions v, it is possible to find the decompositions f and g in polynomial time. In Section 4, we propose two new algorithms both of which are guaranteed to monotonically reduce

the objective at every iteration and which converge to a local minima. Further we note that the per-iteration cost of our algorithms is in general much less than [30], and empirically verify that our algorithms are orders of magnitude faster on real data. We show that, unlike in [30], our algorithms can be extended to easily optimize equation (1) under cardinality, knapsack, and matroid constraints. Moreover, one of our algorithms can actually handle complex combinatorial constraints, such as spanning trees, matchings, cuts, etc. Further in Section 5, we give a hardness result that there does not exist any polynomial time algorithm with any polynomial time multiplicative approximation guarantees unless P=NP, even when it is easy to find or when we are given the decomposition f and g, thus justifying the need for heuristic methods to solve this problem. We show, however, that it is possible to get additive bounds by showing polynomial time computable upper and lower bound on the optima. We also provide computational bounds for all our algorithms (including the submodular-supermodular procedure), a problem left open in [30].

Finally we perform a number of experiments on the feature selection problem under various cost models, and show how our algorithms used to maximize the mutual information perform better than greedy selection (which would be near optimal under the naïve Bayes assumptions) and with less cost.

2 Modular Upper and Lower bounds

The Taylor series approximation of a convex function provides a natural way of providing lower bounds on such a function. In particular the first order Taylor series approximation of a convex function is a lower bound on the function, and is linear in x for a given y and hence given a convex function ϕ, we have:

$$\phi(x) \geq \phi(y) + \langle \nabla\phi(y), x - y \rangle. \tag{2}$$

Surprisingly, any submodular function has both a tight lower [6] **and** upper bound [15], unlike strict convexity where there is only a tight first order lower bound.

2.1 Modular Lower Bounds

Recall that for submodular function f, the submodular polymatroid, base polytope and the sub-differential with respect to a set Y [10] are respectively:

$$\mathcal{P}_f = \{x : x(S) \leq f(S), \forall S \subseteq V\} \tag{3}$$

$$\mathcal{B}_f = \mathcal{P}_f \cap \{x : x(V) = f(V)\} \tag{4}$$

$$\partial f(Y) = \{y \in \mathbb{R}^V : \forall X \subseteq V, f(Y) - y(Y) \leq f(X) - y(X)\}$$

The extreme points of this sub-differential are easy to find and characterize, and can be obtained from a greedy algorithm ([6, 10]) as follows:

Theorem 2.1. *([10], Theorem 6.11) A point y is an extreme point of $\partial f(Y)$, iff there exists a chain $\emptyset = S_0 \subset S_1 \subset \cdots \subset S_n$ with $Y = S_j$ for some j, such that $y(S_i \setminus S_{i-1}) = y(S_i) - y(S_{i-1}) = f(S_i) - f(S_{i-1})$.*

Let σ be a permutation of V and define $S^\sigma_i = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$ as σ's chain containing Y, meaning $S^\sigma_{|Y|} = Y$ (we say that σ's chain contains Y). Then we can define a sub-gradient h^f_Y corresponding to f as:

$$h^f_{Y,\sigma}(\sigma(i)) = \begin{cases} f(S^\sigma_1) & \text{if } i = 1 \\ f(S^\sigma_i) - f(S^\sigma_{i-1}) & \text{otherwise} \end{cases}.$$

We get a modular lower bound of f as follows:

$$h^f_{Y,\sigma}(X) \leq f(X), \forall X \subseteq V, \text{ and } \forall i, h^f_{Y,\sigma}(S^\sigma_i) = f(S^\sigma_i),$$

which is parameterized by a set Y and a permutation σ. Note $h(X) = \sum_{i \in X} h(i)$, and $h^f_{Y,\sigma}(Y) = f(Y)$. Observe the similarity to convex functions, where a linear lower bound is parameterized by a vector y.

2.2 Modular Upper Bounds

For f submodular, [31] established the following:

$$f(Y) \leq f(X) - \sum_{j \in X \setminus Y} f(j | X \setminus j) + \sum_{j \in Y \setminus X} f(j | X \cap Y),$$

$$f(Y) \leq f(X) - \sum_{j \in X \setminus Y} f(j | (X \cup Y) \setminus j) + \sum_{j \in Y \setminus X} f(j | X)$$

Note that $f(A|B) \triangleq f(A \cup B) - f(B)$ is the gain of adding A in the context of B. These upper bounds in fact characterize submodular functions, in that a function f is a submodular function *iff* it follows either of the above bounds. Using the above, two tight modular upper bounds ([15]) can be defined as follows:

$$f(Y) \leq m^f_{X,1}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j | X \setminus j) + \sum_{j \in Y \setminus X} f(j | \emptyset),$$

$$f(Y) \leq m^f_{X,2}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j | V \setminus j) + \sum_{j \in Y \setminus X} f(j | X).$$

Hence, this yields two tight (at set X) modular upper bounds $m^f_{X,1}, m^f_{X,2}$ for any submodular function f. For briefness, when referring either one we use m^f_X.

3 Submodular-Supermodular Procedure

We now review the submodular-supermodular procedure [30] to minimize functions expressible as a difference between submodular functions (henceforth called DS functions). Interestingly, any set function

Algorithm 1 The submodular-supermodular (SubSup) procedure [30]

1: $X^0 = \emptyset$; $t \leftarrow 0$;
2: **while** not converged (i.e., $(X^{t+1} \neq X^t)$) **do**
3: Randomly choose a permutation σ^t whose chain contains the set X^t.
4: $X^{t+1} := \text{argmin}_X f(X) - h^g_{X^t,\sigma^t}(X)$
5: $t \leftarrow t + 1$
6: **end while**

can be expressed as a DS function using suitable submodular functions as shown below. The result was first shown in [30] using the Lovász extension. We here give a new combinatorial proof, which avoids Hessians of polyhedral convex functions and which provides a way of constructing (a non-unique) pair of submodular functions f and g for an arbitrary set function v.

Lemma 3.1. *[30] Given any set function v, it can be expressed as a DS functions $v(X) = f(X) - g(X), \forall X \subseteq V$ for some submodular functions f and g.*

Proof. Given a set function v, we can define $\alpha = \min_{X \subset Y \subseteq V \setminus j} v(j|X) - v(j|Y)$[1]. Clearly $\alpha < 0$, since otherwise v would be submodular. Now consider any (strictly) submodular function g, i.e., one having $\beta = \min_{X \subset Y \subseteq V \setminus j} g(j|X) - g(j|Y) > 0$. Define $f'(X) = v(X) + \frac{|\alpha'|}{\beta} g(X)$ with any $\alpha' \leq \alpha$. Now it is easy to see that f' is submodular since $\min_{X \subset Y \subseteq V \setminus j} f'(j|X) - f'(j|Y) \geq \alpha + |\alpha'| \geq 0$. Hence $v(X) = f'(X) - \frac{|\alpha'|}{\beta} g(X)$, is a difference between two submodular functions. □

The above proof requires the computation of α and β which has, in general, exponential complexity. Using the construction above, however, it is easy to find the decomposition f and g under certain conditions on v.

Lemma 3.2. *If α or at least a lower bound on α for any set function v can be computed in polynomial time, functions f and g corresponding to v can obtained in polynomial time.*

Proof. Define g as $g(X) = \sqrt{|X|}$. Then $\beta = \min_{X \subset Y \subseteq V \setminus j} \sqrt{|X|+1} - \sqrt{|X|} - \sqrt{|Y|+1} + \sqrt{|Y|} = \min_{X \subset V \setminus j} \sqrt{|X|+1} - \sqrt{|X|} - \sqrt{|X|+2} + \sqrt{|X|+1} = 2\sqrt{n-1} - \sqrt{n} - \sqrt{n-2}$. The last inequality follows since the smallest difference in gains will occur at $|X| = n - 2$. Hence β is easily computed, and given a lower bound on α, from Lemma 3.1 the decomposition can be obtained in polynomial time. A similar argument holds for g being other concave functions over $|X|$. □

[1] We denote $j, X, Y : X \subset Y \subseteq V \setminus \{j\}$ by $X \subset Y \subseteq V \setminus j$.

The submodular supermodular (SubSup) procedure is given in Algorithm 1. At every step of the algorithm, we minimize a submodular function which can be performed in strongly polynomial time [32, 35] although the best known complexity is $O(n^5\eta + n^6)$ where η is the cost of a function evaluation. Algorithm 1 is guaranteed to converge to a local minima and moreover the algorithm monotonically decreases the function objective at every iteration, as we show below.

Lemma 3.3. *[30] Algorithm 1 is guaranteed to decrease the objective function at every iteration. Further, the algorithm is guaranteed to converge to a local minima by checking at most $O(n)$ permutations at every iteration.*

Due to space constraints, we omit the proof of this lemma which is in any case described in [30, 13].

Algorithm 1 requires performing a submodular function minimization at every iteration which while polynomial in n is (due to the complexity described above) not practical for large problem sizes. So while the algorithm reaches a local minima, it can be costly to find it. A desirable result, therefore, would be to develop new algorithms for minimizing DS functions, where the new algorithms have the same properties as the SubSup procedure but are much faster in practice. We give this in the following sections.

4 Alternate algorithms for minimizing DS functions

In this section we propose two new algorithms to minimize DS functions, both of which are guaranteed to monotonically reduce the objective at every iteration and converge to local minima. We briefly describe these algorithms in the subsections below.

4.1 The supermodular-submodular (SupSub) procedure

In the submodular-supermodular procedure we iteratively minimized $f(X) - g(X)$ by replacing g by it's modular lower bound at every iteration. We can instead replace f by it's modular upper bound as is done in Algorithm 2, which leads to the *supermodular-submodular* procedure.

In the SupSub procedure, at every step we perform submodular maximization which, although NP complete to solve exactly, admits a number of fast constant factor approximation algorithms [7, 8]. Notice that we have two modular upper bounds and hence there are a number of ways we can choose between them. One way is to run both maximization procedures with the two modular upper bounds at every iteration in parallel,

Algorithm 2 The supermodular-submodular (SupSub) procedure

1: $X^0 = \emptyset$; $t \leftarrow 0$;
2: **while** not converged (i.e., $(X^{t+1} \neq X^t)$) **do**
3: $\quad X^{t+1} := \text{argmin}_X \, m^f_{X^t}(X) - g(X)$
4: $\quad t \leftarrow t + 1$
5: **end while**

and choose the one which is better. Here by better we mean the one in which the function value is lesser. Alternatively we can alternate between the two modular upper bounds by first maximizing the expression using the first modular upper bound, and then maximize the expression using the second modular upper bound. Notice that since we perform approximate submodular maximization at every iteration, we are not guaranteed to monotonically reduce the objective value at every iteration. If, however, we ensure that at every iteration we take the next step only if the objective v does not increase, we will restore monotonicity at every iteration. Also, in some cases we converge to local optima as shown in the following theorem.

Theorem 4.1. *Both variants of the supermodular-submodular procedure (Algorithm 2) monotonically reduces the objective value at every iteration. Moreover, assuming a submodular maximization procedure in line 3 that reaches a local maxima of $m^f_{X^t}(X) - g(X)$, then if Algorithm 2 does not improve under both modular upper bounds then it reaches a local optima of v.*

Proof. For either modular upper bound, we have:

$$\begin{aligned} f(X^{t+1}) - g(X^{t+1}) &\overset{a}{\leq} m^f_{X^t}(X^{t+1}) - g(X^{t+1}) \\ &\overset{b}{\leq} m^f_{X^t}(X^t) - g(X^t) \\ &\overset{c}{=} f(X^t) - g(X^t), \end{aligned}$$

where (a) follows since $f(X^{t+1}) \leq m^f_{X^t}(X^{t+1})$, and (b) follows since we assume that we take the next step only if the objective value does not increase and (c) follows since $m^f_{X^t}(X^t) = f(X^t)$ from the tightness of the modular upper bound.

To show that this algorithm converges to a local minima, we assume that the submodular maximization procedure in line 3 converges to a local maxima. Then observe that if the objective value does not decrease in an iteration under both upper bounds, it implies that $m^f_{X^t}(X^t) - g(X^t)$ is already a local optimum in that (for both upper bounds) we have $m^f_{X^t}(X^t \cup j) - g(X^t \cup j) \geq m^f_{X^t}(X^t) - g(X^t), \forall j \notin X^t$ and $m^f_{X^t}(X^t \backslash j) - g(X^t \backslash j) \geq m^f_{X^t}(X^t) - g(X^t), \forall j \in X^t$. Note that $m^f_{X^t,1}(X^t \backslash j) = f(X^t) - f(j|X^t \backslash j) = f(X^t \backslash j)$ and $m^f_{X^t,2}(X^t \cup j) = f(X^t) + f(j|X^t) = f(X^t \cup j)$ and hence if both modular upper bounds are at a local optima, it implies $f(X^t) - g(X^t) = m^f_{X^t,1}(X^t) - g(X^t) \leq m^f_{X^t,1}(X^t \backslash j) - g(X^t \backslash j) = f(X^t \backslash j) - g(X^t \backslash j)$. Similarly $f(X^t) - g(X^t) = m^f_{X^t,2}(X^t) - g(X^t) \leq m^f_{X^t,2}(X^t \cup j) - g(X^t \cup j) = f(X^t \cup j) - g(X^t \cup j)$. Hence X^t is a local optima for $v(X) = f(X) - g(X)$, since $v(X^t) \leq v(X^t \cup j)$ and $v(X^t) \leq v(X^t \backslash j)$. □

To ensure that we take the largest step at each iteration, we can use the recently proposed tight (1/2)-approximation algorithm in [7] for unconstrained non-monotone submodular function maximization — this is the best possible in polynomial time for the class of submodular functions independent of the P=NP question. The algorithm is a form of bi-directional randomized greedy procedure and, most importantly for practical considerations, is linear time [7]. Lastly, note that this algorithm is closely related to a local search heuristic for submodular maximization [8]. In particular, if instead of using the greedy algorithm entirely at every iteration, we take only one local step, we get a local search heuristic. Hence, via the SupSub procedure, we may take larger steps at every iteration as compared to a local search heuristic.

4.2 The modular-modular (ModMod) procedure

The submodular-supermodular procedure and the supermodular-submodular procedure were obtained by replacing g by it's modular lower bound and f by it's modular upper bound respectively. We can however replace both of them by their respective modular bounds, as is done in Algorithm 3.

Algorithm 3 Modular-Modular (ModMod) procedure

1: $X^0 = \emptyset$; $t \leftarrow 0$;
2: **while** not converged (i.e., $(X^{t+1} \neq X^t)$) **do**
3: $\quad$ Choose a permutation σ^t whose chain contains the set X^t.
4: $\quad X^{t+1} := \text{argmin}_X \, m^f_{X^t}(X) - h^g_{X^t,\sigma^t}(X)$
5: $\quad t \leftarrow t + 1$
6: **end while**

In this algorithm at every iteration we minimize only a modular function which can be done in $O(n)$ time, so this is extremely easy (i.e., select all negative elements for the smallest minimum, or all non-positive elements for the largest minimum). Like before, since we have two modular upper bounds, we can use any of the variants discussed in the subsection above. Moreover, we are still guaranteed to monotonically decrease the objective at every iteration and converge to a local minima.

Theorem 4.2. *Algorithm 3 monotonically decreases the function value at every iteration. If the function value does not increase on checking $O(n)$ different permutations with different elements at adjacent positions and with both modular upper bounds, then we have reached a local minima of v.*

Proof. Again we can use similar reasoning as the earlier proofs and observe that:

$$\begin{aligned} f(X^{t+1}) - g(X^{t+1}) &\leq m^f_{X^t}(X^{t+1}) - h^g_{X^t,\sigma^t}(X^{t+1}) \\ &\leq m^f_{X^t}(X^t) - h^g_{X^t,\sigma^t}(X^t) \\ &= f(X^t) - g(X^t) \end{aligned}$$

We see that considering $O(n)$ permutations each with different elements at $\sigma^t(|X^t|-1)$ and $\sigma^t(|X^t|+1)$, we essentially consider all choices of $g(X^t \cup j)$ and $g(X^t \backslash j)$, since $h^g_{X^t,\sigma^t}(S_{|X^t|+1}) = f(S_{|X^t|+1})$ and $h^g_{X^t,\sigma^t}(S_{|X^t|-1}) = f(S_{|X^t|-1})$. Since we consider both modular upper bounds, we correspondingly consider every choice of $f(X^t \cup j)$ and $f(X^t \backslash j)$. Note that at convergence we have that $m^f_{X^t}(X^t) - h^g_{X^t,\sigma^t}(X^t) \leq m^f_{X^t}(X) - h^g_{X^t,\sigma^t}(X), \forall X \subseteq V$ for $O(n)$ different permutations and both modular upper bounds. Correspondingly we are guaranteed that (since the expression is modular) $\forall j \notin X^t, v(j|X^t) \geq 0$ and $\forall j \in X^t, v(j|X^t \backslash j) \geq 0$, where $v(X) = f(X) - g(X)$. Hence the algorithm converges to a local minima. □

An important question is the choice of the permutation σ^t at every iteration X^t. We observe experimentally that the quality of the algorithm depends strongly on the choice of permutation. Observe that $f(X) - g(X) \leq m^f_{X^t}(X) - h^g_{X^t,\sigma^t}(X)$, and $f(X^t) - g(X^t) = m^f_{X^t}(X^t) - h^g_{X^t,\sigma^t}(X^t)$. Hence, we might obtain the greatest local reduction in the value of v by choosing permutation $\sigma^* \in \text{argmin}_\sigma \min_X (m^f_{X^t}(X) - h^g_{X^t,\sigma^t}(X))$, or the one which maximizes $h^g_{X^t,\sigma^t}(X)$. We in fact might expect that choosing σ^t ordered according to greatest gains of g, with respect to X^t, we would achieve greater descent at every iteration. Another choice is to choose the permutation σ based on the ordering of gains of v (or even $m^f_{X^t}$). Through the former we are guaranteed to at least progress as much as the local search heuristic. Indeed, we observe in practice that the first two of these heuristics performs much better than a random permutation for both the ModMod and the SubSup procedure, thus addressing a question raised in [30] about which ordering to use. Practically for the feature selection problem, the second heuristic seems to work the best.

4.3 Constrained minimization of a difference between submodular functions

In this section we consider the problem of minimizing the difference between submodular functions subject to constraints. We first note that the problem of minimizing a submodular function under even simple cardinality constraints in NP hard and also hard to approximate [36]. Since there does not yet seem to be a reasonable algorithm for constrained submodular minimization at every iteration, it is unclear how we would use Algorithm 1. However the problem of submodular maximization under cardinality, matroid, and knapsack constraints though NP hard admits a number of constant factor approximation algorithms [31, 26] and correspondingly the cardinality constraints can be easily introduced in Algorithm 2. Moreover, since a non-negative modular function can be easily, directly and even exactly optimized under cardinality, knapsack and matroid constraints [16], Algorithm 3 can also easily be utilized. In addition, since problems such as finding the minimum weight spanning tree, min-cut in a graph, etc., are polynomial time algorithms in a number of cases, Algorithm 3 can be used when minimizing a non-negative function v expressible as a difference between submodular functions under combinatorial constraints. If v is non-negative, then so is its modular upper bound, and then the ModMod procedure can directly be used for this problem — each iteration minimizes a non-negative modular function subject to combinatorial constraints which is easy in many cases [16, 14].

5 Theoretical results

In this section we analyze the computational and approximation bounds for this problem. For simplicity we assume that the function v is normalized, i.e $v(\emptyset) = 0$. Hence we assume that v achieves it minima at a negative value and correspondingly the approximation factor in this case will be less than 1.

We note in passing that the results in this section are mostly negative, in that they demonstrate theoretically how complex a general problem such as $\min_X [f(X) - g(X)]$ is, even for submodular f and g. In this paper, rather than consider these hardness results pessimistically, we think of them as providing justification for the heuristic procedures given in Section 4 and [30]. In many cases, inspired heuristics can yield good quality and hence practically useful algorithms for real-world problems. For example, the ModMod procedure (Algorithm 3) and even the SupSub procedure (Algorithm 2) can scale to very large problem sizes, and thus can provide useful new strategies for the applications listed in Section 1.

5.1 Hardness

Observe that the class of DS functions is essentially the class of general set functions, and hence the problem of finding optimal solutions is NP-hard. This is not surprising since general set function minimization is inapproximable and there exist a large class of functions where all (adaptive, possibly randomized) algorithms perform arbitrarily poorly in polynomial time [37]. Clearly as is evident from Theorem 3.1, even the problem of finding the submodular functions f and g requires exponential complexity. We moreover show in the following theorem, however, that this problem is multiplicatively inapproximable even when the functions f and g are easy to find.

Theorem 5.1. *Unless $P = NP$, there cannot exist any polynomial time approximation algorithm for $\min_X v(X)$ where $v(X) = [f(X) - g(X)]$ is a positive set function and f and g are given submodular functions. In particular, let n be the size of the problem instance, and $\alpha(n) > 0$ be any positive polynomial time computable function of n. If there exists a polynomial-time algorithm which is guaranteed to find a set $X' : f(X') - g(X') < \alpha(n)OPT$, where OPT=$\min_X f(X) - g(X)$, then $P = NP$.*

The proof of this theorem is in [13]. In fact we show below that independent of the $P = NP$ question, there cannot exist a sub-exponential time algorithm for this problem. The theorem below gives information theoretic hardness for this problem.

Theorem 5.2. *For any $0 < \epsilon < 1$, there cannot exist any deterministic (or possibly randomized) algorithm for $\min_X[f(X) - g(X)]$ (where f and g are given submodular functions), that always finds a solution which is at most $\frac{1}{\epsilon}$ times the optimal, in fewer than $e^{\epsilon^2 n/8}$ queries.*

Again the proof of this theorem is in [13]. Essentially the theorems above say that even when we are given (or can easily find) a decomposition such that $v(X) = f(X) - g(X)$, there exist set functions such that any algorithm (either adaptive or randomized) will perform arbitrarily poorly and this problem is inapproximable. Hence any algorithm trying to find the global optimum for this problem [3] can only be exponential.

5.2 Polynomial time lower and upper bounds

The decomposition theorem of [5] shows that any submodular function can be decomposed into a modular function plus a monotone non-decreasing and *totally normalized* polymatroid rank function. Specifically, given submodular f, g we have $f'(X) \triangleq f(X) - \sum_{j \in X} f(j|V\backslash j)$ and $g'(X) \triangleq g(X) - \sum_{j \in X} g(j|V\backslash j)$ with f', g' being totally normalized polymatroid rank functions. Hence we have: $v(X) = f'(X) - g'(X) + k(X)$, with modular $k(X) = \sum_{j \in X} v(j|V\backslash j)$.

The algorithms in the previous sections are all based on repeatedly finding upper bounds for v. The following lower bounds directly follow from the results above. (The proof of this is in [13])

Theorem 5.3. *We have the following two lower bounds on the minimizers of $v(X) = f(X) - g(X)$:*

$$\min_X v(X) \geq \min_X f'(X) + k(X) - g'(V)$$
$$\min_X v(X) \geq f'(\emptyset) - g'(V) + \sum_{j \in V} \min(k(j), 0)$$

The above lower bounds essentially provide bounds on the minima of the objective and thus can be used to obtain an additive approximation guarantee. The algorithms described in this paper are all polynomial time algorithms (as we show below) and correspondingly from the bounds above we can get an estimate on how far we are from the optimal.

5.3 Computational Bounds

We now provide computational bounds for ϵ-approximate versions of our algorithms. Note that this was left as an open question in [30]. Finding the local minimizer of DS functions is PLS complete since it generalizes the problem of finding the local optimum of the MAX-CUT problem [34]. However we show that an ϵ-approximate version of this algorithm will converge in polynomial time.

Definition 5.1. *An ϵ-approximate version of an iterative monotone non-decreasing algorithm for minimizing a set function v is defined as a version of that algorithm, where we proceed to step $t+1$ only if $v(X^{t+1}) \leq v(X^t)(1+\epsilon)$.*

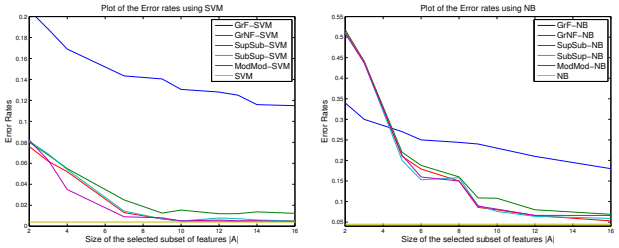
Note that the ϵ-approximate versions of algorithms 1, 2 and 3, are guaranteed to converge to ϵ-approximate local optima. W.l.o.g., assume that $X^0 = \emptyset$. Then we have the following computational bounds:

Theorem 5.4. *The ϵ-approximate versions of algorithms 1, 2 and 3 have a worst case complexity of $O(\frac{\log(|M|/|m|)}{\epsilon} T)$, where $M = f'(\emptyset) + \sum_{j \in V} \min(v(j|V\backslash j), 0) - g'(V)$, $m = v(X^1)$ and $O(T)$ is the complexity of every iteration of the algorithm (which corresponds to respectively the submodular minimization, maximization, or modular minimization in algorithms 1, 2 and 3)..*

The proof of this theorem is in [13]

Observe that for the algorithms we use, $O(T)$ is strongly polynomial in n. The best strongly polynomial time algorithm for submodular function minimization is

$O(n^5\eta + n^6)$ [32] (the lower bound is currently unknown). Further the worst case complexity of the greedy algorithm for maximization is $O(n^2)$ while the complexity of modular minimization is just $O(n)$. Note finally that these are worst case complexities and actually the algorithms run much faster in practice.



(a) SVM (b) NB

Figure 1: Plot showing the accuracy rates vs. the number of features on the Mushroom data set.

6 Experiments

We test our algorithms on the feature subset selection problem in the supervised setting. Given a set of features $X_V = \{X_1, X_2, \cdots, X_{|V|}\}$, we try to find a subset of these features A which has the most information from the original set X_V about a class variable C under constraints on the size or cost of A. Normally the number of features $|V|$ is quite large and thus the training and testing time depend on $|V|$. In many cases, however, there is a strong correlation amongst features and not every feature is novel. We can thus perform training and testing with a much smaller number of features $|A|$ while obtaining (almost) the same error rates.

The question is how to find the most representative set of features A. The mutual information between the chosen set of features and the target class C, $I(X_A; C)$, captures the relevance of the chosen subset of features. In most cases the selected features are not independent given the class C so the naïve Bayes assumption is not applicable, meaning this is not a pure submodular optimization problem. As mentioned in Section 1, $I(X_A; C)$ can be exactly expressed as a difference between submodular functions $H(X_A)$ and $H(X_A|C)$.

6.1 Modular Cost Feature Selection

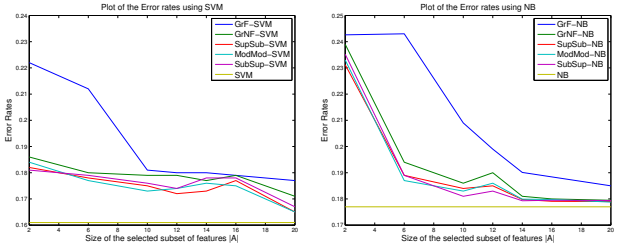
In this subsection, we look at the problem of maximizing $I(X_A; C) - \lambda|A|$, as a regularized feature subset selection problem. Note that a mutual information $I(X_A; C)$ query can easily be estimated from the data by just a single sweep through this data. Further we have observed that using techniques such as Laplace smoothing helps to improve mutual information estimates without increasing computation. In these experiments, therefore, we estimate the mutual information directly from the data and run our algorithms to find the representative subset of features.

We compare our algorithms on two data sets, i.e., the Mushroom data set [12] and the Adult data set [20] obtained from [9]. The Mushroom data set has 8124 examples with 112 features, while the Adult data set has 32,561 examples with 123 features. In our experiments we considered subsets of features of sizes between 5%-20% of the total number of features by varying λ. We tested the following algorithms for the feature subset selection problem. We considered two formulations of the mutual information, one under naïve Bayes, where the conditional entropy $H(X_A|C)$ can be written as $H(X_A|C) = \sum_{j \in A} H(X_i|C)$ and another where we do not assume such factorization. We call these two formulations *factored* and *non-factored* respectively. We then considered the simple greedy algorithm, of iteratively adding features at every step to the factored and non-factored mutual information, which we call GrF and GrNF respectively. Lastly, we use the new algorithms presented in this paper on the non-factored mutual information.

We then compare the results of the greedy algorithms with those of the three algorithms for this problem, using two pattern classifiers based on either a linear kernel SVM (using [4]) or a naïve Bayes (NB) classifier. We call the results obtained from the supermodular-submodular heuristic as "SupSub", the submodular-supermodular procedure [30] as "SubSup", and the modular-modular objective as "ModMod." In the SubSup procedure, we use the minimum norm point algorithm [11] for submodular minimization, and in the SubSup procedure, we use the optimal algorithm of [7] for submodular maximization. We observed that the three heuristics generally outperformed the two greedy procedures, and also that GRF can perform quite poorly, thus justifying our claim that the naïve Bayes assumption can be quite poor. This also shows that although the greedy algorithm in that case is optimal, the features are correlated given the class and hence modeling it as a difference between submodular functions gives the best results. We also observed that the SupSub and ModMod procedures perform comparably to the SubSup procedure, while the SubSup procedure is *much* slower in practice. Comparing the running times, the ModMod and the SupSub procedure are each a few times slower then the greedy algorithm (ModMod is slower due computing the modular semigradients), while the SubSup procedure is around 100 times slower. The SubSup procedure is slower due to general submodular function minimization which can be quite slow.

The results for the Mushroom data set are shown in

Figure 1. We performed a 10 fold cross-validation on the entire data set and observed that when using all the features SVM gave an accuracy rate of 99.6% while the all-feature NB model had an accuracy rate of 95.5%. The results for the Adult database are in Figure 2. In this case with the entire set of features the accuracy rate of SVM on this data set is 83.9% and NB is 82.3%.

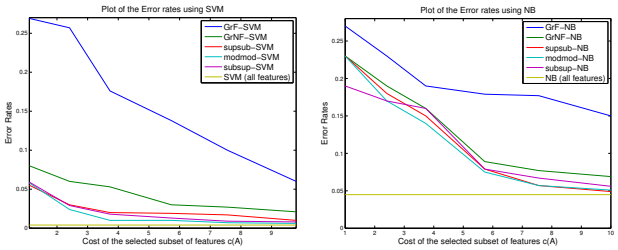


(a) SVM (b) NB

Figure 2: Plot showing the accuracy rates vs. the number of features on the Adult data set.

In the mushroom data, the SVM classifier significantly outperforms the NB classifier and correspondingly GrF performs much worse than the other algorithms. Also, in most cases the three algorithms outperform GrNF. In the adult data set, both the SVM and NB perform comparably although SVM outperforms NB. However in this case also we observe that our algorithms generally outperform GrF and GrNF.

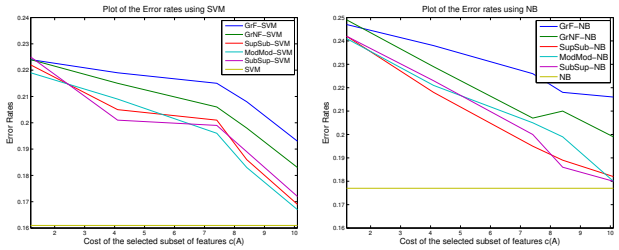
6.2 Submodular cost feature selection



(a) SVM (b) NB

Figure 3: Plot showing the accuracy rates vs. the cost of features for the Mushroom data set

We perform synthetic experiments for the feature subset selection problem under submodular costs. The cost model we consider is $c(A) = \sum_i \sqrt{m(A \cap S_i)}$. We partitioned V into sets $\{S_i\}_i$ and chose the modular function m randomly. In this set of experiments, we compare the accuracy of the classifiers vs. the *cost* associated with the choice of features for the algorithms. Recall, with simple (modular) cardinality costs the



(a) SVM (b) NB

Figure 4: Plot showing the accuracy rates vs. the cost of features for the Adult data set

greedy algorithms performed decently in comparison to our algorithms in the adult data set, where the NB assumption is reasonable. However with submodular costs, the objective is no longer submodular even under the NB assumption and thus the greedy algorithms perform much worse. This is unsurprising since the greedy algorithm is approximately optimal only for monotone submodular functions. This is even more strongly evident from the results of the mushrooms data-set (Figure 3)

7 Discussion

We have introduced new algorithms for optimizing the difference between two submodular functions, provided new theoretical understanding that provides some justification for heuristics, have outlined applications that can make use of our procedures, and have tested in the case of feature selection with modular and submodular cost features. Our new ModMod procedure is fast at each iteration and experimentally does about as well as the SupSub and SubSup procedures. The ModMod procedure, moreover, can also be used under various combinatorial constraints, and therefore the ModMod procedure may hold the greatest promise as a practical heuristic. An alternative approach, not yet evaluated, would be to try the convex-concave procedure [39] on the Lovász extensions of f and g since subgradients in such case are so easy to obtain.

Acknowledgments: We thank Andrew Guillory, Manas Joglekar, Stefanie Jegelka, and the rest of the submodular group at UW for discussions. This material is based upon work supported by the National Science Foundation under Grant No. (IIS-1162606), and is also supported by a Google, a Microsoft, and an Intel research award.

References

[1] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123

(1–3):155 – 225, 2002. ISSN 0166-218X. doi: 10.1016/S0166-218X(01)00341-9.

[2] N. Brenner, S.P. Strong, R. Koberle, W. Bialek, and R.R.R. Steveninck. Synergy in a neural code. *Neural Computation*, 12(7):1531–1552, 2000.

[3] K. Byrnes. Maximizing general set functions by submodular decomposition. *Arxiv preprint arXiv:0906.0120*, 2009.

[4] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] W. H. Cunningham. Decomposition of submodular functions. *Combinatorica*, 3(1):53–68, 1983.

[6] J. Edmonds. Submodular functions, matroids and certain polyhedra. *Combinatorial structures and their Applications*, 1970.

[7] M. Feldman, J. Naor, and R. Schwartz. A tight (1/2) linear-time approximation to unconstrained submodular maximization. *To appear, FOCS*, 2012.

[8] Uriel Fiege, Vahab Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM J. COMPUT.*, 40(4):1133–1155, July 2007.

[9] A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml.

[10] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier Science, 2005.

[11] S. Fujishige and S. Isotani. A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, 7: 3–17, 2011.

[12] W. Iba, J. Wogulis, and P. Langley. Trading off simplicity and coverage in incremental concept learning. In *Proceedings of Fifth International Conference on Machine Learning*, pages 73–79, 1988.

[13] R. Iyer and J. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *Extended version*, 2012.

[14] S. Jegelka and J. Bilmes. Cooperative cuts: Graph cuts with submodular edge weights. Technical report, Technical Report TR-189, Max Planck Institute for Biological Cybernetics, 2010.

[15] S. Jegelka and J. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, June 2011.

[16] S. Jegelka and J. Bilmes. Online algorithms for submodular minimization with combinatorial constraints. *In Proc. ICML*, 2011.

[17] Stefanie Jegelka and Jeff A. Bilmes. Approximation bounds for inference using cooperative cuts. In *ICML*, Bellevue, Washington, 2011.

[18] Stefanie Jegelka and Jeff A. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, June 2011.

[19] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. 9th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2003.

[20] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the second international conference on knowledge discovery and data mining*, volume 7, 1996.

[21] V. Kolmogorov and C. Rother. Minimizing non-submodular functions with graph cuts–a review. *IEEE TPAMI*, 29(7):1274–1279, 2007.

[22] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE TPAMI*, 26(2):147–159, 2004.

[23] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of Uncertainity in Artificial Intelligence*. UAI, 2005.

[24] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.

[25] Andreas Krause, Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *Journal of Machine Learning Research (JMLR)*, 9:2761–2801, December 2008.

[26] J. Lee, V.S. Mirrokni, V. Nagarajan, and M. Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 323–332. ACM, 2009.

[27] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. *NAACL*, 2010.

[28] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. *In Proc. ACL*, 2011.

[29] Hui Lin and Jeff A. Bilmes. Optimal selection of limited vocabulary speech corpora. In *Proc.*

Annual Conference of the International Speech Communication Association (INTERSPEECH), Florence, Italy, August 2011.

[30] Mukund Narasimhan and Jeff Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Uncertainty in Artificial Intelligence (UAI)*, Edinburgh, Scotland, July 2005. Morgan Kaufmann Publishers.

[31] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[32] J.B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

[33] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2nd printing edition, 1988.

[34] A.A. Schäffer. Simple local search problems that are hard to solve. *SIAM journal on Computing*, 20:56, 1991.

[35] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[36] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 697–706. IEEE, 2008.

[37] L. Trevisan. Inapproximability of combinatorial optimization problems. *The Computing Research Repository*, 2004.

[38] M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[39] A.L. Yuille and A. Rangarajan. The concave-convex procedure (cccp). *Advances in Neural Information Processing Systems*, 2:1033–1040, 2002.

Incentive Decision Processes

Sashank J. Reddi
Machine Learning Department
Carnegie Mellon University
sjakkamr@cs.cmu.edu

Emma Brunskill
Computer Science Department
Carnegie Mellon University
ebrun@cs.cmu.edu

Abstract

We consider Incentive Decision Processes, where a principal seeks to reduce its costs due to another agent's behavior, by offering incentives to the agent for alternate behavior. We focus on the case where a principal interacts with a greedy agent whose preferences are hidden and static. Though IDPs can be directly modeled as partially observable Markov decision processes (POMDP), we show that it is possible to directly reduce or approximate the IDP as a polynomially-sized MDP: when this representation is approximate, we prove the resulting policy is boundedly-optimal for the original IDP. Our empirical simulations demonstrate the performance benefit of our algorithms over simpler approaches, and also demonstrate that our approximate representation results in a significantly faster algorithm whose performance is extremely close to the optimal policy for the original IDP.

1 Introduction

Consider a landlord who pays for her tenant's heating bill, or an insurance company that covers an insuree's health care bills. These are two instances where a principal agent incurs a cost for the behavior of another agent. In such scenarios the principal may be able to provide incentives to attempt to change the agent's behavior; however, the principal typically does not know the preferences of the agent, and these preferences determine which incentives will be accepted by the agent in return for altered behavior.

We formalize this problem as an instance of sequential decision making under uncertainty. Despite much AI interest in this general field, there has been limited research on multi-step decision processes where a principal agent's own costs are a function of another agent's behavior. We call these processes Incentive Decision Processes (IDPs) and in this paper we focus on the case where a principal interacts with a greedy myopic agent whose preferences are hidden and static. The objective is to compute a decision policy for the principal that minimizes its total expected sum of costs over the horizon of interactions with the agent. Computing this policy would be trivial if the agent's preferences were known to the principal: the principal would simply offer the incentive for an agent action that minimized the principal's cost amongst all incentive-action pairs that would be accepted by the agent. However, the agent preferences are typically hidden. It is important to consider the case where a principal cannot simply ask the agent for its preferences. Naturally a strategic rational agent may wish to misrepresent its stated preferences in order to gain additional reward, and we will briefly touch on this issue at the end of the paper. But our focus will be on agents that act myopically and therefore truthfully (as we will later prove). We argue that this model is reasonable for many important situations where the agent is not adversarial, but is either unaware or in denial of its own utility function, such as asking a person about their preference for consuming healthy food or exercising different amounts, or where it is impossible for the agent to directly respond (such as teaching a baby a new behavior).

Our work is most closely related to the research of Zhang and colleagues [12, 13, 4] on environment design. However, Zhang et al.'s work primarily addresses inducing the agent to follow a particular policy within a short number of interactions, whereas our research focuses on how to provide incentives to minimize the total cost incurred by the principal over many interactions with the agent. We believe our objective is likely to be particularly relevant in multiple real-world domains, where the cost of any diagnostic steps to reveal the hidden preferences of the agent, such as a tenant, office employee, or insuree, must be balanced with the expected benefit of that information in reducing the cost to the principal. In this sense our algorithms will

address the exploration versus exploitation tradeoff in the context of Incentive Decision Processes.

We will show constructing a policy for the principal in an IDP may be modeled as a partially observable Markov decision process (POMDP) planning problem. However we will demonstrate that we can leverage the significant structure in IDPs to achieve much more efficient optimal or boundedly-optimal algorithms. Our first contribution is to prove that if there are only two different actions that the agent may choose, then the incentive decision process can be reduced to a polynomially sized MDP. We will also describe a similar approximate mapping for multiple action incentivized decision processes, and also present formal bounds on the quality of policies for the approximate representation. Our empirical simulations demonstrate the performance benefit of our sequential decision-theoretic algorithms over simpler approaches. We will finish by briefly discussing multiple extensions to our model, including dynamic agent preferences and strategically rational agents. When proofs are omitted, they are provided in the appendix.

2 Background

A Markov decision process (MDP) is described by a tuple $\langle S, A, p(s'|s,a), C, \gamma \rangle$. S is a set of states, A is a set of actions, and $p(s'|s,a)$ is the transition model and stores for each state s and action a the probability of transitioning to any state s'. C is the cost model and represents the cost of taking action a in state s and transitioning to state s'. $\gamma \in [0,1]$ is a discount factor. A stationary policy $\pi : S \rightarrow A$ is a mapping from states to actions.[1] The value of a policy π for a horizon H from state s is the expected sum of costs over H steps when following the policy from state s,

$$V_\pi^H(s) = \sum_{s'} p(s'|s,\pi(s))[c(s,\pi(s),s') + \gamma V_\pi^{H-1}(s')]$$

This equation is also known as the Bellman equation [1]. In a slightly modified form (known as a Bellman update) it can be used to iteratively compute the optimal value function and optimal policy. If the horizon is infinite, then the value function is independent of the time step. The optimal policy is the policy that has the lowest value.

Partially observable MDPs (POMDPs) [10, 6] are often used to describe decision processes where the state is not directly observable. POMDPs are represented by a tuple $\langle S, A, O, p(s'|s,a), p(o|s',a), C, \gamma \rangle$. Here O is a set of observations, and $p(o|s',a)$ is the probability of observing o after taking action a and transitioning to state s'. A belief b is a probability distribution over the possible world states, given the prior history of actions taken and observations received. b is a sufficient statistic for the history, and can be updated as new actions are taken, and new observations are received, using a Bayes filter. In POMDP planning the objective is to compute a policy $\pi : b \rightarrow a$ that maximizes the expected sum of future rewards.

[1]The policy can also depend on the time step.

3 Related Work

Since the agent's preferences are unknown to the principal, our work is loosely related to research on preference elicitation [3]. However, preference elicitation only focuses on identifying an agent's hidden reward model, or identifying it sufficiently to make a single decision. In contrast, in our work the principal seeks to minimize its expected sum of costs. Similarly, inverse reinforcement learning [9] seeks to identify an agent's hidden reward model given the agent's behavior, but our focus is on minimizing the cost to a principal by altering the agent's behavior through the offering of incentives, which may not involve fully identifying the agent's hidden reward model.

Research on multiple agents interacting includes sophisticated models for representing and updating each agent's internal estimate of the state of the other agent. Such representations can lead to infinite nestings of state estimates [11]. Interactive POMDPs (I-POMDPs) are used to represent such problems, and use a finite representation of each agent's state for tractability [5]. Such rich representations remain very expressive, but at the price of computational intractability for all but very small domains.

The closest work to our own is that of Zhang and colleagues [12, 14, 13, 4] on policy teaching and environment design. Their earliest work [12] focuses on finding a set of incentives for the agent that minimizes the principal's expected cost. Their setting is quite general: the agent acts in a MDP and the principal provides a set of incentives over the MDP states. Zhang et al. provide an algorithm for identifying the optimal incentives to provide within a finite (but unknown) number of episodes, where in each multi-step episode the agent follows their optimal policy given their (hidden) reward function plus the provided incentives. Our presented work focuses on a more restrictive setting of a greedy agent selecting among a fixed set of actions at each step, but within this context we provide formal guarantees of our algorithm's performance over a *single* multi-step episode as the principal and agent interact. Our IDP model is most similar to Zhang et al.'s recent work [4] on an incentivized multi-armed bandit. However, this work focuses on inducing a particular action selection by the agent given some budget, whereas our focus is on minimizing the total expected cost for

the principal. Also, we are interested in long horizon decision processes, versus their algorithmic work only considers short (≤ 3 step) horizons.

4 Incentive Decision Process Model

Consider a principal interacting multiple times with an agent. At each time step the agent chooses an action from a set of agent actions $A = \{a_1, a_2, \cdots, a_N, a_{N+1}\}$. We assume that the agent has an internal reward R_{a_n} for every action a_n such that $R_{a_i} > R_{a_j}$ if $i > j$. Therefore a_{N+1} will yield the highest reward for the agent, and so we define a_{N+1} as the default agent action that the agent will select if no incentives are provided. All other actions $a_1, \cdots, a_N$ will be referred to as alternate agent actions. The agent's rewards are hidden from the principal, but due to the assumed structure, the principal does know that action a_{N+1} is the agent's default action. Each agent action a_i is also associated with a particular cost c_i to the principal and the costs are such that $c_i < c_j$ if $i < j$. Therefore by definition the default agent action a_{N+1} is of highest cost to the principal.

At each time step the principal offers an incentive δ for a particular action a_n ($n \in (1, N)$) to the agent if the agents chooses action a_n on this time step. The offered incentive is selected from a set of K incentives $\Delta = \{\delta_1, \cdots, \delta_K\}$ where $\delta_i < \delta_j$ if $i < j$. We assume that the agent acts to maximize its immediate reward, and will accept the offered incentive and take action a_n if doing so yields a higher reward than the default action, $R_{a_n} + \delta > R_{a_{N+1}}$. Otherwise the agent will reject the incentive and take the default action a_{N+1} and receive reward $R_{a_{N+1}}$. If the agent accepts the offered incentive for action a_n, the principal will incur an immediate cost of $c_n + \delta$; otherwise, the principal's immediate cost is c_{N+1}. We will assume that the cost of each alternate agent actions, plus the maximum possible incentive, is always less than the cost of the default agent action: $c_N + \delta_K < c_{N+1}$. This implies that the principal would always prefer the agent to accept an incentive and take one of the alternate agent actions. We also assume that for each alternate agent action a_n, there exists an incentive in Δ such that if that incentive is offered, the agent would prefer to accept that incentive and take the alternate agent action instead of the default action. Let $t_n = R_{a_{N+1}} - R_{a_n} \in \Delta$ be the least incentive for which the agent prefers action a_n to the default agent action a_{N+1}. Observe that lower indexed actions have lower rewards to the agent, and therefore will require incentives equal or greater than higher indexed rewards, $t_i \geq t_j$ for $i < j$. We use $I = (t_1, \cdots, t_N)$ to denote the tuple of true incentives for the N alternate agent actions. These incentives are initially unknown to the principal. The principal is provided with an initial joint probability distribution P_0 which gives the probability that $t_i = \delta_{\mathbf{k}}$ for all $n \in \{1, \cdots, N\}$ and $k \in \{1, \cdots, K\}$.

Note the set of alternate actions and set of possible internal rewards for each action for the agent is common knowledge. The agent's actual reward for each alternate action is prive knowledge to the agent. Since the agent is assumed to act in a myopic fashion, it does not matter if the principal's costs for each alternate action, and its set of possible incentives, are private or common knowledge.

The objective of the principal is to minimize its expected sum of costs over H interactions with the agent. The costs may be multiplied by a discount factor $\gamma \in [0, 1]$. We will consider both the finite horizon case and the discounted infinite horizon case ($H = \infty$). In this paper we will design tractable algorithms that compute a decision policy to achieve the principal's objective.

For example, the landlord-tenant scenario can be modeled as an IDP where the tenant's (agent's) default temperature setting is the default action, and other temperatures correspond to alternate agent actions. Each temperature maps to a cost to the landlord (principal). At each step the principal offers an incentive to the agent if the agent sets the thermostat to an alternate temperature that the principal specifies.

An IDP can also be modeled as a POMDP. In the IDP POMDP the state is the agent's hidden t_n for all $n \in \{1, \cdots, N\}$, the actions are the cross product of all alternate agent actions with the possible incentives Δ (since the principal can offer any incentive paired with any alternate agent action), and the observations are the binary accept or reject response of the agent. For most of this paper we assume that the agent's hidden preferences are static, and so the state transition model is a delta function. The observation model is that an agent's acceptance of an offer of δ_k for alternate action a_n indicates that $t_n \leq \delta_k$ (since an agent will accept any incentive equal or greater than t_n for alternate action a_n), and a reject indicates that $t_n > \delta_k$. Note that the observation model depends on how we assume the agent acts, which is a function of its internal hidden reward and the offered incentive. Since we assumed that the agent is myopically greedy, the agent will accept an incentive if the sum of offered incentive and the agent's internal reward is greater than the agent's hidden reward for its default action choice. The initial belief in the POMDP is given by P_0. Computing the optimal policy for this POMDP will enable the principal to optimally offer (a_n, δ_k) pairs that balance reducing the uncertainty over the agent's hidden true incentives I with minimizing the expected cost to the principal, given the current belief state over I.

However, computing an optimal policy for a POMDP can take doubly exponential time and existing methods often struggle to scale to large state or action spaces or long time horizons, all of which can feature in IDPs.

Fortunately there is significant structure in IDPs. We will now show how we can leverage this structure to reduce the problem to simpler decision processes, and we will present efficient algorithms for computing exact or boundedly-optimal policies.

5 Single Alternate Agent Action IDPs

We first consider IDPs when there is a single alternate action a_1. It is known that a POMDP can be reduced to a continuous state MDP where each state represents a belief: note that since the states are probability distributions, the MDP has an infinite state space. Here we show the surprising result that a single alternate agent action IDP can be reduced to a polynomially sized MDP, which means they can be solved much more efficiently than standard POMDPs.

We first describe how to perform belief updating in a single alternate agent action IDP. Let the possible range of incentives for alternate agent action a_1 be $S_1 = [s_1, e_1]$ if and only if $\delta_{s_1} \leq t_1 \leq \delta_{e_1}$. $S = [S_1, \cdots, S_N]$ are the incentive ranges for all alternate agent actions; for $N = 1$, $S = S_1$. We are interested in maintaining the probability that $t_1 = \delta_i$ for all $\delta_i \in \Delta$. Given the initial probability distribution P_0, we can directly compute the initial incentive range S_1. The initial probability that $t_1 = \delta_i$ is

$$p_{1,S}(\delta_i) = \frac{P_0(\delta_i)\mathbb{1}(\delta_i, S)}{\sum_{k=1}^{K} P_0(\delta_k)\mathbb{1}(\delta_k, S)}, \quad (1)$$

where $\mathbb{1}(\delta_i, S) = 1$ if $\delta_{s_1} \leq \delta_i \leq \delta_{e_1}$ (if incentive δ_i is within the possible incentive range), and, with a slight abuse of notation, we use $P_0(\delta_i)$ to represent the initial probability that the agent's $t_1 = \delta_i$. Note that initially the denominator will be 1, because the range will cover all incentives δ_i for which $P_0(\delta_i) > 0$. On the first time step the principal will offer the agent an incentive δ_i to take the alternate agent action a_1. If the agent accepts, the principal knows that $t_1 \leq \delta_i$, due to the assumed monotonic structure of the incentives and the greedy myopic agent behavior. Therefore the new possible incentive range if the agent accepts δ_i is $S_1 = [s_1, i]$, since the upper bound on the possible index is at most i. If the agent rejects the offered incentive, then $t_1 > \delta_i$ and so the new incentive range must be $S_1 = [i{+}1, e_1]$. In either case the new probability of the agent accepting a particular incentive δ_j is identical to Equation 1 using the updated incentive range S_1.

Each offered incentive, and associated agent response, can set the probability of some incentives to zero. Since t_1 is static, we prove it is sufficient to maintain and update S_1 in order to compute the current $p_{1,S}(\delta_j)$ given the history of offered incentives and agent responses. Therefore, the belief state over the agent's incentive t_1 will be of the form $B^1_{i,j} = (0, \cdots, 0, p^1_{i,S}, \cdots, p^1_{j,S}, 0, \cdots, 0)$ where $S_1 = (i, j)$ and $p^1_{j,S} = p_{1,S}(\delta_j)$. We will shortly prove that there are only $O(K^2)$ possible such states.

Before we do so, we first note that an IDP-MDP is an instance of a deterministic POMDP, since the observations are a deterministic function of the underlying agent's preferences, and the transition model is a delta function because these preferences are assumed to be static. Prior work [7, 2] has proved that a deterministic POMDP with N_s states can be converted to a finite-state MDP with a number of states that is an *exponential function* $((1 + N_s)^{N_s})$ of the number of POMDP states. This is significantly smaller MDP than generic POMDPs which map to a MDP with an infinite number of states. However, we will show that an IDP with $N = 1$ can be mapped to a MDP with $B^1_{i,j}$ as the MDP states. We now prove that the resulting MDP has a number of states that is only a *polynomial* function of the number of POMDP states (K).

Theorem 5.1. *An IDP with $N = 1$ alternate agent actions is equivalent to an MDP with $O(K^2)$ states.*

Proof. We see that $B^1_{1,K}$ is the initial state of the IDP-MDP. Suppose $B^1_{i,j}$ is the current state. Consider the case where incentive δ_k is offered. The agent might either accept or reject the incentive. As stated, the agent's response to each offer can eliminate certain range of the incentives but the ratio amongst the remaining ones does not change. If the agent accepts the incentive, the new probability vector will be $B^1_{i,k}$ since the response indicates that $t_1 \in [\delta_i, \delta_k]$, otherwise if the agent rejects the incentive, the new probability vector will be $B^1_{k+1,j}$, since $t_1 \in [\delta_{k+1}, \delta_j]$. Hence, by induction, at the t^{th} step, the probability vector is of the form $B^1_{i,j}$ and only states of this form are reachable from the initial probability vector. Now i and j can each take on at most K values, so the maximum number of possible $B^1_{i,j}$ is at most $O(K^2)$. Therefore the number of belief states is bounded by $O(K^2)$ and therefore the IDP with one alternate agent action can be reduced to a MDP with $O(K^2)$ states. □

We now define the IDP-MDP for a single alternate agent action. We use $v(l)$ to denote the l-th element of a vector v. The state space is the $B^1_{i,j}$ vectors just described. The action space is the incentive space Δ. The MDP transition model is the probability of transitioning from one $B^1_{i,j}$ to other belief states after the principal offers the agent an incentive. For a given

Algorithm 1 ValH1

Input: IDP-MDP M, state $B^1_{i,j}$, matrix V of horizon-values, matrix π of horizon-policy, h
if $h = 0$ **then**
  $V^h(B^1_{i,j}) = 0$ {no more time steps}
else
  for $k = i : j-1$ **do**
    if $V^{h-1}(B^1_{i,k}) = \emptyset$ **then**
      $[V^{h-1}(B^1_{i,k}), V, \pi]$=ValH1($M$,$B^1_{i,k}$,$V$,$\pi$,$h$–1) {Calc. value if agent accepts δ_k}
    end if
    if $V^{h-1}(B^1_{k+1,j}) = \emptyset$ **then**
      $[V^{h-1}(B^1_{k+1,j}), V, \pi]$=ValH1($M$,$B^1_{k+1,j}$,$V$,$\pi$,$h$–1) {Calc. value if agent rejects δ_k}
    end if
    $Q^h(B^1_{i,j}, \delta_k) = p(B^1_{i,k}|B^1_{i,j}, \delta_k)(\delta_k + V^{h-1}(B^1_{i,k})) + p(B^1_{k+1,j}|B^1_{i,j}, \delta_k)(c_2 + V^{h-1}(B^1_{k+1,j}))$
  end for
  $Q^h(B^1_{i,j}, \delta_j) = \delta_j h$
  $V^h(B^1_{i,j}) = \min Q^h(B^1_{i,j}, \cdot)$
  $\pi^h(B^1_{ij}) = \arg\min Q^h(B^1_{i,j}, \cdot)$
end if
Return $[V^h(B^1_{i,j}), V, \pi]$

state $B^1_{i,j}$, and a given offered incentive δ_k, with probability $\sum_{i \le l \le k} B^1_{i,j}(l)$ the agent will accept the offer and the new state will be $B^1_{i,k}$, and with probability $1 - \sum_{i \le l \le k} B^1_{i,j}(l)$ the agent will reject the offer, and the new state will be $B^1_{k+1,j}$. The cost model of the MDP takes the form $c(B^1_{i,j}, \delta_k, B^1_{i,k}) = \delta_k + c_1$ for transitions where the agent accepts the incentive, and $c(B^1_{i,j}, \delta_k, B^1_{k+1,j}) = c_2$ when the agent rejects.

We now show the special structure of this problem allows us to introduce more efficient planning algorithms that require only $O(K^3)$ and $O(min(H, K)\, K^3)$ computation time for the infinite and finite horizon case, respectively. First note that the principal only needs to consider offering incentives δ_k within the range of $i \le k \le j$ for a state $B^1_{i,j}$, as other incentives will be automatically rejected (or accepted but at higher cost than necessary for the principal). We start by considering the infinite horizon case. Let the expected sum of discounted costs of state $B^1_{i,j}$ be $V(B^1_{i,j})$. We define a MDP state s to be non-recurring with respect to a policy π if the state can only be reached once while executing the π. A state s of the MDP is called self-absorbing if executing π from s results in a self-transition to state s with probability 1. A policy π is NonRecur-Absorb if each MDP state is either non-recurring or self-absorbing with respect to π.

First recall that for an infinite horizon MDP, there always exists an optimal policy that is stationary (the decision depends on only the current state and is independent of the time step). Therefore we will consider only stationary policies. Next note that for an infinite horizon IDP-MDP, if incentive δ_j is offered in state $B^1_{i,j}$, then the MDP remains in same state $B^1_{i,j}$. If one of the other possible actions (offering incentives $\delta_i, \cdots, \delta_{j-1}$) is taken, then the MDP will transition to a new state, and will never return to the state $B^1_{i,j}$ since the number of non-zero entries in the probability vector $B^1_{i,j}$ monotonically decreases. This implies all infinite-horizon policies for an IDP-MDP are NonRecur-Absorb. As each state can only be reached once (or is an absorbing state) during execution, we only need to compute a Bellman backup for the state-action values Q once for each state-action pair. We compute a decision policy by computing the state-action values of the initial state by recursively computing the value of each reachable state. The values of subsequent states are stored and cached so that they can be re-used if the same state is reached through a different trajectory of offered incentives and responses. There are $O(K^2)$ MDP states and $O(K)$ actions for each state, so there are $O(K^3)$ state-action pairs. The cost of computing the Bellman backup for a given state-action pair can be done in constant time given the values of the possible next states (since there are only at most 2 possible next states). Therefore the algorithm takes $O(K^3)$ time. Note this is significantly faster than value iteration on generic MDPs with N_s states and N_a actions which requires $O(\frac{N_a N_s^2}{1-\gamma})$ time to compute a near-optimal policy.

For the finite horizon case, let $V^h(B^1_{i,j})$ denote the expected sum of costs to the principal for state $B^1_{i,j}$ for a horizon of h future interactions with the agent. Every policy is not NonRecur-Absorb because the principal could offer the same incentive for multiple time steps (staying in the same state) and then offer a new incentive and transition to another state. However, we can prove the following result:

Theorem 5.2. *There exists an optimal policy for an* IDP-MDP *with* $N = 1$ *that is* NonRecur-Absorb.

The proof intuition is that we can always re-order an optimal policy so that it only offers incentives that keep the MDP in the same state at the final time steps.

The algorithm for the finite horizon case is displayed in Algorithm 1. Here the state's value and policy will depend on the time step. Each time the horizon decreases, either a state becomes self-absorbing with a cost of $\delta_j h$ or least one non-zero element of the probability vector $B^1_{i,j}$ becomes zero. Since the initial state $B^1_{1,K}$ has K non-zero elements, it will take at most K time steps until a state is reached

which is self-absorbing (as all single-element beliefs are self-absorbing). Therefore, the algorithm takes $O(min(H, K)\, K^3)$ running time

6 Multiple Alternate Action IDPs

We next consider IDPs with multiple alternate agent actions, namely $N > 1$. We now need to maintain a belief state over the joint of the alternate agent actions crossed with the incentives. As stated in the definition of the IDP, we assume we are provided with an initial probability distribution over the true incentives t_n for each alternate action a_n, P_0. We know from the prior section that all beliefs for an $N = 1$ IDP are of the form $B^1_{i,j} = (0, \cdots, 0, p^1_{i,S}, \cdots, p^1_{j,S}, 0, \cdots, 0)$ where $S_1 = (i, j)$. We now extend the model to handle multiple actions. Let $B_S = (B^n_S)\,|^N_{n=1}$ where B^n_S denotes the probability vector $(0, \cdots, 0, p^n_{i,S}, \cdots, p^n_{j,S}, 0, \cdots, 0)$ where $S_n = (i, j)$ and $p^n_{i,S}$ is the probability that $t_n = \delta_i$ given the initial joint distribution P_0 and the possible incentive ranges S_n for each alternate agent action. $S_n = (s_n, e_n)$, for all $n \in \{1, \cdots, N\}$.

Theorem 6.1. *The number of states of* IDP-MDP *for $N > 1$ is $O(K^{2N})$ states.*

Proof. Assume that the current state is B_S, where $S_n = (s_n, e_n)$ for $1 \le n \le N$. Consider the case where incentive δ_i is offered for action a_n. The agent will accept this offer with the probability that its true incentive t_n for a_n is $\le \delta_i$, which is equal to $\sum_{j \le i} p^n_{j,S}$. If the offer is accepted, then $s_n \le t_n \le i$, which implies $S_n = (s_n, i)$. In addition, all alternate agent actions $m > n$ with higher agent reward than alternate agent action a_n will also all accept t_n, so the incentive range of all such actions will also be updated $S_m = (s_m, i)$ for all $m > n$. This defines a new S, which together with P_0, completely defines the new distribution over the probability of each incentive for each alternate agent action. If the agent rejects the offered incentive, then $s_n \ge i + 1$, and so $S_n = (i + 1, e_n)$. All alternate agent actions $m < n$ with a lower agent reward than action a_n will also reject the reward, which results in $S_m = (i + 1, e_m)$ for all $m < n$. There are N alternate agent actions, and we know from Theorem 5.1 that there are at $O(K^2)$ states per individual action, therefore there are at most $O(K^{2N})$ possible states. Due to the relationship among the agent costs and rewards, this number will often be much lower. □

An IDP with $N > 1$ alternate agent actions has many of the same properties as an IDP with $N = 1$ actions, and therefore we can use algorithms similar to those described in the prior section (see the text and Algorithm 1) to solve the IDP-MDP with $N > 1$. Excluding the cost for the belief updates, this requires $O(NK^{2N+1})$ and $O(min(H, NK)K^{2N+1})$ computation time for the infinite and finite horizon case, respectively. While this is computationally tractable for small N, this scales poorly as N increases.

To address this, we will now provide efficient approximate algorithms for computing a decision policy in IDPs with large N. Note that computing the probability of the next possible states, given an offered incentive δ_k for alternate agent action a_n involves computing the marginal probabilities $p^n_{k,S}$ that $t_n = \delta_k$ given the current S. This is an $O(K^N)$ operation due to summing over all other action-incentive probabilities. Hence, any approximation algorithm will take at least $O(K^N)$ time due to the bottleneck of belief updating. This cost can be reduced by assuming structure in the joint probability distribution, such as using graphical models with bounded treewidth. We leave further exploration of this issue for future work.

We now define a new special form of a Markov decision process. Let a SEQ-MDP be an IDP-MDP with $N > 1$ with the following additional restrictions on the allowable principal's actions at different states:

1. For the initial state, the principal can offer any incentive for only alternate agent action a_1 (assuming that initially $S_1 = (s_1, e_1)$ and $s_1 \ne e_1$).

2. For any other state, the principal can offer any incentive for alternate agent action a_n, only when $S_i = (s_i, e_i)$ and $e_i = s_i$ for all $i < n$. In other words, t_i must be known for all alternate agent actions $i < n$ before the principal can offer an incentive for a_n.

A SEQ-MDP restricts the possible decision policy set Π to policies that only provide incentives for alternate agent actions with higher cost to the principal only after finding the true incentive for the lower cost actions. Note that the SEQ-MDP policy may not identify the true incentive for all actions, but can decide to stick with a previously identified a_n, δ_k pair.

We now analyze the number of states in a SEQ-MDP. Consider that at the present state the principal is offering incentives for alternate agent action a_n. We know that $S_i = (s_i, e_i)$ for $i > n$ and $s_i = e_i$ for $i < n$. Define $c_i + \delta_j$ as the minimum action cost + true incentive across all $i < n$. Since we would never take any other $i' < n$ with a higher cost, $c_i + \delta_j$ is sufficient to summarize all useful information from prior offered actions. Therefore we can represent a state in the SEQ-MDP as the tuple (a_i, δ_j, B_{S_n}). For each offered alternate agent action, there can be $O(K^2)$ states, as in the IDP-MDP with $N = 1$. We also must track the current minimal cost alternate agent action and offered incentives, and there are $O(NK)$ such combinations. Finally we also have to monitor

the current alternate agent action, and there are $O(N)$ possibilities. The product of these quantities yields a state space of $O(N^2K^3)$. For similar arguments as given in Section 5, we can construct optimal policies that are NonRecur-Absorb for SEQ-MDP and use planning methods similar to those described in Section 5. There are $O(K)$ possible actions for each state in a SEQ-MDP. Therefore, again excluding costs for computing the marginal probabilities, the computational cost of computing an optimal infinite-horizon policy for a SEQ-MDP is $O(N^2K^4)$, and the finite horizon H cost is $O(\min(H, NK)N^2K^4)$.

We now prove that computing the optimal policy for a SEQ-MDP yields a boundedly-optimal policy for the original IDP-MDP with $N > 1$.

Theorem 6.2. *An optimal policy for a* SEQ-MDP *that is a transformation of an* IDP-MDP *with $N > 1$, has an expected cost V^{seq} which is bounded by $V^{seq} = V^* + \sum_{k=1}^{K}(\delta_k - \delta_1) + N(c_{N+1} - c_1)$, where V^* is the optimal value for the* IDP-MDP.

Proof. We will make a constructive argument by providing a policy $\pi \in \Pi$ that realizes the bound V^{seq}. Consider a policy π that starts with the lowest cost action, and offers the highest possible incentive. The policy proceeds by sequentially decreasing the offered incentive δ_k for alternate agent action a_1 until the agent rejects the offered incentive. Let this highest incentive at which the agent rejects be δ_{j-1}. That means that the true incentive t_1 for alternate agent action a_1 is δ_j. Since a_1 is the lowest cost and lowest reward alternate action, we know that δ_j will be accepted for all other alternate agent actions $i > 1$, since all such actions yield higher reward to the agent. Therefore we need not offer δ_j for any other agent actions. The policy then moves on to offering incentives for alternate agent action a_2, starting with offering δ_{j-1}. The policy again continues to incrementally decrease the offered incentive until it reaches a reject, at which it then starts to provide incentives for a_3, and so on. The policy is similarly defined for all subsequent alternate agent actions. The defined policy π operates only on the SEQ-MDP version of the IDP-MDP.

Note that after the principal makes an offer, either an incentive for a particular action, or a particular incentive for all actions can be eliminated. Since there can at most be K accepts, and at most N rejects (as at most one offer can be rejected for each action), there are at most $K + N$ steps.

Each of the at most N rejected offers will cause the algorithm to incur at most $c_{N+1} - c_1$ additional cost compared to the optimal policy, since the principal will pay the cost of the default action c_{N+1}. Each offer that was accepted that is above the true incentive will result in an additional cost of at most $\delta_i - \delta_1$. Note that each incentive δ_i is offered only once if it is accepted for an action a_n. Therefore, the total cost is bounded by $V^{seq} = V^* + N(c_{N+1} - c_1) + \sum_{k=1}^{K}(\delta_k - \delta_1)$. This is true irrespective of the distribution over incentives. It can be easily seen that the policy $\pi \in \Pi$, as required. □

This bound does not use any information about the probability distribution. A tighter bound using the probability distribution can be obtained but is not presented here for ease of exposition. Also, an alternate algorithm with bound $V^* + K(c_{N+1} - c_1) + \sum_{i=1}^{N}(c_N - c_i)$ is possible by starting from action a_N rather than a_1 and progressing to lower cost actions only after identifying the true incentive for the higher cost actions. An argument similar to the one made in Theorem 6.2 can be used to prove this bound. In next section, our empirical results show the optimal policy for a SEQ-MDP performs well in practice.

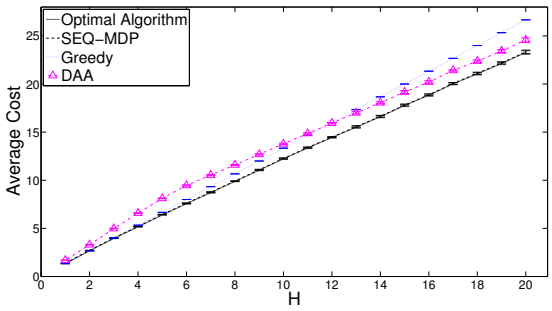
7 Experiments

We now empirically evaluate our algorithms.

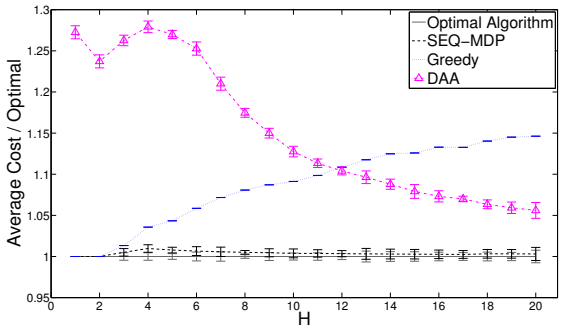
In our simulations the cost of the default agent action was set to $c_{N+1} = 2$. For N alternate agent actions, the cost to the principal of the agent taking action a_n was set to $c_n = \left(\frac{n}{N}\right)^{\eta}$ where η is a constant. For K incentives, the value of the k-th incentive was $\delta_k = k/K$. Note the maximum cost of the alternate agent actions is 1, and the maximum incentive is 1. Therefore the cost to the principal of an agent accepting an incentive $c_n + \delta_k$ is always less than or equal than the cost to the principal of the agent's default action c_{N+1}, as our IDP model assumed (Section 4). The initial belief was set to a uniform joint distribution over the possible incentive-alternate agent action space: recall that this distribution must always respect the constraints that $t_i \geq t_j$ for $i < j$ since $R_{a_i} < R_{a_j}$. In each experimental run we fixed K,N,η and the horizon H, and sampled a true hidden vector of incentives for the agent from the initial belief. We then executed the policy of each algorithm and recorded the total cost accumulated over H steps. We simulated 1000 runs and averaged over each run's total sum of costs. We repeated this for 10 rounds (each of 1000 runs) and computed the standard deviation error bars for the average total cost.

We compare the performance of our algorithms to two natural methods. The first is a greedy algorithm. In the greedy algorithm, given a current state B_S (with $S_n = (s_n, e_n)$), the principal offers the incentive δ_k for alternate action a_n that minimizes the expected immediate cost to the principal,

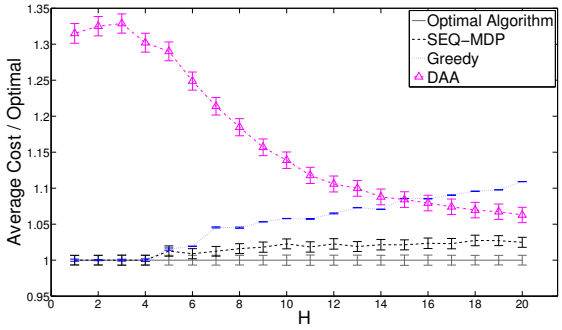
$$[\delta_k, a_n] = \underset{\delta_k, a_n}{\operatorname{argmin}} \sum_{k'=s_n}^{k} p^n_{k',S}(\delta_k + c_n) + \sum_{k'=k+1}^{e_n} p^n_{k',S}(c_{N+1}).$$



(a) Avg total cost over H (w/1 std. error bars).$N = 3, K = 5$.



(b) Ratio of avg. total cost to optimal. $N = 3, K = 5$.

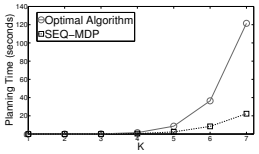


(c) Ratio of costs. $N = 5, K = 3$.

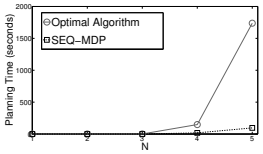
Figure 1: Average total cost for varying horizons H.

The second method first uses a binary-search-like procedure, starting from a_1, to identify the true incentive for each alternate agent action. It then selects the alternate agent action, and incentive, with the lowest immediate cost, and then offers that for all remaining steps. We label this algorithm "diagnose-and-act (DAA)." It should be noted that DAA can be viewed as a first running a simple inverse reinforcement learning approach.

Figure 1(a) displays the average total cost incurred for different horizons for $N = 3$ alternate agent actions, and $K = 5$ incentives. Here we set $\eta = 1$ and $N \leq 5$ but results from varying η (from 0.75 to 1.25) and N yielded similar results. As expected, the optimal policy performs best for all horizons. To better explore the difference between the algorithms, we replotted the results by dividing by the optimal algorithm's average total cost at each horizon H, resulting in a figure where the optimal algorithm is always 1, and all other algorithms are displayed as their ratio to the optimal performance (see Figure 1(b)). For low horizons the greedy algorithm performs very close to optimal, because there is little benefit to gathering information to find a lower cost alternate since there is little time to use such information. However, as H increases, greedy performs significantly worse than the optimal algorithm. DAA algorithm performs poorly at low horizons because it tries to identify the optimal incentive for each action, and this can take longer than the available horizon to accomplish. At larger horizons DAA performs better. For very large horizons we expect that DAA would converge to the performance of the optimal policy: in such cases the loss incurred from not identifying the best action-incentive pair will likely become very high, meaning that it will be optimal to identify the minimal cost incentive.



(a) $N = 3$, $H = 20$



(b) $K = 4$, $H = 20$

Figure 2: Planning time (sec) as a function of K or N for the optimal algorithm and SEQ-MDP.

The SEQ-MDP approach has very good performance. Indeed it is was visually and statistically indistinguishable from the optimal algorithm for all horizons H for $K = 5, N = 3$. SEQ-MDP is not always identical to the optimal algorithm's performance: for example, in Figure 1(c) we display results for $K = 3$ and $N = 5$. Here there are a larger number of possible incentive-action combinations (3^5) relative to the available horizon, and so consecutively stepping through the alternate agent actions may not be the optimal strategy.

However SEQ-MDP performs very close to the optimal algorithm's performance, and it is significantly better than other approaches. The SEQ-MDP approximation also requires much less computation time than the optimal algorithm. Figures 2(a) and 2(b) compare the average computational time to execute a single run ($H = 20$). Figure 2(a) fixes the number $N = 3$ of alternate agent actions, and varies the number of offered incentives K, and Figure 2(b) fixes $K = 4$ and varies N. The SEQ-MDP algorithm scales much better than the optimal algorithm in both cases.

8 Discussion and Extensions

So far we have focused on Incentive Decision Processes where a principal interacts with a myopic greedy agent whose preferences are hidden and static. Here we briefly consider two extensions of this model.

Dynamic Preferences: In many real applications the agent's preferences will change over time, and the principal may not know if the agent's preferences have changed. This significantly complicates the decision problem. One important common class of dynamic changes is where the preferences change locally. For example, a tenant's thermal preferences may locally shift during seasonal changes. Let us assume that if true incentive t_n for alternate agent action a_n is δ_k, then t_n remains the same with probability λ_0, shifts to δ_{k+d} with probability λ_1, and shifts to δ_{k-d} with probability λ_2 after each potential preference change[2]. We will restrict our attention to the case where the maximum number of times the agent might change its preferences is at most L. For simplicity we will also assume that we know the time epochs at which changes can occur, but our results also apply when we do not have prior information about these time epochs. Similarly, we will state our results for $N = 1$ but these results can be extended to $N > 1$.

Such dynamic preference IDPs can be represented as a POMDP with $O(K^3)$ states: in a static IDP there are $O(K^2)$ belief states; a change in preference shifts the non-zero elements of the belief state; there are at most $O(K)$ possible shifts; and as we don't know whether a shift occurs or not, this becomes the hidden state of our POMDP. Interestingly, we can reduce this POMDP to a finite state MDP:

Theorem 8.1. *A dynamic preference IDP ($N=1$) can be represented as a MDP with $O(K^{2(L+1)})$ states.*

This result is significant because we have mapped a non-deterministic POMDP to a finite state MDP. While a similar result is known to hold for deterministic POMDPs, it does not hold for general POMDPs, which typically map to a MDP with an infinite number of states. It is important to emphasize that this result holds for infinite horizon dynamic preference IDPs.

We believe that substantial further reductions in the state space may be possible with minimal performance loss since each possible change is local and small.

Rational Strategic Agents: Often the agent itself will seek to optimize its long term expected sum of rewards, reasoning about the potential future incentives the principal might provide if the agent chooses to accept or reject different offers. In such situations, the previous reduction to a MDP will no longer hold because the principal cannot trust that the agent's responses reflect its true hidden incentives.

Many of the policies we have considered so far will provide a higher incentive when an offer is rejected, and a lower incentive if the offer is accepted. Let us denote such policies π^{simple}. Such policies are strategy proof for a horizon of $H = 1$:

Theorem 8.2. *The agent will act truthfully for $H=1$.*

Indeed it is fairly easy to see that acting truthfully will always maximize the agent's immediate reward. However, this does not hold for longer horizons.

Theorem 8.3. *π^{simple} is not a strategy proof policy for arbitrary horizons.*

The proof provides an example of a $H = 2$ IDP where the agent can increase its reward by acting in discord with its true preferences. In the future we intend to develop algorithms for automatically constructing equilibrium decision policies for the principal and agent, and to design strategy proof policies.

Other directions: There are multiple other extensions to the Incentive Design Problems we have considered in this paper. Another interesting variant is when there are multiple agents (say U), and the principal can only provide incentives to 1 agent at each time step. It can be shown that an approach similar to binary search will lead to an $O((c_2-c_1)U \log K)$ additive bound with respect to the optimal algorithm in case of single alternate action. Interestingly, this problem can also be reduced to multi-arm bandit (MAB) problem with superprocesses [8]. Though computing optimal policies for superprocess MABs is generally difficult, it may be possible to leverage the IDP structure to compute efficient algorithms. We also plan to use IDPs to construct personalized adaptive incentives for reducing energy consumption.

9 Conclusion

We have introduced Incentive Decision Processes, where the objective is to compute a decision policy for a principal to minimize its expected sum of costs by providing incentives to a myopic agent. If the agent's hidden preferences are static, then we can represent an IDP (either exactly for 1 alternate action, or approximately for multiple actions) as a polynomially-sized Markov decision process, instead of as a standard POMDP. Our empirical results showed our sequential decision theoretic techniques significantly outperform simpler comparison algorithms.

[2] $\lambda_0 + \lambda_1 + \lambda_2 = 1$.

References

[1] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[2] B. Bonet. Deterministic POMDPs revisited. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2009.

[3] C. Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, pages 239–246, 2002.

[4] Y. Chen, J. Kung, D. Parkes, A. Procaccia, and H. Zhang. Incentive design for adaptive agents. In *AAMAS*, 2011.

[5] P. Doshi, Y. Zeng, and Q. Chen. Graphical models for interactive POMDPs: representations and solutions. *Autonomous Agents and Multi-Agent Systems*, 18:376–416, 2009.

[6] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[7] M. Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Brown University, 1996.

[8] A. Mahajan and D. Teneketzis. *Multi-armed Bandit Problems*, chapter 6. 2007.

[9] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[10] E. Sondik. The Optimal Control of Partially Observable Markov Processes. *DTIC Research Report AD0730503*, 1971.

[11] L. Zettlemoyer, B. Milch, and L. Pack Kaelbling. Multi-agent filtering with infinitely nested beliefs. In *NIPS*, 2008.

[12] H. Zhang and D. Parkes. Value-based policy teaching with active indirect elicitation. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2008.

[13] H. Zhang, Y. Chen, and D. Parkes. A general approach to environment design with one agent. In *IJCAI*, 2009.

[14] H. Zhang, D. Parkes, and Y. Chen. Policy teaching through reward function learning. In *EC*, 2009.

Active Imitation Learning via Reduction to I.I.D. Active Learning

Kshitij Judah
School of E.E.C.S.
Oregon State University
Corvallis, OR 97331-5501, USA
judahk@eecs.oregonstate.edu

Alan P. Fern
School of E.E.C.S.
Oregon State University
Corvallis, OR 97331-5501, USA
afern@eecs.oregonstate.edu

Thomas G. Dietterich
School of E.E.C.S.
Oregon State University
Corvallis, OR 97331-5501, USA
tgd@cs.orst.edu

Abstract

In standard passive imitation learning, the goal is to learn a target policy by passively observing full execution trajectories of it. Unfortunately, generating such trajectories can require substantial expert effort and be impractical in some cases. In this paper, we consider active imitation learning with the goal of reducing this effort by querying the expert about the desired action at individual states, which are selected based on answers to past queries and the learner's interactions with an environment simulator. We introduce a new approach based on reducing active imitation learning to i.i.d. active learning, which can leverage progress in the i.i.d. setting. Our first contribution, is to analyze reductions for both non-stationary and stationary policies, showing that the label complexity (number of queries) of active imitation learning can be substantially less than passive learning. Our second contribution, is to introduce a practical algorithm inspired by the reductions, which is shown to be highly effective in four test domains compared to a number of alternatives.

1 Introduction

Traditionally, passive imitation learning involves learning a policy that performs nearly as well as an expert's policy based on a set of trajectories of that policy. However, generating such trajectories is often tedious or even impractical for an expert (e.g. real-time low-level control of multiple game agents). In order to address this issue, we consider active imitation learning where full trajectories are not required, but rather the learner asks queries about specific states, which the expert labels with the correct actions. The goal is to learn a policy that is nearly as good as the expert's policy using as few queries as possible.

The active learning problem for i.i.d. supervised learning has received considerable attention both in theory and practice (Settles, 2009), which motivates attempting to leverage that work for active imitation learning. However, the direct application of i.i.d. approaches to active imitation learning can be problematic. This is because i.i.d. active learning algorithms assume access to either a target distribution over unlabeled input data (in our case states) or a large sample drawn from it. The goal then is to select the most informative query to ask, usually based on some combination of label (in our case actions) uncertainty and unlabeled data density. Unfortunately, in active imitation learning, the learner does not have direct access to the target state distribution, which is the state distribution induced by the unknown expert policy.

In principle, one could approach active imitation learning by assuming a uniform or an arbitrary distribution over the state space and then apply an existing i.i.d. active learner. However, such an approach can perform very poorly. This is because if the assumed distribution is considerably different from that of the expert, then the learner is prone to ask queries in states rarely or even never visited by the expert. For example, consider a bicycle balancing problem. Clearly, asking queries in states where the bicycle has entered an unavoidable fall is not very useful because no action can prevent a crash. However, i.i.d. active learning technique will tend to query in such uninformative states, leading to poor performance, as shown in our experiments. Furthermore, in the case of a human expert, a large number of such queries poses serious usability issues, since labeling such states is clearly a wasted effort from the expert's perspective.

In this paper, we consider the problem of reducing active imitation learning to i.i.d. active learning both in theory and practice. Our first contribution is to analyze the PAC label complexity (number of expert queries) of a reduction for learning non-stationary policies, which requires only minor modification to exist-

ing results for passive learning. Our second contribution is to introduce a reduction for learning stationary policies resulting in a new algorithm *Reduction-based Active Imitation Learning (RAIL)* and an analysis of the label complexity. The resulting complexities for active imitation learning are expressed in terms of the label complexity for the i.i.d. case and show that there can be significant query savings compared to existing results for passive imitation learning. Our third contribution is to describe a new practical algorithm, RAIL-DW, inspired by the RAIL algorithm, which makes a series of calls to an i.i.d. active learning algorithm. We evaluate RAIL-DW in four test domains and show that it is highly effective when used with an i.i.d. algorithm that takes the unlabeled data density into account.

2 Related Work

Active learning has been studied extensively in the i.i.d. supervised learning setting (Settles, 2009) but to a much lesser degree for sequential decision making, which is the focus of active imitation learning. Several studies have considered active learning for *reinforcement learning (RL)* (Clouse, 1996; Mihalkova and Mooney, 2006; Gil et al., 2009; Doshi et al., 2008), where learning is based on both autonomous exploration and queries to an expert. Rather, in our imitation learning framework, we do not assume a reward signal and learn only from expert queries. Other prior work (Shon et al., 2007) studies active imitation learning in a multiagent setting where the expert is itself a reward seeking agent and hence is not necessarily a helpful expert. Here we consider only helpful experts.

One approach to imitation learning is *inverse RL (IRL)* (Ng and Russell, 2000), where a reward function is learned based on a set of target policy trajectories. The learned reward function and transition dynamics are then given to a planner to obtain a policy. There has been limited work on active IRL. This includes a Bayesian approach (Lopes et al., 2009), where a posterior over reward functions is used to select a "most informative" query. Another Bayesian approach (Cohn et al., 2010) models uncertainty about the entire MDP model and uses Expected Myopic Gain (EMG) to select a query about transition dynamics and rewards. While promising, the scalability of these approaches is hindered by the assumptions made by IRL and have only been demonstrated on small problems. In particular, they require that the exact domain dynamics are provided or can be learned and that an efficient planner is available.

To facilitate scalability, rather than follow an IRL framework we consider a *direct imitation* framework where we attempt to directly learn a policy instead of the reward function and/or transition dynamics. Unlike inverse RL, this framework does not require an exact dynamic model nor an efficient planner. Rather, our approach requires only a simulator of the environment dynamics that is able to generate trajectories given a policy. Such a simulator is often available, even when a compact description of the transition dynamics and/or a planner are not.

Recent active learning work in the direct imitation framework includes confidence based autonomy (Chernova and Veloso, 2009), and the related dogged learning framework (Grollman and Jenkins, 2007), where a policy is learned as it is executed. When the learner is uncertain about what to do at a state, the policy is paused and the expert is queried about what action to take, resulting in a policy update. One difficulty in applying this approach is setting the uncertainty threshold for querying the expert. While an automated threshold selection approach is suggested (Chernova and Veloso, 2009), our experiments show that it is not always effective. Like our work, this approach requires a dynamics simulator.

Recently, Ross and Bagnell (2010); Ross et al. (2011) proposed algorithms for imitation learning that are able to actively query the expert at particular states during policy execution. They show that under certain assumptions these algorithms have better theoretical performance guarantees than pure passive imitation learning. Unfortunately, these algorithms query the expert quite aggressively making them impractical for human experts or computationally expensive with automated experts. In contrast, our work focuses on active querying for the purpose of minimizing the expert's labeling effort. Like our work, they also require a dynamics simulator to help select queries.

3 Problem Setup and Background

We consider imitation learning in the framework of Markov decision processes (MDPs). An MDP is a tuple $\langle S, A, T, R, I \rangle$, where S is the set of states, A is the finite set of actions, $T(s, a, s')$ is the transition function denoting the probability of transitioning to state s' upon taking action a in state s, $R(s) \in [0, 1]$ is the reward function giving the immediate reward in state s, and I is the initial state distribution. A stationary policy $\pi : S \mapsto A$ is a deterministic mapping from states to actions such that $\pi(s)$ indicates the action to take in state s when executing π. A non-stationary policy is a tuple $\pi = (\pi_1, \ldots, \pi_T)$ of T stationary policies such that $\pi(s, t) = \pi_t(s)$ indicates the action to take in state s and at time t when executing π, where T is the time horizon. The expert's policy, which we assume is deterministic, is denoted as π^*.

The T-horizon value of a policy $V(\pi)$ is the expected total reward of trajectories that start in $s_1 \sim I$ at

time $t = 1$ and then execute π for T steps. We use d_π^t to denote the state distribution induced at time step t by starting in $s_1 \sim I$ and then executing π. Note that $d_\pi^1 = I$ for all policies. We use $d_\pi = \frac{1}{T}\sum_{t=1}^{T} d_\pi^t$ to denote the state distribution induced by policy π over T time steps. To sample an (s, a) pair from d_π^t, we start in $s_1 \sim I$, execute π to generate a trajectory $\mathcal{T} = (s_1, a_1, \ldots, s_T, a_T, s_{T+1})$ and set $(s, a) = (s_t, a_t)$. Similarly, to sample from d_π, we first sample a random time step $t \in \{1, \ldots, T\}$, and then sample an (s, a) pair from d_π^t. Note that in order to sample from d_{π^*} (or $d_{\pi^*}^t$), we need to execute π^*. Throughout the paper, we assume that the only way π^* can be executed is by querying the expert for an action in the current state and executing the given action, which puts significant burden on the expert.

The regret of a policy π with respect to an expert policy π^* is equal to $V(\pi^*) - V(\pi)$. In imitation learning, the goal is to learn a policy π from a hypothesis class H (e.g. linear action classifiers), that has a small regret. In the passive setting of imitation learning, the learner is provided with a training set of full execution trajectories of π^* and the state-action pairs (or a sample of them) are passed to an i.i.d. supervised learning algorithm. To help avoid the cost of generating full trajectories, active imitation learning allows the learner to pose *action queries*. In an action query, a state s is presented to the expert and the expert returns the desired action $\pi^*(s)$.

In addition to having access to the expert for answering queries, we assume that the learner has access to a simulator of the MDP. The input to the simulator is a policy π and a horizon T. The simulator output is a state trajectory that results from executing π for T steps starting in the initial state. The learner is allowed to interact with this simulator as part of its query selection process. The simulator is not assumed to provide a reward signal, which means that the learner cannot find π by pure reinforcement learning.

Since our analysis in the next two sections is based on reducing to i.i.d. active learning and comparing to i.i.d. passive learning, we briefly review the *Probably Approximately Correct (PAC)* (Valiant, 1984) learning formulation for the i.i.d. setting. Here we consider the realizable PAC setting, which will be the focus of our initial analysis. Section 4.3, extends to the non-realizable, or agnostic setting. In passive i.i.d. supervised learning, N i.i.d. data samples are drawn from an unknown distribution $D_\mathcal{X}$ over an input space $\mathcal{X}$ and are labeled according to an unknown target classifier $f : \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{Y}$ denotes the label space. In the realizable PAC setting it is assumed that f is an element of a known class of classifiers H and given a set of N examples a learner then outputs a hypothesis $h \in H$. Let $e_f(h, D_\mathcal{X}) = \mathbb{E}_{x \sim D_\mathcal{X}}[h(x) \neq f(x)]$ denote the generalization error of the returned classifier h. Standard PAC learning theory provides a bound on the number of labeled examples that are sufficient to guarantee that for any distribution $D_\mathcal{X}$, with probability at least $1-\delta$, the returned classifier h will satisfy $e_f(h, D_\mathcal{X}) \leq \epsilon$. We will denote this bound by $N_p(\epsilon, \delta)$, which corresponds to the label/query complexity of i.i.d. passive supervised learning for a class H. We will also denote a passive learner that achieves this label complexity as $L_p(\epsilon, \delta)$.

In i.i.d. active learning, the learner is given access to two resources rather than just a set of training data: 1) A "cheap" resource (Sample) that can draw an unlabeled sample from $D_\mathcal{X}$ and provide it to the learner when requested, 2) An "expensive" resource (Label) that can label a given unlabeled sample according to target concept f when requested. Given access to these two resources, an active learning algorithm is required to learn a hypothesis $h \in H$ while posing as few queries to Label as possible. It can, however, pose a much larger number of queries to Sample (though still polynomial) as it is cheap. We use $N_a(\epsilon, \delta)$ to denote the label complexity (i.e. number of calls to Label) that is sufficient for an active learner to return an h that for any $D_\mathcal{X}$ with probability at least $1 - \delta$ satisfies $e_f(h, D_\mathcal{X}) \leq \epsilon$. Similarly, we will denote an active learner that achieves this label complexity as $L_a(\epsilon, \delta, D)$, where the final argument D indicates that the Sample function used by L_a samples from distribution D. It has been shown that often N_a can be exponentially smaller than N_p for hypothesis classes with bounded complexity (e.g. VC-dimension). In particular, in the realizable case, ignoring δ, $N_p = O(\frac{1}{\epsilon})$ whereas $N_a = O(\log(\frac{1}{\epsilon}))$ giving exponential improvement in label complexity over passive learning.

4 Reductions for Active Imitation Learning

We now consider reductions from active imitation learning to active i.i.d. learning for the cases of deterministic non-stationary and stationary policies.

4.1 Non-Stationary Policies

Syed and Schapire (2010) analyze the traditional reduction from passive imitation learning to passive i.i.d. learning for non-stationary policies. The algorithm uses queries to sample N expert trajectories, noting that the state-action pairs at time t can be viewed as i.i.d. draws from distribution $d_{\pi^*}^t$. The algorithm, then returns the non-stationary policy $\hat{\pi} = (\hat{\pi}_1, \ldots, \hat{\pi}_T)$, where $\hat{\pi}_t$ is the policy returned by running the learner L_p on examples from time t. Let $\epsilon_t = e_{\pi_t^*}(\hat{\pi}_t, d_{\pi^*}^t)$ be the generalization error at time

t. Lemma 3[1] in (Syed and Schapire, 2010) shows that if for each time step $\epsilon_t \leq \epsilon$, then $V(\hat{\pi}) \geq V(\pi^*) - \epsilon T^2$. Hence, if we are interested in learning a $\hat{\pi}$ whose overall regret is no more than ϵ, then we need to provide at least $N_p(\frac{\epsilon}{T^2}, \frac{\delta}{T})$ examples to L_p to ensure that $\epsilon_t \leq \frac{\epsilon}{T^2}$ holds at all time steps with probability at least $1 - \delta$. Therefore, the passive label complexity of this algorithm is $T \cdot N_p(\frac{\epsilon}{T^2}, \frac{\delta}{T})$.

Our goal now is to provide a reduction from active imitation learning to i.i.d. active learning that can achieve an improved label complexity. This is not as simple as replacing the calls to L_p in the above approach with calls to an active learner L_a. This is because the active learner at time step t would require the ability to sample from the unlabeled distribution $d^t_{\pi^*}$, which requires executing the expert policy for t steps, which in turn requires t label queries to the expert that would count against the label complexity.

It turns out that for a slightly more sophisticated reduction to i.i.d. passive learning introduced by Ross and Bagnell (2010), it is possible to simply replace L_p with L_a and maintain the potential benefit of active learning. Ross and Bagnell (2010) introduced the forward training algorithm for non-stationary policies, which trains a non-stationary policy in a series of T iterations. In particular, iteration t trains policy $\hat{\pi}_t$ by calling a passive learner L_p on a labeled data set drawn from the state distribution induced at time t by the non-stationary policy $\hat{\pi}^{t-1} = (\hat{\pi}_1, \ldots, \hat{\pi}_{t-1})$, where $\hat{\pi}_1$ is learned on states drawn from the initial distribution I. The motivation for this approach is to train the policy at time step t based on the same state-distribution that it will encounter when being run after learning. By doing this, Ross and Bagnell (2010) show that the algorithm has a worst case regret of $T^2\epsilon$ and under certain assumptions can achieve a regret as low as $O(T\epsilon)$.

Importantly, the state-distribution used to train $\hat{\pi}_t$ given by $d^t_{\hat{\pi}^{t-1}}$ is easy for the learner to sample from without making queries to the expert. In particular, to generate a sample the learner can simply simulate $\hat{\pi}^{t-1}$, which is available from previous iterations, from a random initial state and return the state at time t. Thus, we can simply replace the call to L_p at iteration t with a call to L_a with unlabeled state distribution $d^t_{\hat{\pi}^{t-1}}$ as input. More formally, the *active forward training algorithm* is given by the following iteration: $\hat{\pi}_t = L_a(\epsilon, \frac{\delta}{T}, D^t)$, where $D^1 = I$ and $D^t = d^t_{\hat{\pi}^{t-1}}$.

Theorem 3.1 in (Ross and Bagnell, 2010) gives the worst case bound on the regret of the forward training algorithm which assumes the generalization error at each iteration is bounded by ϵ. Since we also maintain that assumption when replacing L_p with L_a (the active variant) we immediately inherit that bound.

Proposition 1. *Given a PAC i.i.d. active learning algorithm L_a, if active forward training is run by giving L_a parameters ϵ and $\frac{\delta}{T}$ at each step, then with probability at least $1-\delta$ it will return a non-stationary policy $\hat{\pi}^T$ such that $V(\hat{\pi}^T) \geq V(\pi^*) - \epsilon T^2$.*

Note that L_a is run with $\frac{\delta}{T}$ as the reliability parameter to ensure that all T iterations succeed with the desired probability. Proposition 1 shows that the overall label complexity of active forward training in order to achieve a regret less than ϵ with probability at least $1 - \delta$ is $T \cdot N_a(\frac{\epsilon}{T^2}, \frac{\delta}{T})$. Comparing to the corresponding label complexity of passive imitation learning $T \cdot N_p(\frac{\epsilon}{T^2}, \frac{\delta}{T})$ we see that an improved label complexity of active learning in the i.i.d. case translates to improved label complexity of active imitation learning. In particular, in the realizable learning case, we know that N_a can be exponentially smaller than N_p in terms of its dependence on $\frac{1}{\epsilon}$.

4.2 Stationary Policies

A drawback of active forward training is that it is impractical for large T and the resulting policy cannot be run indefinitely. We now consider the case of learning stationary policies, first reviewing the existing results for passive imitation learning.

In the traditional approach, a stationary policy $\hat{\pi}$ is trained on the expert state distribution d_{π^*} using a passive learning algorithm L_p returning a stationary policy $\hat{\pi}$. Theorem 2.1 in (Ross and Bagnell, 2010) states that if the generalization error of $\hat{\pi}$ with respect to the i.i.d. distribution d_{π^*} is bounded by ϵ then $V(\hat{\pi}) \geq V(\pi^*) - \epsilon T^2$. Since generating i.i.d. samples from d_{π^*} can require up to T queries (see Section 3) the passive label complexity of this approach for guaranteeing a regret less than ϵ with probability at least $1 - \delta$ is $T \cdot N_p(\frac{\epsilon}{T^2}, \delta)$.

The above approach cannot be converted into an active imitation learner by simply replacing the call to L_p with L_a, since again we cannot sample from the unlabeled distribution d_{π^*} without querying the expert. To address this issue, we introduce a new algorithm called *RAIL (Reduction-based Active Imitation Learning)* which makes a sequence of T calls to an i.i.d. active learner, noting that it is likely to find a useful stationary policy well before all T calls are issued. RAIL is an idealized algorithm intended for analysis, which achieves the theoretical goals but has a number of inefficiencies from a practical perspective. Later in Section 5 we describe the practical instantiation that

[1]The main result of (Syed and Schapire, 2010) holds for stochastic expert policies and requires a more complicated analysis that results in a looser bound. Lemma 3 is strong enough for deterministic expert policies, which is the assumption made in our work.

is used in our experiments.

RAIL is similar in spirit to active forward training, though its analysis is quite different and more involved. Like forward-training RAIL iterates for T iterations, but on each iteration, RAIL learns a new stationary policy $\hat{\pi}^t$ that can be applied across all time steps. Iteration $t+1$ of RAIL learns a new policy $\hat{\pi}^{t+1}$ that achieves a low error rate at predicting the expert's actions with respect to the state distribution of the previous policy $d_{\hat{\pi}^t}$. More formally, given an i.i.d. active learner L_a, the RAIL algorithm is defined by the following iteration:

- **(Initialize)** $\hat{\pi}^0$ is an arbitrary policy, possibly based on prior knowledge or existing data,
- **(Iterate** $t = 1 \cdots, T$**)** $\hat{\pi}^t = L_a(\epsilon, \frac{\delta}{T}, d_{\hat{\pi}^{t-1}})$.

Thus, similar to active forward training, RAIL makes a sequence of T calls to an active learner. Unlike forward training, however, the unlabeled data distributions used at each iteration contains states from all time points within the horizon, rather than being restricted to a states arising at a particular time point. Because of this difference, the active learner is able to ask queries across a range of time point and we might expect policies learned in earlier iterations to achieve non-trivial performance throughout the entire horizon. In contrast, at iteration t the policy produced by forward training is only well defined up to time t.

The complication faced by RAIL, however, compared to forward training, is that the distribution used to train $\hat{\pi}^{t+1}$ differs from the state distribution of the expert policy d_{π^*}. This is particularly true in early iterations of RAIL since $\hat{\pi}^0$ is initialized arbitrarily. We now show that as the iterations proceed, we can bound the similarity between the state distributions of the learned policy and the expert, which allows us to bound the regret of the learned policy. We first state the main result which we prove below.

Theorem 1. *Given a PAC i.i.d. active learning algorithm L_a, if RAIL is run with parameters ϵ and $\frac{\delta}{T}$ passed to L_a at each iteration, then with probability at least $1-\delta$ it will return a stationary policy $\hat{\pi}^T$ such that $V(\hat{\pi}^T) \geq V(\pi^*) - \epsilon T^3$.*

From this we see that the impact of moving from non-stationary to stationary policies in the worst case is a factor of T in the regret bound. Similarly the bound is a factor of T worse than the comparable result above for passive imitation learning, which suffered a worst-case regret of ϵT^2. From this we see that the total label complexity for RAIL required to guarantee a regret of ϵ with probability $1-\delta$ is $T \cdot N_a(\frac{\epsilon}{T^3}, \frac{\delta}{T})$ compared to the above label complexity of passive learning $T \cdot N_p(\frac{\epsilon}{T^2}, \delta)$. Thus, for a given policy class, if the label complexity of i.i.d. active learning is substantially less than the label complexity of passive learning, then our reduction can leverage those savings. For example, in the realizable learning case, ignore the dependence on δ (which is only logarithmic), we get an active label complexity of $T \cdot O(\log \frac{T^3}{\epsilon})$ versus the corresponding passive complexity of $T \cdot O(\frac{T^2}{\epsilon})$.

For the proof we introduce the quantity $P_\pi^t(M)$, which is the probability that a policy π is consistent with a length t trajectory generated by the expert policy π^* in MDP M. It will also be useful to index the state distribution of π by the MDP M, denoted by $d_\pi(M)$. The main idea is to show that at iteration t, $P_{\hat{\pi}^t}^t(M)$ is not too small, meaning that the policy at iteration t mostly agrees with the expert for the first t actions. We first state two lemmas, which are useful for the final proof. First, we bound the regret of a policy in terms of $P_\pi^T(M)$.

Lemma 1. *For any policy π, if $P_\pi^T(M) \geq 1-\epsilon$, then $V(\pi) \geq V(\pi^*) - \epsilon T$.*

Proof. Let Γ^* and Γ be all state-action sequences of length T that are consistent with π^* and π respectively. If $R(\mathcal{T})$ is the total reward for a sequence $\mathcal{T}$ then we get the following:

$$\begin{aligned}
V(\pi) &= \sum_{\mathcal{T}\in\Gamma} \Pr(\mathcal{T} \mid M, \pi) R(\mathcal{T}) \\
&\geq \sum_{\mathcal{T}\in\Gamma\cap\Gamma^*} \Pr(\mathcal{T} \mid M, \pi) R(\mathcal{T}) \\
&= \sum_{\mathcal{T}\in\Gamma^*} \Pr(\mathcal{T} \mid M, \pi^*) R(\mathcal{T}) - \sum_{\mathcal{T}\in\Gamma^*-\Gamma} \Pr(\mathcal{T} \mid M, \pi^*) R(\mathcal{T}) \\
&= V(\pi^*) - \sum_{\mathcal{T}\in\Gamma^*-\Gamma} \Pr(\mathcal{T} \mid M, \pi^*) R(\mathcal{T}) \\
&\geq V(\pi^*) - T \cdot \sum_{\mathcal{T}\in\Gamma^*-\Gamma} \Pr(\mathcal{T} \mid M, \pi^*) \\
&\geq V(\pi^*) - \epsilon T
\end{aligned}$$

The last two inequalities follow since the reward for a sequence must be no more than T and our assumption about $P_\pi^T(M)$. □

Next, we show how the value of $P_\pi^t(M)$ changes across one iteration of learning.

Lemma 2. *For any policies π and $\hat{\pi}$ and $1 \leq t < T$, if $e_{\pi^*}(\hat{\pi}, d_\pi(M)) \leq \epsilon$, then $P_{\hat{\pi}}^{t+1}(M) \geq P_\pi^t(M) - T\epsilon$.*

Proof. We define $\hat{\Gamma}$ to be all sequences of state-action pairs of length $t+1$ that are consistent with $\hat{\pi}$. Also define Γ to be all length $t+1$ state-action sequences that are consistent with π on the first t state-action pairs (so need not be consistent on the final pair). We also define $\hat{M}$ to be an MDP that is identical to M, except that the transition distribution of any state-action pair (s,a) is equal to the transition distribution

of action $\pi(s)$ in state s. That is, all actions taken in a state s behave like the action selected by π in s.

We start by arguing that if $e_{\pi^*}(\hat{\pi}, d_\pi(M)) \leq \epsilon$ then $P_{\hat{\pi}}^{t+1}(\hat{M}) \geq 1 - T\epsilon$, which relates our error assumption to the MDP $\hat{M}$. To see this note that for MDP $\hat{M}$, all policies including π^*, have state distribution given by d_π. Thus by the union bound $1 - P_{\hat{\pi}}^{t+1}(\hat{M}) \leq \sum_{i=1}^{t+1} \epsilon_i$, where ϵ_i is the error of $\hat{\pi}$ at predicting π^* on distribution d_π^i. This sum is bounded by $T\epsilon$ since $e_{\pi^*}(\hat{\pi}, d_\pi(M)) = \frac{1}{T}\sum_{i=1}^{T} \epsilon_i$. Using this fact we can now derive the following:

$$
\begin{aligned}
P_{\hat{\pi}}^{t+1}(M) &= \sum_{\mathcal{T} \in \hat{\Gamma}} \Pr(\mathcal{T} \mid M, \pi^*) \\
&\geq \sum_{\mathcal{T} \in \Gamma \cap \hat{\Gamma}} \Pr(\mathcal{T} \mid M, \pi^*) \\
&= \sum_{\mathcal{T} \in \Gamma} \Pr(\mathcal{T} \mid M, \pi^*) - \sum_{\mathcal{T} \in \Gamma - \hat{\Gamma}} \Pr(\mathcal{T} \mid M, \pi^*) \\
&= P_\pi^t(M) - \sum_{\mathcal{T} \in \Gamma - \hat{\Gamma}} \Pr(\mathcal{T} \mid M, \pi^*) \\
&= P_\pi^t(M) - \sum_{\mathcal{T} \in \Gamma - \hat{\Gamma}} \Pr(\mathcal{T} \mid \hat{M}, \pi^*) \\
&\geq P_\pi^t(M) - \sum_{\mathcal{T} \notin \hat{\Gamma}} \Pr(\mathcal{T} \mid \hat{M}, \pi^*) \\
&\geq P_\pi^t(M) - T\epsilon
\end{aligned}
$$

The equality of the fourth line follows since Γ contains all sequences whose first t actions are consistent with π with all possible combinations of the remaining action and state transition. Thus, summing over all such sequences yields the probability that π^* agrees with the first t steps. The equality of the fifth line follows because $\Pr(\mathcal{T} \mid M, \pi^*) = \Pr(\mathcal{T} \mid \hat{M}, \pi^*)$ for any $\mathcal{T}$ that is in Γ and for which π^* is consistent (has non-zero probability under π^*). The final line follows from the above observation that $P_{\hat{\pi}}^{t+1}(\hat{M}) \geq 1 - T\epsilon$. □

We can now complete the proof of the main theorem.

Proof of Theorem 1. Using failure parameter $\frac{\delta}{T}$ ensures that with at least probability $1 - \delta$ that for all $1 \leq t < T$ we will have $e_{\pi^*}(\hat{\pi}^{t+1}, d_{\hat{\pi}^t}(M)) \leq \epsilon$. As a base case, we have $P_{\hat{\pi}^1}^1 \geq 1 - T\epsilon$, since the the error rate of $\hat{\pi}^1$ relative to the initial state distribution at time step $t = 1$ is at most $T\epsilon$. Combining these facts with Lemma 2 we get that $P_{\hat{\pi}^T}^T \geq 1 - \epsilon T^2$. Combining this with Lemma 1 completes the proof. □

4.3 Agnostic Case

Above we considered the realizable setting, where the expert's policy was assumed to be in a known hypothesis class H. In the agnostic case, we do not make such an assumption. The learner still outputs a hypothesis from a class H, but the unknown policy is not necessarily in H. The agnostic i.i.d. PAC learning setting is defined similarly to the realizable setting, except that rather than achieving a specified error bound of ϵ with high probability, a learner must guarantee an error bound of $\inf_{\pi \in H} e_f(\pi, D_\mathcal{X}) + \epsilon$ with high probability (where f is the target), where $D_\mathcal{X}$ is the unknown data distribution. That is, the learner is able to achieve close to the best possible accuracy given class H. In the agnostic case, it has been shown that exponential improvement in label complexity with respect to $\frac{1}{\epsilon}$ is achievable when $\inf_{\pi \in H} e_f(\pi, D_\mathcal{X})$ is relatively small compared to ϵ (Dasgupta, 2011). Further, there are many empirical results for practical active learning algorithms that demonstrate improved label complexity compared to passive learning.

It is straightforward to extend our above results for non-stationary and stationary policies to the agnostic case by using agnostic PAC learners for L_p and L_a. Space precludes full details and here we outline the extension for RAIL. Note that the RAIL algorithm will call L_a using a sequence of unlabeled data distributions, where each distribution is of the form d_π for some $\pi \in H$ and each of which may yield a different minimum error given H. For this purpose, we define $\epsilon^* = \sup_{\pi \in H} \inf_{\hat{\pi} \in H} e_{\pi^*}(\hat{\pi}, d_\pi)$ to be the minimum generalization error achievable in the worst case considering all possible state distributions d_π that RAIL might possibly encounter. With minimal changes to the proof of Theorem 1, we can get an identical result, except that the regret is $(\epsilon^* + \epsilon)T^3$ rather than just ϵT^3. A similar change in regret holds for passive imitation learning. This shows that in the agnostic setting we can get significant improvements in label complexity via active imitation learning when there are significant savings in the i.i.d. case.

5 Practical Instantiation of RAIL

Despite the theoretical guarantees, a potential drawback of RAIL is that the unlabeled state distributions used at early iterations may be quite different from d_{π^*}. In particular, at iteration t, the distributions at times $t' > t$ have no guarantees with respect to $d_{\pi^*}^{t'}$. Thus, early iterations may focus substantial query effort on parts of the state-space that are not relevant to π^*. Another issue is that the idealized version does not share data across iterations, which is potentially wasteful. We now describe our practical instantiation of RAIL, called RAIL-DW (for density weighted), which addresses these issues in several ways.

First, we use an incremental version of RAIL that asks only one query per iteration and accumulates data across iterations. This allows rapid updating of state distributions and prevents RAIL from wasting its query budget on earlier inaccurate distributions but rather focus on later more accurate distributions.

Second, recall that at iteration $t+1$, RAIL learns using an unlabeled data distribution $d_{\hat{\pi}^t}$, where $\hat{\pi}^t$ is the policy learned at iteration t. In order to help improve the

accuracy of this unlabeled distribution (with respect to d_{π^*}), instead of using a point estimate for $\hat{\pi}^t$, we treat $\hat{\pi}^t$ as a Bayesian classifier for purposes of defining $d_{\hat{\pi}^t}$. In particular, at iteration t let D_t be the set of state-action pairs collected from previous iterations. We use this to define a posterior $P(\hat{\pi}|\mathcal{D})$ over policies in our policy class H. This, in turn, defines a posterior unlabeled state distribution $d_{D_t} = \mathbb{E}_{\hat{\pi}\sim P(\hat{\pi}|\mathcal{D})}[d_{\hat{\pi}}(s)]$ which is used in place of $d_{\hat{\pi}^t}$ in RAIL. Note that we can sample from this distribution by first sampling a policy $\hat{\pi}$ and then sampling a state from $d_{\hat{\pi}}$, all of which can be done without interaction with the expert. We observe that in practice d_{D_t} is a significantly more useful estimate of d_{π^*} than the point estimate, since its more highly weighted states tend to carry significant weight according to d_{π^*}.

To summarize, iteration t of RAIL-DW carries out the following steps, adding one state-action pair to the data set: (D_1 is the initial, possibly empty, data set)

1. Run active learner L_a using unlabeled data distribution d_{D_t} to select a single query state s.
2. Query the expert about s to obtain action $\pi^*(s)$ and let $D_{t+1} = D_t \cup \{(s, \pi^*(s))\}$.

The iteration repeats until the query budget is exceeded. It remains to specify the specific i.i.d. active learning algorithm that we use and how we sample from d_{D_t}.

Since it is important that the i.i.d. active learner be sensitive to the unlabeled data distribution, we employ a density-weighted learning algorithm (hence the name RAIL-DW). In particular, we use density-weighted query-by-committee (McCallum and Nigam, 1998) in our implementation. Given a sample of unlabeled data points, this approach uses bagging (Breiman, 1996) to generate a committee for query selection and also uses a density estimator to estimate the density of the unlabeled data points. The selected query is the state that maximizes the product of state density and committee disagreement. We use the entropy of the vote distribution (Dagan and Engelson, 1995) as a typical measure of committee disagreement. We use a committee of size 5 in our experiments and estimate density of unlabeled points via simple distance based binning.

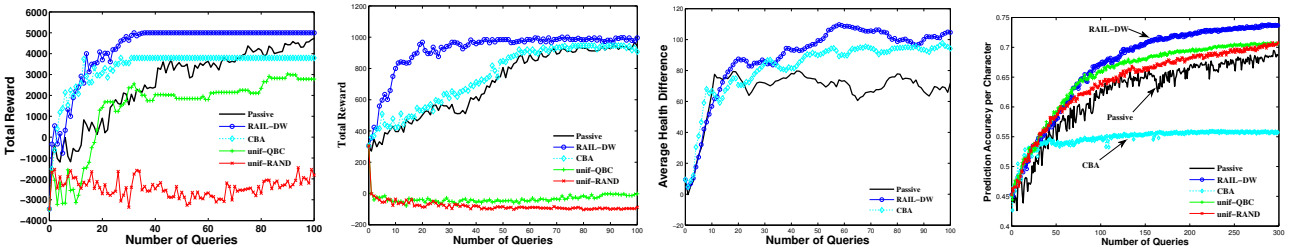
Finally, for sampling we assume a class of linear parametric policies and assume a uniform prior over the parameters. We approximate sampling from d_{D_t} via bagging: where we generate K bootstrap samples of D_t and a policy is learned from each using a supervised learner. This produces a set of K policies and each is executed to generate K trajectories. The states on those trajectories are given to the active learner as the unlabeled data set. We set $K = 5$ in our experiments.

6 Experiments

We empirically evaluate RAIL-DW on four domains: 1) Cart-pole, 2) Bicycle, 3) Wargus and 4) The structured prediction domain NETtalk. We compare RAIL-DW against the following baselines: 1) *Passive*, which simulates the traditional approach by starting at the initial state and querying the expert about what to do at each visited state, 2) *unif-QBC*, which views all the states as i.i.d. according to the uniform distribution and applies the standard query-by-committee (QBC) (Seung et al., 1992) active learning approach. Intuitively, this approach will select the state with highest action uncertainty according to the current data set and ignore the state distribution, 3) *unif-RAND*, which selects states to query uniformly at random, and 4) *Confidence based autonomy (CBA)* (Chernova and Veloso, 2009), which, starting at the initial state, executes the current policy until the learner's confidence falls below an automatically determined threshold at which point it queries the expert for an action. CBA may decide to stop asking queries once the confidence exceeds the threshold in all states. We use the exact automated threshold adjustment strategy proposed in (Chernova and Veloso, 2009). For all of these methods, we employed the SimpleLogistic classifier in Weka (Hall et al., 2009) to learn policies.

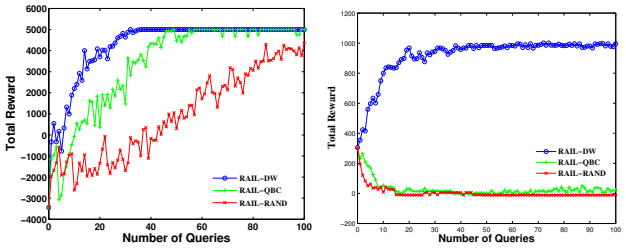
Cart-Pole. Cart-pole is an RL benchmark where a cart must balance an attached vertical pole by applying left or right forces to the cart. An episode ends when either the pole falls or the cart goes out of bounds. There are two actions, left and right, and four state variables describing the position and velocity of the cart and the angle and angular velocity of the pole. We made slight modifications to the usual setting where we allow the pole to fall down and become horizontal and the cart to go out of bounds (we used default [-2.4, 2.4] as the bounds for the cart). We let each episode run for a fixed length of 5000 time steps. This opens up the possibility of generating several "undesirable" states where either the pole has fallen or the cart is out of bounds that are rarely or never generated by the expert's state distribution. The expert policy was a hand-coded policy that can balance the pole indefinitely. For each learner, we ran experiments from 30 random initial states close to the equilibrium start state, generating learning curves for each one and averaging the results. We report total reward with a reward function (unknown to the learner) that is +1 for each time step where the pole is balanced and the cart is in bounds and -1 otherwise.

Figure 1(a) shows the results for cart-pole. We observe that RAIL learns quickly and achieves optimal performance with only 30-35 queries. Passive, on the other hand, takes 100 queries to get close to the optimal per-



(a) (b) (c) (d)

Figure 1: Active imitation learning results: (a) Cart-pole (b) Bicycle balancing (c) Wargus (d) NETtalk.



(a) (b)

Figure 2: Performance of RAIL with different base active learners on (a) Cart-pole (b) Bicycle balancing.

formance. The reason for this difference is clear when one visualizes the states queried by RAIL versus Passive (not shown for space reasons). The queries posed by RAIL tend to be close to the decision boundary of the expert policy, while passive asks many uninformative queries that are not close to the boundary. However, a naive reduction to active learning can be dangerous, as demonstrated by the poor performance of unif-QBC. Further, RAIL performs much better than random query selection as demonstrated by the performance of unif-RAND. By ignoring the real data distribution altogether and incorrectly assuming it to be uniform, these naive methods end up asking many queries that are not relevant to learning the expert policy (e.g. states where the pole is in an unrecoverable fall or the cart is out of bounds). CBA, like RAIL, learns quickly but settles at a suboptimal performance. This is because it becomes confident and stops asking queries prematurely. This shows that CBA's automatic threshold adjustment mechanism did not work well in this domain. We did try several adjustments to the threshold adjustment strategy, but were unable to find one that was robust across our four domains and thus we report results for the original strategy.

We also conducted experiments to study the effects of using i.i.d. active learners that ignore the data distribution in RAIL. Figure 2(a) shows the performance of RAIL with three different base active learners, the density-weighted QBC, the standard QBC (without density weighting), and random selection of unlabeled data points. We see that RAIL-DW performs better than both RAIL-QBC and RAIL-RAND. This shows that it is critical for the i.i.d. active learner to exploit the state density information that is estimated by RAIL at each iteration.

Bicycle Balancing. Bicycle balancing is a variant of the Bicycle RL benchmark (Randløv and Alstrøm, 1998). The goal is to balance a bicycle moving at a constant speed for 1000 time steps. If the bicycle falls, it remains fallen for the rest of the episode. The state space is described using nine variables measuring various angles, angular velocities, and positions of the bicycle. There are five possible actions each specifying a particular handle-bar torque and rider displacement. The learner's policy is represented as a linear logistic regression classifier over features of state-action pairs. A feature vector is defined as follows: Given a state s, a set of 20 basis functions is computed. This set is repeated for each of the 5 actions giving a feature vector of length 100. The expert policy was hand coded and can balance the bicycle for up to 26K time steps.

We used a similar evaluation procedure as for Cart-pole giving +1 reward for each time step where the bicycle is kept balanced and -1 otherwise. Figure 1(b) compares each approach. The results are similar to those of Cart-pole with RAIL-DW being the top performer. Unif-RAND and Unif-QBC show notably poor performance in this domain. This is because bicycle balancing is a harder learning problem than cart-pole with many more uninformative states (an unrecoverable fall or fallen state). Similarly, 2(b) shows very poor performance for versions of RAIL that use distribution unaware active learners.

Wargus. We consider controlling a group of 5 friendly close-range military units against a group of 5 enemy units in the real-time strategy game Wargus, similar to the setup in (Judah et al., 2010). The objective is to win the battle while minimizing the loss in total health of friendly units. The set of actions available to each friendly unit is to attack any one of the remaining units present in the battle (including other friendly units, which is always a bad choice). In our setup, we allow the learner to control one of the units throughout the battle whereas the other friendly units are controlled by a fixed "reasonably good" policy. This situation

would arise when training the group via coordinate ascent on the performance of individual units. The expert policy corresponds to the same policy used by the other units. Note that poor behavior from even a single unit generally results in a huge loss. Providing full demonstrations in real time in such tactical battles is very difficult for human players and quite time consuming if demonstrations are done in slow motion, which motivates state-based active learning.

We designed 21 battle maps differing in the initial unit positions, using 5 for training and 16 for testing. We average results across five learning trials. Passive learns along the expert's trajectory in each map on all 5 maps considered sequentially according to a random ordering. For RAIL, in each iteration a training map is selected randomly and a query is posed in the chosen map. For CBA, a map is selected randomly and CBA is allowed to play an episode in it, pausing and querying as and when needed. If the episode ends, another map is chosen randomly and CBA continues to learn in it. After each query, the learned policy is tested on the 16 test maps. We use the difference in the total health of friendly and enemy units at the end of the battle as the performance metric (which is positive for a win). We did not run experiments for unif-QBC and unif-RAND because it is difficult to define the space of feasible states over which to sample uniformly.

The results are shown in figure 1(c). We see that although Passive learns quickly for the first 10 queries, it fails to improve further. This shows that the states located in this initial prefix of the expert's trajectory are very useful, but thereafter Passive gets stuck on the uninformative part of the trajectory till its query budget is over. On the other hand, RAIL and CBA continue to improve beyond Passive, with RAIL being slightly better, which indicates that they are able to locate and query more informative states.

NETTalk. We evaluate RAIL on a structured prediction task of stress prediction in the NETtalk data set (Dietterich et al., 2008). Given a word, the goal is to assign one of the five stress labels to each letter of the word in the left to right order. It is straightforward to view structured prediction as imitation learning (see for example Ross and Bagnell, 2010) where at each time step (letter location), the learner has to execute the correct action (i.e. predict correct stress label) given the current state. The state consists of features describing the input (the current letter and its immediate neighbors) and previous L predictions made by the learner (the prediction context). In our experiments, we use $L = 1, 2$ and show results only for $L = 2$ noting that $L = 1$ was qualitatively similar.

We use character accuracy as a measure of performance. Passive learns along the expert's trajectory on each training sequence considered in the order it appears in the training set. Therefore, Passive always learns on the correct context, i.e. previous L characters correctly labeled. RAIL and Unif-QBC can select the best query across the entire training set. CBA, like Passive, considers training sequences in the order they appear in the training set and learns on each sequence by querying at each sequence position. In order to minimize the impact of the ordering of training examples on Passive and CBA, we ran five learning trials for each learner, each with a randomized ordering. We report final performance as the learning curves averaged across all five trials.

Figure 1(d) presents the NETtalk results. The results are qualitatively similar to Cart-Pole, except that Unif-QBC and Unif-RAND do quite well in this domain but not as well as RAIL-DW. Again we see that CBA performs poorly because in all five trials it prematurely stops asking queries, showing its sensitivity to its threshold adjustment mechanism.

Overall Observations. Overall, we can draw a number of conclusions from the experiments. First, RAIL-DW proved to be the most robust and effective active imitation learning approach in all of our domains. Second, the choice of the i.i.d. active learner used for RAIL is important. In particular, performance can be poor when the active learning algorithm does not take density information into account. Using density weighted query-by-committee was effective in all of our domains. Third, we found that CBA is quite sensitive to the threshold adjustment mechanism and we were unable to find an alternative mechanism that works across our domains. Fourth, we showed that a more naive application of i.i.d. active learning in the imitation setting is not effective.

7 Summary

We considered reductions from active imitation learning based on i.i.d. active learning, which allow for advances in the i.i.d. setting to translate to imitation learning. First, we analyzed the label complexity of reductions for both non-stationary and stationary policies, showing that the complexity of active imitation learning can be significantly less than passive learning. Second, we introduced RAIL-DW, a practical variant of the reduction for stationary policies, and showed empirically in four domains that it is highly effective compared to a number of alternatives.

Acknowledgements

The authors acknowledge support of the ONR ATL program N00014-11-1-0105.

References

L. Breiman. Bagging predictors. *Machine learning*, 1996.

S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *JAIR*, 2009.

J.A. Clouse. An introspection approach to querying a trainer. Technical report, University of Massachusetts, 1996.

R. Cohn, M. Maxim, E. Durfee, and S. Singh. Selecting operator queries using expected myopic gain. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2010.

I. Dagan and S.P. Engelson. Committee-based sampling for training probabilistic classifiers. In *ICML*, 1995.

Sanjoy Dasgupta. Two faces of active learning. *Theor. Comput. Sci.*, 2011.

T. Dietterich, G. Hao, and A. Ashenfelter. Gradient tree boosting for training conditional random fields. *JMLR*, 2008.

F. Doshi, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. In *ICML*, 2008.

A. Gil, H. Stern, and Y. Edan. A Cognitive Robot Collaborative Reinforcement Learning Algorithm. *World Academy of Science, Engineering and Technology*, 2009.

D.H. Grollman and O.C. Jenkins. Dogged learning for robots. In *ICRA*, 2007.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 2009.

Kshitij Judah, Saikat Roy, Alan Fern, and Thomas Dietterich. Reinforcement learning via practice and critique advice. In *AAAI*, 2010.

Manuel Lopes, Francisco S. Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Principles of Data Mining and Knowledge Discovery*, 2009.

Andrew McCallum and Kamal Nigam. Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

L. Mihalkova and R. Mooney. Using active relocation to aid reinforcement learning. In *FLAIRS*, 2006.

A.Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

S. Ross and J.A. Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010.

Stephane Ross, Geoffrey Gordon, and J. Andrew (Drew) Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Burr Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2009.

H.S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *COLT*, 1992.

A.P. Shon, D. Verma, and R.P.N. Rao. Active imitation learning. In *AAAI*, 2007.

Umar Syed and Robert Schapire. A reduction from apprenticeship learning to classification. In *NIPS*, 2010.

L. G. Valiant. A theory of the learnable. *Commun. ACM*, 1984.

A Theory of Goal-Oriented MDPs with Dead Ends

Andrey Kolobov **Mausam** **Daniel S. Weld**
{akolobov, mausam, weld}@cs.washington.edu
Dept of Computer Science and Engineering
University of Washington
Seattle, USA, WA-98195

Abstract

Stochastic Shortest Path (SSP) MDPs is a problem class widely studied in AI, especially in probabilistic planning. They describe a wide range of scenarios but make the restrictive assumption that the goal is reachable from any state, i.e., that dead-end states do not exist.Because of this, SSPs are unable to model various scenarios that may have catastrophic events (e.g., an airplane possibly crashing if it flies into a storm). Even though MDP algorithms have been used for solving problems with dead ends, a principled theory of SSP extensions that would allow dead ends, including theoretically sound algorithms for solving such MDPs, has been lacking. In this paper, we propose three new MDP classes that admit dead ends under increasingly weaker assumptions. We present Value Iteration-based as well as the more efficient heuristic search algorithms for optimally solving each class, and explore theoretical relationships between these classes. We also conduct a preliminary empirical study comparing the performance of our algorithms on different MDP classes, especially on scenarios with unavoidable dead ends.

1 Introduction

Stochastic Shortest Path (SSP) MDPs [Bertsekas, 1995] is a class of probabilistic planning problems thoroughly studied in AI. They describe a wide range of scenarios where the objective of the agent is to reach a goal state in the least costly way in expectation from any non-goal state using actions with probabilistic outcomes.

While SSPs are a popular model, they have a serious limitation. They assume that a given MDP has at least one *complete proper policy*, a policy that reaches the goal from any state with 100% probability. Basic algorithms for solving SSP MDPs, such as Value Iteration (VI) [Bellman, 1957], fail to converge if this assumption does not hold. In the meantime, this requirement effectively disallows the existence of *dead ends*, states from which reaching the goal is impossible, and of catastrophic events that lead to these states. Such catastrophic failures are a possiblity to be reckoned with in many real-world planning problems, be it sending a rover on Mars or navigating a robot in a building with staircases. Thus, insisting on the absence of dead ends significantly limits the applicability of SSPs. Moreover, verifying that a given MDP has no dead ends can be nontrivial, further complicating the use of this model.

Researchers have realized that allowing dead ends in goal-oriented MDPs would break some existing methods for solving them [Little and Thiebaux, 2007]. They have also suggested algorithms that are aware of the possible presence of dead-end states [Kolobov et al., 2010] and try to avoid them when computing a policy [Keyder and Geffner, 2008, Bonet and Geffner, 2005]. However, these attempts have lacked a theoretical analysis of how to incorporate dead ends into SSPs in a principled way, and of what the optimization criteria in the presence of dead ends should be. This paper bridges the gap by introducing three new MDP classes with progressively weaker assumptions about the existence of dead ends, analyzing their properties, and presenting optimal VI-like and the more efficient heuristic search algorithms for them.

The first class we present, SSPADE, is a small extension of SSP that has well-defined easily-computable optimal solutions if dead ends are present but are avoidable *provided that the process starts at a known initial state* s_0.

The second and third classes introduced in this paper admit that dead ends may exist and the probability of running into them from the initial state may be positive no matter how hard the agent tries. If the chance of a catastrophic event under any policy is nonzero, a key question is: should we prefer policies that minimize the expected cost of getting to the goal even at the expense of an increased risk of failure, or those that reduce the risk of failure above all else?

The former criterion characterizes scenarios where entering a dead end, while highly undesirable, has a finite "price". For instance, suppose the agent buys an expensive ticket for a concert of a favorite band in another city, but remem-

bers about it only on the day of the event. Getting to the concert venue requires a flight, either by hiring a business jet or by a regular airline with a layover. The first option is very expensive but almost guarantees making the concert on time. The second is much cheaper but, since the concert is so soon, missing the connection, a somewhat probable outcome, means missing the concert. Nonetheless, the cost of missing the concert is only the price of the ticket, so a rational agent would choose to travel with a regular airline. Accordingly, one of the MDP classes we propose, fSSPUDE, assumes that the agent can put a price (penalty) on ending up in a dead end state and wants to compute a policy with the least expected cost (including the possible penalty). While seemingly straightforward, this intuition is tricky to operationalize because of several subtleties. We overcome these subtleties and show how fSSPUDE can be solved with easy modifications to existing SSP algorithms.

In the third MDP class we introduce, iSSPUDE, not only are dead ends unavoidable, but the cost of hitting one is assumed to be infinitely large. Consider, for example, the task of planning an ascent to the top of Mount Everest for a group of human alpinists. Such an ascent is fraught with inherent lethal risks, and to any human, the price of their own life can be taken as infinite. Note the conceptual difficulty with this setting: since every policy reaches an infinite-cost state, the expected cost of any policy is also infinite. This makes SSP's cost-minimization criterion uninformative. Instead, for an undertaking as above, a natural primary objective is to maximize the *probability* of getting to the goal (i.e., to minimize the chance of getting into a lethal accident, a dead-end state). However, of all policies maximizing this chance, we would prefer the least costly one (in expectation). This is exactly the multiobjective criterion we propose for this class of MDPs. Solving iSSPUDE is much more involved than handling the previous two classes, and we introduce two novel algorithms for it.

Intuitively, the objectives of fSSPUDE and iSSPUDE MDPs are related — as fSSPUDE's dead-end penalty gets bigger, the optimal policies of the two classes coincide. We provide a theoretical and an empirical analysis of this insight, showing that solving fSSPUDE yields an optimal policy for iSSPUDE if the dead-end penalty is high enough.

Thus, the paper makes four contributions: (1) three new goal-oriented MDP models that admit the existence of dead-end states; (2) optimal VI and heuristic search algorithms for solving them; (3) theoretical results describing equivalences among problems in these classes; and (4) an empirical evaluation tentatively answering the question: which class should be used when modeling a given scenario involving unavoidable dead ends?

2 Background and Preliminaries

SSP MDPs. In this paper, we extend an MDP class known as the Stochastic Shortest Path (SSP) problems with an optional initial state, defined as tuples of the form $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}, s_0 \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{T}$ is a transition function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ that gives the probability of moving from s_i to s_j by executing a, $\mathcal{C}$ is a map $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that specifies action costs, $\mathcal{G}$ is a set of (absorbing) goal states, and s_0 is an optional start state. For each $g \in G$, $\mathcal{T}(g, a, g) = 1$ and $\mathcal{C}(g, a) = 0$ for all $a \in A$, which forces the agent to stay in g forever while accumulating no reward.

An SSP must also satisfy two conditions: (1) It must have at least one *complete proper policy*, a rule prescribing an action for every state with which an agent can reach a goal state from any state with probability 1. (2) Every *improper* policy must incur the cost of ∞ from all states from which it cannot reach the goal with probability 1.

When the initial state is unknown, solving an SSP MDP means finding a policy whose execution from any state allows an agent to reach a goal state while incurring the least expected cost. We call such a policy *complete optimal*, and denote any complete policy as π. When the initial state is given, we are interested in computing an *optimal (partial) policy rooted at* s_0, i.e., one that reaches the goal in the least costly way from s_0 and is defined for every state it can reach from s_0 (though not necessarily for other states).

To make the notion of policy cost more concrete, we define a *cost function* as a mapping $J : \mathcal{S} \to \mathbb{R} \cup \{\infty\}$ and let random variables S_t and A_t denote, respectively, the state of the process after t time steps and the action selected in that state. Then, the cost function J^π of policy π is

$$J^\pi(s) = \mathbb{E}^\pi_s \left[\sum_{t=0}^{\infty} \mathcal{C}(S_t, A_t) \right] \quad (1)$$

In other words, the cost of a policy π at a state s is the expectation of the total cost the policy incurs if the execution of π is started in s. In turn, every cost function J has a policy π^J that is *J-greedy*, i.e., that satisfies

$$\pi^J(s) = \arg\min_{a \in \mathcal{A}} \left[\mathcal{C}(s, a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') J(s') \right] \quad (2)$$

Optimally solving an SSP MDP means finding a policy that minimizes J^π. Such policies are denoted π^*, and their cost function $J^* = J^{\pi^*}$, called the *optimal cost function*, is defined as $J^* = \min_\pi J^\pi$. J^* also satisfies the following condition, the *Bellman equation*, for all $s \in \mathcal{S}$:

$$J(s) = \min_{a \in \mathcal{A}} \left[\mathcal{C}(s, a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') J(s') \right] \quad (3)$$

Value Iteration for SSP MDPs. The Bellman equation suggests a dynamic programing method of solving SSPs, known as Value Iteration (VI_{SSP}) [Bellman, 1957]. VI_{SSP} starts by initializing state costs with an arbitrary *heuristic cost function* $\hat{J}$. Afterwards, it executes several sweeps of the state space and updates every state during every sweep by using the Bellman equation (3) as an assignment operator, the *Bellman backup operator*. Denoting the cost func-

tion after the i-th sweep as J_i, it can be shown that the sequence $\{J_i\}_{i=1}^{\infty}$ converges to J^*. A complete optimal policy π^* can be derived from J^* via Equation 2 .

Heuristic Search for SSP MDPs. Because it stores and updates the cost function for the entire $\mathcal{S}$, VI$_{SSP}$ can be slow and memory-inefficient even on relatively small SSPs. However, if the initial state s_0 is given we are interested in computing $\pi^*_{s_0}$, an optimal policy from s_0 only, which typically does not visit (and hence does not need to be defined for) all states. This can be done with a family of algorithms based on VI called *heuristic search*. Like VI, these algorithms need to be initialized with a heuristic $\hat{J}$. However, if $\hat{J}$ is *admissible*, i.e., satisfies $\hat{J}(s) \leq J^*(s)$ for all states, then heuristic search algorithms can often compute J^* for the states relevant to reaching the goal from s_0 without updating or even memoizing costs for many of the other states. At an abstract level, the operation of any heuristic search algorithm is represented by the FIND-AND-REVISE framework [Bonet and Geffner, 2003a]. As formalized by FIND-AND-REVISE, any heuristic search algorithm starts with an admissible $\hat{J}$ and explicitly or implicitly maintains the graph of a policy greedy w.r.t. the current J, updating the costs of states *only in this graph* via Bellman backups. Since the initial $\hat{J}$ makes many states look "bad" a-priori, they never end up in the greedy graph and hence never have to be stored or updated. This makes heuristic search algorithms, e.g., LRTDP [Bonet and Geffner, 2003b], work more efficiently than VI and still produce an optimal $\pi^*_{s_0}$.

GSSP and MAXPROB MDPs. Unfortunately, many interesting probabilistic planning scenarios fall outside of the SSP MDP class. One example is MAXPROB MDPs [Kolobov et al., 2011], goal-oriented problems where the objective is to maximize the probability of getting to the goal, not minimize the cost. To discuss MAXPROB, we need the following definition:

Definition For an MDP with a set of goal states $\mathcal{G} \subset \mathcal{S}$, the *goal-probability function* of a policy π, denoted P^π, gives the probability of reaching the goal from any state s. Mathematically, letting $S_t^{\pi_s}$ be a random variable denoting a state the MDP may end up if policy π is executed starting in state s for t time steps,

$$P^\pi(s) = \sum_{t=1}^{\infty} P[S_t^{\pi_s} = g \in \mathcal{G}, S_{t'}^{\pi_s} = s \notin \mathcal{G} \; \forall \, 1 \leq t' < t] \quad (4)$$

Each term in the above summation denotes the probability that, if π is executed starting at s, the MDP ends up in a goal state at step t and not earlier. Once the system enters a goal state, it stays in that goal state forever, so the sum of all such terms is the probability of the system ever entering a goal state under π.

To introduce MDPs with dead ends, we will only need the following informal definition of MAXPROB MDPs. A MAXPROB MDP is a problem that can be derived from any goal-oriented MDP (e.g., an SSP) by disregarding the cost function $\mathcal{C}$ and maximizing the *probability* of reaching the goal instead of the expected cost. More specifically, solving a MAXPROB means finding the optimal goal-probability function from the above definition, one that satisfies $P^*(s) = \arg\max_\pi P^\pi(s)$ for all states. Alternatively, $1 - P^*(s)$ can be interpreted as the smallest probability of running into a dead end from s for any policy. Thus, solving a MAXPROB derived from a goal-oriented MDP by discarding action costs can be viewed as a way to identify *dead ends*:

Definition For a goal-oriented MDP, a *dead-end state* (or dead end, for short) is a state s for which $P^*(s) = 0$.

MAXPROBs, SSPs themselves, and many other MDPs belong to the broader class of Generalized SSP MDPs (GSSPs) [Kolobov et al., 2011]. GSSPs are defined as tuples $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}, s_0 \rangle$ of the same form as SSPs, but relax both of the additional conditions in the SSP definition. In particular, they do not require the existence of a complete proper policy as SSPs do. To state the main results of this paper, we will not need the specifics of GSSP's technical definition, and will only refer to the GSSP properties and algorithms described below.

Value Iteration for GSSP MDPs. In the case of SSPs, VI$_{SSP}$ yields a complete optimal policy for these MDPs independently of the initializing heuristic $\hat{J}$. For a GSSP MDP, such a policy need not exist, so there is no analog of VI$_{SSP}$ that works for all problems in this class. However, for MAXPROB, a subclass of GSSP particularly important to us in this paper, such an algorithm, called VI$_{MP}$, can be designed. Like VI$_{SSP}$, VI$_{MP}$ can be initialized with an arbitrary heuristic function, but instead of the Bellman backup operator it uses its generalized version that we call *Bellman backup with Escaping Traps* (BET) in this paper. BET works by updating the initial heuristic function with Bellman backup, until it arrives at a fixed-point function $P^\times$. For SSPs, Bellman backup has only one fixed point, the optimal P^*, so if we were working with SSPs, we would stop here. However, for GSSPs (and MAXPROB in particular) this is not the case — P^* is only *one of* Bellman backup's fixed points, and the current fixed point $P^\times$ may not be equal to P^*. Crucially, to check whether $P^\times$ is optimal, BET applies the *trap elimination* operator to it, which involves constructing the transition graph that uses actions of *all* policies greedy w.r.t. $P^\times$. If $P^\times \neq P^*$, trap elimination generates a new, non-fixed-point $P^{\times'}$, on which BET again acts with Bellman backup, and so on. The fact that VI$_{MP}$ and FRET, the heuristic search framework for GSSPs considered below, sometimes need to build a greedy transition graph w.r.t. a cost function is important for analyzing the performance of algorithms introduced in this paper (Section 8).

VI$_{MP}$'s main property, whose proof is a straightforward extension of the results in the GSSP paper [Kolobov et al., 2011], is similar to VI$_{SSP}$'s:

Theorem 1. *On MAXPROB MDPs, VI_{MP} converges to the optimal goal-probability function P^* independently of the initializing heuristic function $\hat{J}$.*

Heuristic Search for GSSP MDPs. Although a complete optimal policy does not necessarily exist for a GSSP, one rooted at s_0 always does and can be found by any heuristic search algorithm conforming to an FIND-AND-REVISE analogue for GSSPs, FRET [Kolobov et al., 2011]. Like FIND-AND-REVISE on SSPs, FRET guarantees convergence to $\pi^*_{s_0}$ if the initializing heuristic is admissible.

3 MDPs with Avoidable Dead Ends

All definitions of the SSP class in the literature [Bertsekas, 1995, Bonet and Geffner, 2003a, Kolobov et al., 2011] require that the goal be reachable with 100%-probability from every state in the state space, even when initial state s_0 is known and the objective is to find an optimal policy rooted only at that state. We first extend SSPs to the easiest case — when dead ends exist but can be avoided entirely from s_0.

Definition A Stochastic Shortest Path MDP with Avoidable Dead Ends (SSPADE) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}, s_0 \rangle$ where $\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}$, and s_0 are as in the SSP MDP definition, under the following conditions:

- The initial state s_0 is known.
- There exists at least one proper policy rooted at s_0.
- Every *improper* policy has $J^\pi(s_0) = \infty$.

Solving a SSPADE MDP means finding a policy $\pi^*_{s_0}$ rooted at s_0 that satisfies $\pi^*(s_0) = \arg\min_\pi J^\pi(s_0)$.

SSPADE has only two notable differences from SSP — the former assumes the initial state to be known, and only requires the existence of a partial proper policy. However, even such small departures from the SSP definition prevent some SSP algorithms from working on SSPADE, as we are about to see.

Value Iteration: Even though dead ends may be avoided from s_0 with an optimal policy, they are still present in the state space. Thus, VI_{SSP}, which operates on the entire state space, does not converge on SSPADEs, since the optimal costs for dead ends are infinite. One might think that we may be able to adapt VI_{SSP} to SSPADE by restricting computation to the subset of states reachable from s_0. However, even this is not true, because SSPADE requirements do not preclude dead ends reachable from s_0. Overall, for VI_{SSP} to work on SSPADEs, we need to detect divergence of state cost sequences – an unsolved problem, to our knowledge.

Heuristic Search: Although VI does not terminate for SSPADE, heuristic search algorithms do. This is because:

Theorem 2. $SSPADE \subset GSSP$.

Proof sketch. SSPADE directly satisfies all requirements of the GSSP definition [Kolobov et al., 2011]. □

In fact, we can also show that heuristic search for SSPADE only needs the regular Bellman backup operator (instead of the BET operator). That is, the FIND-AND-REVISE framework applies to SSPADE without modification. In particular, FIND-AND-REVISE starts with admissible state costs, i.e., underestimates of their optimal costs. As FIND-AND-REVISE updates them, costs of dead ends grow without bound, while costs of other states converge to finite values. Thus, dead ends become unattractive and drop out of the greedy policy graph rooted at s_0, leaving FIND-AND-REVISE with an optimal, proper policy.

At the same time, some of the specific heuristic search algorithms for SSP that implement the FIND-AND-REVISE template may fail to terminate on SSPADE. For instance, LAO* and LRTDP may try to update values of dead ends until convergence, which can never be reached. However, each of them can be easily amended to halt on SSPADE problems with an optimal solution. Thus, the existing heuristic search techniques carry over to SSPADE with little adaptation.

4 MDPs with Unavoidable Dead Ends

In this section, our objective is threefold: (1) to motivate the semantics of SSP extensions that admit unavoidable dead ends; (2) to state intuitive policy evaluation criteria and thereby induce the notion of optimal policy for models in which the agent pays finite and infinite penalty for visiting dead ends; (3) to formally define MDP class SSPUDE with subclasses fSSPUDE and iSSPUDE that model such finite- and infinite-penalty scenarios.

As a motivation, consider an *improper* SSP MDP, one that conforms to the SSP definition except for the requirement of proper policy existence, and hence has unavoidable dead ends. In such an MDP, the objective of finding a policy that minimizes the expected cost of reaching the goal becomes ill-defined. It implicitly assumes that for at least one policy, the cost incurred by all of the policy's trajectories is finite; however, this cost is finite only for proper policies, all of whose trajectories terminate at the goal. Thus, all policies in an improper SSP may have an infinite expected cost, making the cost criterion unhelpful for selecting the "best" policy.

We suggest two ways of amending the optimization criterion to account for unavoidable dead ends. The first is to assign a finite positive penalty D for visiting a dead end. The semantics of an improper SSP altered this way would be that the agent pays D when encountering a dead end, and the process stops. However, this straightforward modification to the MDP cannot be directly operationalized, since the set of dead-ends is not known a-priori and needs to be inferred while planning. Moreover, this definition also has a caveat – it may cause non-dead-end states that lie on potential paths to a dead end to have higher costs than dead ends themselves. For instance, imagine a state s whose only action leads with probability $(1 - \epsilon)$ to a dead end,

with probability $\epsilon > 0$ to the goal, and costs $\epsilon(D+1)$. A simple calculation shows that $J^*(s) = D + \epsilon > D$, even though reaching the goal from s *is* possible. Moreover, notice that this semantic paradox cannot be resolved just by increasing the penalty D, because the cost of s will always exceed the dead-end penalty by ϵ.

Therefore, we change the semantics of the finite-penalty model as follows. Whenever the agent reaches *any* state with the expected cost of reaching the goal equaling D or greater, the agent simply pays the penalty D and "gives up", i.e., the process stops. Intuitively, this setting describes scenarios where the agent can put a price on how desirable reaching the goal is. For instance, in the example from the introduction involving a concert in another city, paying the penalty corresponds to deciding not to go to the concert, i.e., foregoing the pleasure the agent would have derived from attending the performance.

The benefit of putting a "cap" on any state's cost as described above is that the cost of a state under any policy becomes finite, formally defined as

$$\textsc{J}\textsc{f}^{\pi}(s) = \min_{\pi} \left\{ D, \mathbb{E}\left[\sum_{t=0}^{\infty} \mathcal{C}(S_t^{\pi_s}, A_t^{\pi_s})\right] \right\} \tag{5}$$

It can be shown that for an improper SSP, there exists an optimal policy π^* [1], one that satisfies

$$\pi^*(s) = \arg\min_{\pi} \textsc{J}\textsc{f}^{\pi}(s) \; \forall \; s \in \mathcal{S} \tag{6}$$

As we show shortly, we can find such a policy using the expected-cost analysis similar to that for ordinary SSP MDPs. The intuitions just described motivate the fSSPUDE MDP class, defined at the end of this section.

The second way of dealing with dead ends we consider in this paper is to view them as truly irrecoverable situations and assign $D = \infty$ for visiting them. As a motivation, recall the example of planning a climb to the top of Mount Everest. Since dead ends here cannot be avoided with certainty and the penalty of visiting them is ∞, comparing policies based on the expected cost of reaching the goal breaks down — they all have an infinite expected cost. Instead, we would like to find a policy that maximizes the probability of reaching the goal and whose expected cost *over the trajectories that reach the goal* is the smallest.

To describe this policy evaluation criterion more precisely, let $S_t^{\pi_s+}$ be a random variable denoting a distribution over states s' for which $P^{\pi}(s') > 0$ and in which the MDP may end up if policy π is executed starting from state s for t steps. That is, $S_t^{\pi_s+}$ differs from the variable $S_t^{\pi_s}$ used previously by considering *only* states from which π can reach the goal. Using the $S_t^{\pi_s+}$ variables, we can mathematically evaluate π with two ordered criteria by defining the cost of a state as an ordered pair

$$\textsc{Ji}^{\pi}(s) = (P^{\pi}(s), [J^{\pi}|P^{\pi}](s)) \tag{7}$$

$$\text{where} \quad [\mathrm{J}^{\pi}|\mathrm{P}^{\pi}](\mathrm{s}) = \mathbb{E}\left[\sum_{\mathrm{t}=0}^{\infty} \mathcal{C}(\mathrm{S}_{\mathrm{t}}^{\pi_s+}, \mathrm{A}_{\mathrm{t}}^{\pi_s})\right] \tag{8}$$

Specifically, we write $\pi(s) \prec \pi'(s)$, meaning π' is preferable to π at s, whenever $\textsc{Ji}^{\pi}(s) \prec \textsc{Ji}^{\pi'}(s)$, i.e., when either $P^{\pi}(s) < P^{\pi'}(s)$, or $P^{\pi}(s) = P^{\pi'}(s)$ and $[J^{\pi}|P^{\pi}](s) > [J^{\pi'}|P^{\pi'}](s)$. Notice that the second criterion is used conditionally, only if two policies are equal in terms of the probability of reaching the goal, since maximizing this probability is the foremost priority. Note also that if $P^{\pi}(s) = P^{\pi'}(s) = 0$, then both $[J^{\pi}|P^{\pi}](s)$ and $[J^{\pi'}|P^{\pi'}](s)$ are ill-defined. However, since that means that neither π nor π' can reach the goal from s, we define $[J^{\pi}|P^{\pi}](s) = [J^{\pi'}|P^{\pi'}](s) = 0$ for such cases, and hence $\textsc{Ji}^{\pi}(s) = \textsc{Ji}^{\pi'}(s)$.

As in the finite-penalty case, we can demonstrate that there exists a policy π^* that is at least as large as all others at all states under the $\prec$-ordering above, and hence optimal, i.e.

$$\pi^*(s) = \arg\max_{\prec\pi} \textsc{Ji}^{\pi}(s) \; \forall \; s \in \mathcal{S} \tag{9}$$

We are now ready to capture the above intuitions in a definition of the SSPUDE MDP class and its subclasses fSSPUDE and iSSPUDE:

Definition An SSP with Unavoidable Dead Ends (SSPUDE) MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}, D, s_0 \rangle$, where $\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}$, and s_0 are as in the SSP MDP definition, $D \in \mathbb{R}^+ \cup \{\infty\}$ is a penalty incurred if a dead-end state is visited. In a SSPUDE MDP, every improper policy must incur an infinite expected cost as defined by Eq. 1 at all states from which it can't reach the goal with probability 1.

If $D < \infty$, the MDP is called an fSSPUDE MDP, and its optimal solution is a policy π^* satisfying $\pi^*(s) = \min_{\pi} \textsc{J}\textsc{f}^{\pi}(s)$ for all $s \in \mathcal{S}$.

If $D = \infty$, the MDP is called an iSSPUDE MDP, and its optimal solution is a policy π^* satisfying $\pi^*(s) = \max_{\prec\pi} \textsc{Ji}^{\pi}(s)$ for all $s \in \mathcal{S}$.

Our iSSPUDE class is related to multi-objective MDPs, which model problems with several competing objectives, e.g., total time, monetary cost, etc. [Chatterjee et al., 2006, Wakuta, 1995]. Their solutions are Pareto-sets of all non-dominated policies. Unfortunately, such solutions are impractical due to high computational requirements. Moreover, maximizing the probability of goal achievement converts the problem into a GSSP and hence cannot be easily included in those models. A related criterion has also been studied in robotics [Koenig and Liu, 2002].

5 The Case of a Finite Penalty

Equation 5 tells us that for an fSSPUDE instance, the cost of any policy at any state is finite. Intuitively, this implies

[1] We implicitly assume that one of the optimal policies is *deterministic Markovian* — a detail we can actually prove but choose to gloss over in this paper for clarity.

that fSSPUDE should be no harder to solve than SSP. This intuition is confirmed by this following result:

Theorem 3. *fSSPUDE = SSP.*

Proof sketch. To show that every fSSPUDE MDP $M_{fSSPUDE}$ can be converted to an SSP MDP, we augment the action set $\mathcal{A}$ of fSSPUDE with a special action a' that causes a transition to a goal state with probability 1 and that costs D. This MDP is an SSP, since reaching the goal with certainty is possible from every state. At the same time, the optimization criteria of fSSPUDE and SSP clearly yield the same set of optimal policies for it.

To demonstrate that every SSP MDP M_{SSP} is also an fSSPUDE MDP, for every M_{SSP} we can construct an equivalent fSSPUDE MDP by setting $D = J^*(s)$. The set of optimal policies of both MDPs will be the same. (Note, however, that the conversion procedure is impractical, since it assumes that we know $J^*(s)$ before solving the MDP.) □

The above conversion from fSSPUDE to SSP immediately suggests solving fSSPUDE with modified versions of standard SSP algorithms, as we describe next.

Value Iteration: Theorem 3 implies that J^{F*}, the optimal cost function of an fSSPUDE MDP, must satisfy the following modified Bellman equation:

$$J(s) = \min \left\{ D, \min_{a \in \mathcal{A}} \left[\mathcal{C}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') J(s') \right] \right\} \quad (10)$$

Moreover, it tells us that π^* of an fSSPUDE must be greedy w.r.t. J^{F*}. Thus, an fSSPUDE can be solved with arbitrarily initialized VI_{SSP} that uses Equation 10 for updates.

Heuristic Search: By the same logic as above, all FIND-AND-REVISE algorithms and their guarantees apply to fSSPUDE MDPs if they use Equation 10 in lieu of Bellman backup. Thus, all heuristic search algorithms for SSP work for fSSPUDE.

We note that, although this theoretical result is new, some existing MDP solvers use Equation 10 implicitly to cope with goal-oriented MDPs that have unavoidable dead ends. One example is the miniGPT package [Bonet and Geffner, 2005]; it allows the user to specify a value D and then uses it to implement Equation 10 in several algorithms including VI_{SSP} and LRTDP.

6 The Case of an Infinite Penalty

In contrast to fSSPUDE MDPs, no existing algorithm can solve iSSPUDE problems either implicitly or explicitly, so all algorithms for tackling these MDPs that we present in this section are completely novel.

6.1 Value Iteration for iSSPUDE MDPs

As for the finite-penalty case, we begin by deriving a Value Iteration-like algorithm for solving iSSPUDE. Finding a policy satisfying Eq. 9 may seem hard, since we are effectively dealing with a multicriterion optimization problem. Note, however, the optimization criteria are, to a certain degree, independent — we can *first* find the set of policies whose probability of reaching the goal from s_0 is optimal, and *then* select from them the policy minimizing the expected cost of goal trajectories. This amounts to finding the optimal goal-probability function P^* first, then computing the optimal cost function $[J^*|P^*]$ conditional on P^*, and finally deriving an optimal policy from $[J^*|P^*]$. We consider these subproblems in order.

Finding P^*. The task of finding, for every state, the highest probability with which the goal can be reached by any policy in a given goal-oriented MDP has been studied before — it is the MAXPROB problem mentioned in the Background section. Solving a goal-oriented MDP according to the MAXPROB criterion means finding P^* that satisfies

$$P^*(s) = 1 \; \forall s \in \mathcal{G} \quad (11)$$
$$P^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s,a,s') P^*(s') \; \forall s \notin \mathcal{G}$$

As already discussed, this P^* can be found by the VI_{MP} algorithm with an arbitrary initializing heuristic.

Finding $[J^*|P^*]$. We could derive optimality equations for calculating $[J^*|P^*]$ from first principles and then develop an algorithm for solving them. However, instead we present a more intuitive approach. Essentially, given P^*, we will build a modification M^{P^*} of the original MDP whose solution is exactly the cost function $[J^*|P^*]$. M^{P^*} will have no dead ends, have only actions greedy w.r.t. P^*, and have a transition function favoring transitions to states with higher probabilities of successfully reaching the goal. Crucially, M^{P^*} will turn out to be an SSP MDP, so we will be able to find $[J^*|P^*]$ with SSPs' familiar machinery.

To construct M^{P^*}, observe that an optimal policy π^* for an iSSPUDE MDP, one whose cost function is $[J^*|P^*]$, must necessarily use only actions greedy w.r.t. P^*, i.e., those maximizing the right-hand side of Eq. 11. For each state s, denote the set of such actions as $\mathcal{A}_s^{P^*}$. We focus on non-dead ends, because for dead ends $[J^*|P^*](s) = 0$, and they will not be part of M^{P^*}. By Eq. 11, for each such s, each $a^* \in \mathcal{A}_s^{P^*}$ satisfies $P^*(s) = \sum_{s' \in \mathcal{S}} T(s,a^*,s') P^*(s')$. Note that this equality expresses the following relationship between event probabilities:

$$P\left(\begin{array}{c}\text{Goal was reached}\\ \text{from } s \text{ via optimal policy}\end{array}\right)$$
$$= \sum_{s' \in \mathcal{S}} P\left(\begin{array}{c}a^* \text{ caused}\\ s \to s' \text{ transition}\end{array} \bigwedge \begin{array}{c}\text{Goal was reached}\\ \text{from } s \text{ via optimal policy}\end{array}\right),$$

or, in a slightly rewritten form,

$$\sum_{s' \in \mathcal{S}} P\left(\begin{array}{c}a^* \text{ caused}\\ s \to s' \text{ transition}\end{array} \;\middle|\; \begin{array}{c}\text{Goal was reached}\\ \text{from } s \text{ via optimal policy}\end{array}\right) = 1,$$

where $P\left(\begin{array}{c} a^* \text{ caused} \\ s \to s' \text{ transition} \end{array} \middle| \begin{array}{c} \text{Goal was reached} \\ \text{from } s \text{ via optimal policy} \end{array}\right) = \frac{T(s,a^*,s')P^*(s')}{P^*(s)}$.

These equations essentially say that if a^* was executed in s and, as a result of following an optimal policy π^* the goal was reached, then with probability $\frac{T(s,a^*,s_1)P^*(s_1)}{P^*(s)}$ action a^* must have caused a transition from s to s_1, with probability $\frac{T(s,a^*,s_2)P^*(s_2)}{P^*(s)}$ it must have caused a transition to s_2, and so on. This means that if we want to find the vector $[J^*|P^*]$ of expected costs of goal-reaching trajectories under π^*, then it is enough to find the optimal cost function of MDP $M^{P^*} = \langle \mathcal{S}^{P^*}, \mathcal{A}^{P^*}, \mathcal{T}^{P^*}, \mathcal{C}^{P^*}, \mathcal{G}^{P^*}, s_0^{P^*} \rangle$, where $\mathcal{G}^{P^*}$ and $s_0^{P^*}$ (if known) are the same as $\mathcal{G}$ and s_0 for the iSSPUDE M that we are trying to solve; $\mathcal{S}^{P^*}$ is the same as $\mathcal{S}$ for M but does not include dead ends, i.e., states s for which $P^*(s) = 0$; $\mathcal{A}^{P^*} = \cup_{s \in \mathcal{S}} \mathcal{A}_s^{P^*}$, i.e., the set of actions consists of all P^*-greedy actions in each state; for each $a^* \in \mathcal{A}_s^{P^*}$, $\mathcal{T}^{P^*}(s, a^*, s') = \frac{\mathcal{T}(s,a^*,s')P^*(s')}{P^*(s)}$, as above, and a^* is "applicable" only in s; and $\mathcal{C}^{P^*}(s, a)$ is the same as $\mathcal{C}$ for M, except it is defined only for $a \in \mathcal{A}^{P^*}$.

As it turns out, we already know how to solve MDPs such as M^{P^*}:

Theorem 4. *For an iSSPUDE MDP M with $P^*(s_0) > 0$, MDP M^{P^*} constructed from M as above is an SSP MDP.*

Proof sketch. Indeed, M^{P^*} is "almost" like the original iSSPUDE MDP, but has at least one proper policy because, by construction, it has no dead ends. □

Now, as we know [Bertsekas, 1995], J^* for the SSP M^{P^*} satisfies $J^*(s) = \min_{a \in \mathcal{A}^{P^*}} \mathcal{C}^{P^*}(s, a) + \sum_{s' \in \mathcal{S}} \mathcal{T}^{P^*}(s, a, s') J^*(s')$. Therefore, by plugging in $\frac{\mathcal{T}(s,a,s')P^*(s')}{P^*(s)}$ in place of $\mathcal{T}^{P^*}(s, a, s')$ and $[J^*|P^*]$ in place of J^*, we can state the following theorem for the original iSSPUDE MDP M:

Theorem 5. *For an iSSPUDE MDP with the optimal goal-probability function P^*, the optimal cost function $[J^*|P^*]$ characterizing the minimum expected cost of trajectories that reach the goal satisfies*

$$[J^*|P^*](s) = 0 \ \forall s \ \textbf{s.t.} \ P^*(s) = 0 \tag{12}$$

$$[J^*|P^*](s) = \min_{a \in \mathcal{A}^{P^*}} \left\{ \mathcal{C}(s, a) + \sum_{s' \in \mathcal{S}} \frac{\mathcal{T}(s, a, s')P^*(s')}{P^*(s)} [J^*|P^*](s') \right\}$$

Putting It All Together. Our construction not only let us derive the optimality equation for $[J^*|P^*]$, but also implies that $[J^*|P^*]$ can be found via VI, as in the case of SSP MDPs [Bertsekas, 1995], over P^*-optimal actions and non-dead-end states. Moreover, since the optimal policy for an SSP MDP is greedy w.r.t. the optimal cost function and solving an iSSPUDE MDP ultimately reduces to solving an SSP, the following important result holds:

Theorem 6. *For every iSSPUDE MDP, there exists a Markovian deterministic policy π^* that can be derived from P^* and $[J^*|P^*]$ for non-dead-end states using*

$$\pi^*(s) = arg \min_{a \in \mathcal{A}^{P^*}} \left\{ \mathcal{C}(s, a) + \sum_{s' \in \mathcal{S}} \frac{\mathcal{T}(s, a, s')P^*(s')}{P^*(s)} [J^*|P^*](s') \right\} \tag{13}$$

Combining optimality equations 11 and 12 for P^* and $[J^*|P^*]$ respectively with Equation 13, we present a VI-based algorithm for solving iSSPUDE MDPs, called IVI (Infinite-penalty Value Iteration) in Algorithm 1.

Input: iSSPUDE MDP M
Output: Optimal policy π^* for non-dead-end states of M

1. Find P^* using arbitrarily initialized VI_{MP}.
2. Find $[J^*|P^*]$ using arbitrarily initialized VI_{SSP} over M^{P^*} with update equations 12

Return π^* derived from P^* and $[J^*|P^*]$ via Equation 13

Algorithm 1: IVI

6.2 Heuristic Search for iSSPUDE MDPs

Input: iSSPUDE MDP M
Output: Optimal policy $\pi^*_{s_0}$ for non-dead-end states of M rooted at s_0

1. Find $P^*_{s_0}$ using FRET initialized with an admissible heuristic $\hat{P} \geq P^*$
2. Find $[J^*|P^*]_{s_0}$ using FIND-AND-REVISE over M^{P^*} with optimality equations 12, initialized with an admissible heuristic $\hat{J} \leq [J^*|P^*]$.

Return $\pi^*_{s_0}$ derived from $P^*_{s_0}$ and $[J^*|P^*]_{s_0}$ via Equation 13

Algorithm 2: SHS

As we established, solving an iSSPUDE MDP with VI is a two-stage process, whose first stage solves a MAXPROB MDP and whose second stage solves an SSP MDP. In the Background section we mentioned that both of these kinds of MDPs can be solved with heuristic search; MAXPROB — with the FRET framework, and SSP — with the FIND-AND-REVISE framework. This allows us to construct a heuristic search schema called SHS (Staged Heuristic Search) for iSSPUDE MDPs, presented in Algorithm 2.

There are two major differences between Algorithms 1 and 2. The first one is that SHS produces functions $P^*_{s_0}$ and $[J^*|P^*]_{s_0}$ that are guaranteed to be optimal only over the states visited by some optimal policy $\pi^*_{s_0}$ starting from the initial state s_0. Accordingly, the SHS-produced policy $\pi^*_{s_0}$ specifies actions only for these states and does not prescribe any for other states. Second, SHS requires two admissible heuristics to find an optimal (partial) policy, one ($\hat{P}$) being an *upper* bound on P^* and the other ($\hat{J}$) being a *lower* bound on $[J^*|P^*]$.

7 Equivalences of Optimization Criteria

The presented algorithms for MDPs with unavoidable dead ends are significantly more complicated than those for MDPs with unavoidable ones. Nonetheless, intuition tells us that for a given tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}, D, s_0 \rangle$, solving it under the infinite-penalty criterion (i.e., as an iSSPUDE) should yield the same policy as solving it under the finite-penalty criterion (i.e., as an fSSPUDE) if in the latter case the penalty D is very large. This can be stated as a theorem:

Theorem 7. *For iSSPUDE and fSSPUDE MDPs over the same domain, there exists the smallest finite penalty D_{thres} s.t. for all $D > D_{thres}$ the set of optimal policies of fSSPUDE (with penalty D) is identical to the set of optimal policies of iSSPUDE.*

Proof sketch. Although the full proof is technical, its main observation is simple — as D increases, it becomes such a large deterrent against hitting a dead end that any policy with a probability of reaching the goal lower than the optimal P^* starts having a higher expected cost of reaching the goal than policies optimal according to iSSPUDE's criterion. □

As a corollary, if we choose $D > D_{thres}$, we can be sure that at any given state s, all optimal (JF*-greedy) policies of the resulting fSSPUDE will have the same probability of reaching the goal, and this probability is $P^*(s)$ according to the infinite-penalty optimization criterion (and therefore will also have the same conditional expected cost $[J^*|P^*]$)

This prompts a question: what can we say about the probability of reaching the goal of JF*-greedy policies if we pick $D \leq D_{thres}$? Unfortunately, in this case different greedy policies may not only be suboptimal in terms of this probability, but even for a fixed D each may have a different, arbitrarily low chance of reaching the goal. For example, consider an MDP with three states, s_0 (the initial state), d (a dead end), and g (a goal). Action a_d leads from s_0 to d with probability 0.5 and to g with probability 0.5 and costs 1 unit. Action a_g leads from s_0 to g also with probability 1, and costs 3 units. Finally, suppose we solve this MDP as an fSSPUDE with $D = 4$. It is easy to see that both policies, $\pi(s_0) = a_d$ and $\pi(s_0) = a_g$, have the same expected cost, 3. However, the former reaches the goal with probability 0.5, while the latter always reaches it. The ultimate reason for this discrepancy is that the policy evaluation criterion of fSSPUDE is oblivious to policy's probability of reaching the goal, and optimizes for this parameter only indirectly, via policy's expected cost.

To summarize, we have two ways of finding an optimal policy in the infinite-penalty case, either by directly solving the corresponding iSSPUDE instance, or by choosing a sufficiently large D and solving the finite-penalty fSSPUDE MDP. We do not know of a principled way to choose D, but it is typically easy to guess by inspecting the MDP. Thus, although the latter method gives no a-priori guarantees, it often yields a correct answer in practice.

8 Experimental Results

The objective of our experiments was to find out the most practically efficient way of finding the optimal policy in the presence of unavoidable dead ends and infinite penalty for visiting them, by solving an iSSPUDE MDP or an fSSPUDE MDP with a large D. To make a fair comparison between these methods, we employ very similar algorithms to handle them. For both classes, the most efficient optimal solution methods are heuristic search techniques, so in our experiments we assume knowledge of the initial state and use only algorithms of this type.

To solve an fSSPUDE, we use the implementation of the LRTDP algorithm, an instance of the FIND-AND-REVISE heuristic search framework for SSPs, available in the miniGPT package [Bonet and Geffner, 2005]. As a source of admissible heuristic state costs/goal-probability values, we choose the maximum of atom-min-forward heuristic [Haslum and Geffner, 2000] and SixthSense [Kolobov et al., 2011]. The sole purpose of the latter is to soundly identify many of the dead ends and assign the value of D to them. (Identifying a state as a dead end may be nontrivial if the state has actions leading to other states.)

Since solving iSSPUDE involves tackling two MDPs, a MAXPROB and an SSP, to instantiate the SHS schema (Algorithm 2) we use two heuristic search algorithms. For the MAXPROB component, we use a specially adapted version [Kolobov et al., 2011] of miniGPT's LRTDP, equipped with SixthSense (note that the atom-min-forward heuristic is cost-based and does not apply to MAXPROB MDPs). For the SSP component, we use miniGTP's LRTDP, as for fSSPUDE, with atom-min-forward; SixthSense is unnecessary because SSP has no dead ends.

Our benchmarks were problems 1 through 6 of the Exploding Blocks World domain from IPPC-2008 [Bryce and Buffet, 2008] and problems 1 through 15 of the Drive domain from IPPC-06 [Buffet and Aberdeen, 2006]. Most problems in both domains have unavoidable dead ends. To set the D penalty for the fSSPUDE model, we examined each problem and tried to come up with an intuitive, easily justifiable value for it. For all problems, $D = 500$ yielded a policy that was optimal under both the finite-penalty and infinite-penalty criterion.

Solving the fSSPUDE with $D = 500$ and iSSPUDE versions of each problem with the above implementations yielded the same qualitative outcome on all benchmarks. In terms of speed, solving fSSPUDE was at least an order of magnitude faster than solving iSSPUDE. The difference in used memory was occasionally smaller, but only because both algorithms visited nearly the entire state space reachable from s_0 on some problems. Moreover, in terms of memory as well as speed the difference between solving fSSPUDE and iSSPUDE was the largest (that is, solving iSSPUDE was comparatively the *least* efficient) when the given MDP had $P^*_{s_0}(s) = 1$, i.e. the MDP had no dead ends at all or had only avoidable ones.

Although seemingly surprising, these performance patterns have a fundamental reason. Recall that FRET algorithms, used for solving the MAXPROB part of an iSSPUDE, use the BET operator. BET, for every encountered fixed point $P^\times$ of the Bellman backup operator needs to traverse the transition graph involving all actions greedy w.r.t. $P^\times$, starting from s_0. Also, FRET needs to be initialized with an admissible heuristic, in our experiments – SixthSense, which assigns the value of 0 to states it believes to be dead ends and 1 to the rest.

Now, consider how FRET operates on a MAXPROB corresponding to an iSSPUDE instance that does not have any dead ends, i.e. on the kind of iSSPUDE MDPs that, as our experiments show, is most problematic. For such a MAXPROB, there exists only one admissible heuristic function, $\hat{P}(s) = 1$ for all s, because $P^*(s) = 1$ for all s and an admissible $\hat{P}$ needs to satisfy $\hat{P}(s) \geq P^*(s)$ everywhere. Thus, the heuristic FRET starts with is actually the optimal goal-probability function, and as a consequence, is a fixed point of the Bellman backup operator. Therefore, to conclude that $\hat{P}$ is optimal, FRET needs to build its greedy transition graph. Observe, however, that since $\hat{P}$ is 1 everywhere, this transition graph includes every state reachable from s_0, and uses every action in the MDP! Building and traversing it is very expensive.

The same performance bottleneck, although to a lesser extent, can also be observed on iSSPUDE instances that do have unavoidable dead ends. Building large transition graphs significantly slows down FRET (and hence, SHS) even when P^* is far from being 1 everywhere.

The above reasoning may explain why solving iSSPUDE is slow, but by itself does not explain why solving fSSPUDE is fast in comparison. For instance, we might expect the performance of FIND-AND-REVISE algorithms on fSSPUDE to suffer in the following situations. Suppose state s is a dead end not avoidable from s_0 by any policy. This means that $J^*(s) = D$ under the finite-penalty optimization criterion, and that s is reachable from s_0 by any optimal policy. Thus, FIND-AND-REVISE will halt no earlier than the cost of s under the current cost function reaches D. Moreover, suppose that the heuristic $\hat{J}$ initializes the cost of s to 0 — this is one of the possible admissible costs for s. Finally, assume that all actions in s lead back to s with probability 1 and cost 1 unit. In such a situation, an FIND-AND-REVISE algorithm will need to update the cost of s D times before convergence. Clearly, this will make the performance of FIND-AND-REVISE very bad if the chosen value of D is very large. This raises the question: was solving fSSPUDE in the above experiments so much more efficient than solving iSSPUDE due to our choice of (a rather small) value for D?

To dispel these concerns, we solved fSSPUDE instances of the aforementioned benchmarks with $D = 5 \cdot 10^8$ instead of 500. On all of the 21 problems, the increase in speed compared to the case of fSSPUDE with $D = 500$ was no more than a factor of 1.5. The reason for such a small discrepancy is the fact that, at least on our benchmarks, FIND-AND-REVISE almost never runs into the pathological case described above thanks to the atom-min-forward and SixthSense heuristics. They identify the majority of dead ends encountered by LRTDP and immediately set their costs to D. Thus, instead of spending many updates on such states, LRTDP gets their optimal costs in just one step. To test this explanation, we disabled these heuristics and assigned the cost of 0 to all states at initialization. As predicted, the solution time of the fSSPUDE instances skyrocketed by orders of magnitude.

The presented results appear to imply an unsatisfying fact — on iSSPUDE MDPs that are SSPs, the presented algorithms for solving iSSPUDE are not nearly as efficient as algorithmic schema for SSPs, such as FIND-AND-REVISE. The caveat, however, is that the price SSP algorithms pay for efficiency is *assuming* the existence of proper solutions. iSSPUDE algorithms, on the other hand, implicitly *prove* the existence of such a solution, and are therefore theoretically more robust.

9 Conclusion

A significant limitation of SSP MDPs is their inability to model dead-end states, consequences of catastrophic action outcomes that make reaching the goal impossible. While attempts to incorporate dead ends into SSP have been made before, a principled theory of goal-oriented MDPs with dead-end states has been lacking.

In this paper, we present new general MDP classes that subsume SSP and make increasingly weaker assumptions about the presence of dead ends. SSPADE assumes that dead ends are present but an agent can avoid them if it acts optimally from the initial state. fSSPUDE admits unavoidable dead ends but expects that an agent can put a finite price on running into a dead end. iSSPUDE MDPs model scenarios in which entering a dead end carries an infinite penalty and is to be avoided at all costs.

For these MDP classes, we present VI-based and heuristic search algorithms. We also study the conditions under which they have equivalent solutions. Our empirical results show that, in practice, solving fSSPUDE is much more efficient and yields the same optimal policies as iSSPUDE.

In the future, we hope to answer the question: are iSSPUDE MDPs fundamentally harder than fSSPUDE, or can we invent more efficient heuristic search algorithms for them? Besides, as we found out after submitting this article, a more general class of MDPs than iSSPUDE, called S^3P, has been proposed in the literature but not completely solved [Teichteil-Königsbuch, 2012]. Deriving algorithms for it will be our next research objective.

Acknowledgments. This work has been supported by NSF grant IIS-1016465, ONR grant N00014-12-1-0211, and the UW WRF/TJ Cable Professorship.

References

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

[Bertsekas, 1995] Bertsekas, D. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific.

[Bonet and Geffner, 2003a] Bonet, B. and Geffner, H. (2003a). Faster heuristic search algorithms for planning with uncertainty and full feedback. In *IJCAI*, pages 1233–1238.

[Bonet and Geffner, 2003b] Bonet, B. and Geffner, H. (2003b). Labeled RTDP: Improving the convergence of real-time dynamic programming. In *ICAPS'03*, pages 12–21.

[Bonet and Geffner, 2005] Bonet, B. and Geffner, H. (2005). mGPT: A probabilistic planner based on heuristic search. *Journal of Artificial Intelligence Research*, 24:933–944.

[Bryce and Buffet, 2008] Bryce, D. and Buffet, O. (2008). International planning competition, uncertainty part: Benchmarks and results. In *http://ippc-2008.loria.fr/wiki/images/0/03/Results.pdf*.

[Buffet and Aberdeen, 2006] Buffet, O. and Aberdeen, D. (2006). The factored policy gradient planner (ipc-06 version). In *Fifth International Planning Competition at ICAPS'06*.

[Chatterjee et al., 2006] Chatterjee, K., Majumdar, R., and Henzinger, T. A. (2006). Markov decision processes with multiple objectives. In *Proceedings of Twenty-third Annual Symposium on Theoretical Aspects of Computer Science*, pages 325–336.

[Haslum and Geffner, 2000] Haslum, P. and Geffner, H. (2000). Admissible heuristic for optimal planning. In *AIPS*, page 140149.

[Keyder and Geffner, 2008] Keyder, E. and Geffner, H. (2008). The HMDPP planner for planning with probabilities. In *Sixth International Planning Competition at ICAPS'08*.

[Koenig and Liu, 2002] Koenig, S. and Liu, Y. (2002). The interaction of representations and planning objectives for decision-theoretic planning tasks. In *Journal of Experimental and Theoretical Artificial Intelligence*, volume 14, pages 303–326.

[Kolobov et al., 2010] Kolobov, A., Mausam, and Weld, D. (2010). SixthSense: Fast and reliable recognition of dead ends in MDPs. In *AAAI'10*.

[Kolobov et al., 2011] Kolobov, A., Mausam, Weld, D., and Geffner, H. (2011). Heuristic search for Generalized Stochastic Shortest Path MDPs. In *ICAPS'11*.

[Little and Thiebaux, 2007] Little, I. and Thiebaux, S. (2007). Probabilistic planning vs. replanning. In *ICAPS Workshop on IPC: Past, Present and Future*.

[Teichteil-Königsbuch, 2012] Teichteil-Königsbuch, F. (2012). Stochastic safest and shortest path problems. In *AAAI'12*.

[Wakuta, 1995] Wakuta, K. (1995). Vector-valued Markov decisionprocesses and the systems of linear inequalities. *Stochastic Processes and their Applications*, 56:159–169.

Dynamic Stochastic Orienteering Problems for Risk-Aware Applications*

Hoong Chuin Lau, William Yeoh, Pradeep Varakantham, Duc Thien Nguyen, Huaxing Chen
Living Analytics Research Centre
Singapore Management University
Singapore 679668
{hclau,williamyeoh,pradeepv,dtnguyen,hxchen}@smu.edu.sg

Abstract

Orienteering problems (OPs) are a variant of the well-known prize-collecting traveling salesman problem, where the salesman needs to choose a subset of cities to visit within a given deadline. OPs and their extensions with stochastic travel times (SOPs) have been used to model vehicle routing problems and tourist trip design problems. However, they suffer from two limitations – travel times between cities are assumed to be time independent and the route provided is independent of the risk preference (with respect to violating the deadline) of the user. To address these issues, we make the following contributions: We introduce (1) a dynamic SOP (DSOP) model, which is an extension of SOPs with dynamic (time-dependent) travel times; (2) a risk-sensitive criterion to allow for different risk preferences; and (3) a local search algorithm to solve DSOPs with this risk-sensitive criterion. We evaluated our algorithms on a real-world dataset for a theme park navigation problem as well as synthetic datasets employed in the literature.

1 Introduction

An orienteering problem (OP) is a planning problem where the goal is to find a sequence of vertices in a graph that maximizes the sum of rewards from those vertices subject to the constraint that the sum of edge lengths along that sequence is no larger than a threshold [34]. It was motivated by competitive orienteering sports where competitors start and end at specified locations and try to accumulate as much reward as possible from visiting checkpoints within a given deadline. Aside from this problem, researchers have also used OPs to model other problems like vehicle routing [11], production scheduling [2] and, more recently, tourist trip design problems [35].

One of the limitations of OPs is that it assumes that edge lengths are deterministic, which can be an inaccurate assumption in applications with uncertainty. Using vehicle routing problems as an example, the travel time between cities, which are modeled with edge lengths, depends on road congestion and is thus not deterministic. Thus, researchers have extended OPs to Stochastic OPs (SOPs), where edge lengths are now random variables that follow a given distribution and the goal is now to find a sequence that maximizes the sum of expected utilities from vertices in the sequence [6].

Although SOPs more accurately model applications with uncertainty, there are two assumptions that limit its applicability. Firstly, the assumption that the distribution of the random variables is time independent can still be inaccurate. Using the vehicle routing problem again as an example, the level of congestion of a road can change throughout the day – roads are more likely to be congested during commuting hours and less likely to be congested in the middle of the night. Thus, the distribution of travel times on the road should also change throughout the day. Therefore, in this paper, we extend SOPs to Dynamic SOPs (DSOPs).

Secondly, the expected utilities do not capture the risk preference of the user (with respect to violating the deadline) who will be employing the solution. In other words, the solution returned for a user with a hard deadline is the same as the one for a user with a soft deadline. Intuitively, the solution for the user with a soft deadline should be longer, and thus more rewarding, but riskier than the solution for the user with a hard deadline. To address this issue, we intro-

*This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. We also thank Ajay S. Aravamudhan for his help with our experimental evaluations.

duce a risk-sensitive criterion that captures this risk preference. To illustrate the applicability of DSOPs, we evaluated our algorithms on a real-world dataset for a theme park navigation problem as well as synthetic datasets employed in the literature.

2 Motivating Domain

Ubiquitous computing has made substantial progress in recent years, particularly fueled by the increased prevalence of "smart" devices like smart phones. The widespread use of such devices presents a unique opportunity for the delivery of real-time contextualized and personalized information to users. For instance, operators of theme parks have now deployed smart phone applications that allow users to access real-time information like current queueing times at various attractions so that they can better plan their itinerary in the park.[1] Unfortunately, access to current queueing times only allows users to plan myopically, that is, to choose which attraction to go to next. If users also had access to predicted future queueing times, which park operators should be able to provide based on historical data, then the users might be able to make longer term plans that take that information into account. Our research is motivated by this exact problem, where a visitor to a theme park likes to choose (or be recommended) a sequence of attractions that optimizes his/her visitor experience subject to the constraint that the total travel and queueing time in the park is no larger than a threshold. In this paper, we conduct experiments on a real-world dataset for this problem.

3 Background

We now provide a brief description of OPs and SOPs.

3.1 OPs

The *orienteering problem* (OP) [34] is defined by a tuple $\langle G, T, R, v_1, v_n, H\rangle$. $G = \langle V, E\rangle$ is a graph with sets of vertices V and edges E; $T : v_i \times v_j \to \mathbb{R}^+ \cup \{0, \infty\}$ specifies a finite non-negative travel time between vertices v_i and v_j if $e_{ij} \in E$ and ∞ otherwise; and $R : v_i \to \mathbb{R}^+ \cup \{0\}$ specifies a finite non-negative reward for each vertex $v_i \in V$. A solution in an OP is a Hamiltonian path over a subset of vertices including start vertex v_1 and end vertex v_n and whose total travel time is no larger than H. Solving OPs optimally means finding a solution that maximizes the sum of rewards of vertices in its path. Researchers have shown that solving OPs optimally is NP-hard [11]. In this paper, we assume that the end vertex can be any arbitrary vertex.

A simplified version of our motivating theme park navigation problem, where travel and queueing times are deterministic and static, can be modeled as an OP: v_1 corresponds to the entrance of the park, v_n corresponds to the exit of the park, other vertices v_i correspond to attractions in the park, and travel times $T(v_i, v_j)$ correspond to the sum of the travel time between attractions v_i and v_j and the queueing time at attraction v_j.

The start and end vertices in OPs are typically distinct vertices. In the special case where they are the same vertex, the problem is called a *orienteering tour problem* (OTP) [27]. The difference between both formulations is small. It is always possible to add a dummy edge with zero travel time between the start and end vertices to convert an OP to an OTP.

Researchers have proposed several exact branch-and-bound methods to solve OPs [18] including optimizations with cutting plane methods [20, 9]. However, since OPs are NP-hard, exact algorithms often suffer from scalability issues. Thus, constant-factor approximation algorithms [4] are necessary for scalability. Researchers also proposed a wide variety of heuristics to address this issue including sampling-based algorithms [34], local search algorithms [11, 7], neural network-based algorithms [38] and genetic algorithms [32]. More recently, Schilde et al. developed an ant colony optimization algorithm to solve a bi-objective variant of OPs [29].

3.2 Stochastic OPs

The assumption of deterministic travel times is unrealistic in many real-world settings. Using our motivating theme park navigation problem as an example, the travel time of a patron depends on factors like fatigue. Thus, researchers have extended OPs to *Stochastic OPs* (SOPs) [6], where travel times are now random variables that follow a given distribution, and the goal is to find a path that maximizes the sum of expected utilities from vertices in the path. The random variables are assumed to be independent of each other. The expected utility of a vertex is the difference between the expected reward and expected penalty of the vertex. The expected reward (or penalty) of a vertex is the reward (or penalty) of the vertex times the probability that the travel time along the path thus far is no larger (or larger) than H. More formally, the expected utility $U(v_i)$ of a vertex v_i is

$$U(v_i) = P(a_i \leq H)\, R(v_i) - P(a_i > H)\, C(v_i)$$

where the random variable a_i is the arrival time at vertex v_i (that is, the travel time from v_1 to v_i), $R(v_i)$

[1] http://disneyparksmobile.com/ is one such example.

is the reward of arriving at vertex v_i before or at H and $C(v_i)$ is the penalty of arriving at vertex v_i after H. Campbell et al. have extended OP algorithms to solve SOPs including an exact branch-and-bound method and a local search method based on variable neighborhood search [6]. Gupta et al. introduced a constant-factor approximation algorithm for a special case of SOPs, where there is no penalty for arriving at a vertex after H [12].

4 Dynamic Stochastic OPs

Stochastic OPs (SOPs) assume independence of travel time distributions across different edges. However, in many problems, there is a considerable dependence of travel times on the arrival time at a vertex. Using our motivating theme park navigation problem again as an example, the travel time of a user depends on factors like fatigue, and the level of fatigue of a patron increases as they spend more time in the park.

To capture dependencies between travel time distributions, we introduce an extension to SOPs called *Dynamic SOPs* (DSOPs). The key difference from SOPs is that the travel time distribution in DSOPs for moving from vertex v_i to v_j is dependent on the arrival time a_i at vertex v_i. In this paper, we will assume $T_{i,j}$ to be a discrete set of distributions, where each element of the set corresponds to a range of values for a_i. Therefore, the travel time distribution for an arrival time of a_i is represented as $T_{i,j}^{a_i}$ and hence the probability that travel time is m is given by $T_{i,j}^{a_i}(m)$.

4.1 Risk-Sensitive Criterion

While expected utility is a good metric in general, the approach by [6] suffers from many limitations. Firstly, it is a point estimate solution which does not consider the "risk" profile of the user. By "risk", we do not refer to the term used in a financial sense, but rather the level of conservativeness measured in terms of the probability of completing the path within the deadline. In other words, a risk-seeking user will be prepared to choose a sequence of attractions that have a large utility, but with a higher probability of not completing the path within the deadline, compared to a risk-averse user who might choose a more "relaxed" path with lower utility. Secondly, the underlying measurement of expected utility is not intuitive in the sense that a utility value accrued at each attraction does not usually depend on the probability that the user arrives at the attraction by a certain time; but rather, the utility is accrued when the attraction is visited, and the user is concerned with visiting all the attractions (i.e. sum of utilities) within a certain time threshold.

Given the above consideration, we are interested in the problem that allows the user to tradeoff between the level of conservativeness (or risk) against the total utility. More precisely, given a value $0 \leq \epsilon \leq 1$, we are interested in obtaining a path, where the probability of completing the entire path within a deadline H is at least $1 - \epsilon$. Or more precisely,

$$P(a_n \leq H) \geq 1 - \epsilon \tag{1}$$

where a_n is the arrival time at the last vertex of the path. The objective value is therefore inversely proportional to the value of ϵ.

5 Completion Probability Approximations

In this section, we describe two ways of approximating the completion probability $P(a_n \leq H)$, which is used in Equation 1. Given the order $\pi = \langle v_1, v_2, \ldots, v_k, v_n \rangle$, we can use the following expression to compute the completion probability:

$$\begin{aligned} P(a_n \leq H) = & \\ & \int_{a_n=0}^{a_n \leq H} \int_{a_k=0}^{a_k=a_n} \int_{a_{k-1}=0}^{a_{k-1}=a_k} \cdots \int_{a_1=0}^{a_1=a_2} \\ & T_{k,n}^{a_k}(a_n - a_k)\, T_{k-1,k}^{a_{k-1}}(a_k - a_{k-1}) \cdots T_{1,2}^{a_1}(a_2 - a_1) \\ & d(a_1)\, d(a_2) \ldots d(a_k)\, d(a_n) \end{aligned} \tag{2}$$

where a_n is the arrival time at the exit node, and we capture the dependencies on arrival times at each of the vertices by reducing the range of feasible arrival times (for the integrals) based on the previous activities in the order of vertices. Unfortunately, the computation of the expression is expensive since the integrals have to be computed sequentially. To provide an intuition for the time complexity, computing triple integrals take around 30 minutes with an exponential distribution (most scalable of all distributions with integration) on our machine using the Matlab software. To address this issue of scalability, we introduce two approximation approaches, a sampling-based approach and a matrix-based approach.

5.1 Sampling-based Approach

One can approximate the completion probability $P(a_n \leq H)$ of a path by randomly sampling the travel time distributions for each edge along the path, and checking if the arrival time a_n at the last vertex exceeds H. For example, assume that we want to compute $P(a_n \leq H)$ for the path $\pi = \langle v_1, v_2, \ldots, v_k, v_n \rangle$. Using the starting time a_1, we generate a travel time sample from the distribution $T_{1,2}^{a_1}$ to represent the travel time from vertex v_1 to vertex v_2, which is also the arrival time a_2 at vertex v_2. We then generate a

travel time sample from the distribution $T_{2,3}^{a_2}$ to represent the travel time from vertex v_2 to vertex v_3. The arrival time a_3 at vertex v_3 is thus the sum of both travel times. We continue this process until we generate a travel time sample to represent the travel time from vertex v_k to vertex v_n, and the arrival time a_n is thus the sum of all travel times. We count this entire process as a single sample. We can then approximate

$$P(a_n \leq H) \approx \hat{P}(a_n \leq H) = \frac{N^+}{N} \tag{3}$$

where N^+ is the number of samples whose arrival time $a_n \leq H$ is no larger than the deadline H and N is the total number of samples. Unfortunately, this approach does not provide any theoretical guarantees on whether Equation 1 is truly satisfied. However, as we increase the number of samples, the approximation for the actual distribution becomes tighter.

5.2 Matrix-based Approach

Alternatively, one can exploit the fact that the dependencies are primarily due to arrival time at a vertex and not on the entire order of vertices before the current vertex. At a higher level, it implies that the underlying problem is Markovian and hence we can decompose the expression of Equation 2. We also make conservative estimates of the probability such that we can provide theoretical guarantees on whether Equation 1 is truly satisfied.

The key ideas here are (1) to divide the possible arrival times a_i at vertex v_i into a finite number of ranges $\mathbf{r}_{i,1}, \mathbf{r}_{i,2}, \ldots, \mathbf{r}_{i,k}$, where $\mathbf{r}_{i,j}$ is the j-th range of arrival time at vertex v_i and (2) to pre-compute for all pairs of vertices v_i and v_j a conservative estimate $\hat{P}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ of the probability $P(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ of transitioning between ranges of arrival times $\mathbf{r}_{i,p}$ and $\mathbf{r}_{j,q}$. Thus, we can now decompose the expression of Equation 2 to an expression that exploits the Markovian property along with ranges of arrival times:

$$\begin{aligned} P(a_n \leq H) = & \\ & \sum_i P(a_1 \in \mathbf{r}_{1,i}) \cdot \sum_j P(a_2 \in \mathbf{r}_{2,j} | a_1 \in \mathbf{r}_{1,i}) \\ & \cdots \sum_y P(a_k \in \mathbf{r}_{k,y} | a_{k-1} \in \mathbf{r}_{k-1,x}) \\ & \cdot \sum_z P(a_n \in \mathbf{r}_{n,z} | a_k \in \mathbf{r}_{k,y}) \end{aligned}$$

and conservatively approximate it by

$$\begin{aligned} \hat{P}(a_n \leq H) = & \\ & \sum_i \hat{P}(a_1 \in \mathbf{r}_{1,i}) \cdot \sum_j \hat{P}(a_2 \in \mathbf{r}_{2,j} | a_1 \in \mathbf{r}_{1,i}) \\ & \cdots \sum_y \hat{P}(a_k \in \mathbf{r}_{k,y} | a_{k-1} \in \mathbf{r}_{k-1,x}) \\ & \cdot \sum_z \hat{P}(a_n \in \mathbf{r}_{n,z} | a_k \in \mathbf{r}_{k,y}) \end{aligned}$$

It is clear that $\hat{P}(a_n \leq H) \leq P(a_n \leq H)$ is a conservative estimate if $\hat{P}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p}) \leq P(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ are all conservative estimates. $\hat{P}(a_1 \in \mathbf{r}_{1,i})$ depends on the starting time at vertex v_1, which is provided as an input. We now describe how to compute the other probabilities $\hat{P}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$. If the range $\mathbf{r}_{i,p}$ contains only a single point a_i, then

$$\begin{aligned} P(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p}) =& P(a_j \in \mathbf{r}_{j,q} | a_i) \\ =& \int_{a_j \in \mathbf{r}_{j,q}} T_{i,j}^{a_i}(a_j - a_i)\ d(a_j) \end{aligned} \tag{4}$$

However, the realization of the random variable a_i only occurs at runtime, and computing the integral in Equation 4 at runtime is expensive. Thus, we would like to compute a conservative estimate $\hat{P}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ of probability $P(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ for all possible realizations of $a_i \in \mathbf{r}_{i,p}$. We thus compute them as follows:

$$\begin{aligned} \hat{P}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p}) = & \\ & \min_{a_i \in \mathbf{r}_{i,p}} \int_{a_j \in \mathbf{r}_{j,q}} T_{i,j}^{a_i}(a_j - a_i)\ d(a_j) \end{aligned} \tag{5}$$

The value in the integral is the probability of the arrival time a_j to be in the range $\mathbf{r}_{j,q}$ for a given value of a_i. Thus, by taking the minimum of these probabilities over all possible values of a_i in the range $\mathbf{r}_{i,p}$, the conditional probability $\hat{P}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p}) \leq P(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ is a conservative estimate of the true probability.

Once all the probabilities are pre-computed, they form transition matrices

$$\mathbf{P}_{i,j} = \begin{pmatrix} \hat{P}(a_j \in \mathbf{r}_{j,1}) | a_i \in \mathbf{r}_{i,1}) & \hat{P}(a_j \in \mathbf{r}_{j,2}) | a_i \in \mathbf{r}_{i,1}) & \cdots \\ \hat{P}(a_j \in \mathbf{r}_{j,1}) | a_i \in \mathbf{r}_{i,2}) & \hat{P}(a_j \in \mathbf{r}_{j,2}) | a_i \in \mathbf{r}_{i,2}) & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix} \tag{6}$$

which represent the transition probabilities from vertices v_i to v_j. Finally, to compute $\hat{P}(a_n \leq H)$, we compute the multiplication of matrices $\mathbf{P}_1 \cdot \mathbf{P}_{1,2} \cdot \mathbf{P}_{2,3} \cdots \mathbf{P}_{k-1,k} \cdot \mathbf{P}_{k,n}$ and in the resultant matrix, sum up all the probabilities for ranges of arrival times a_n that are less than or equal to the deadline H.

Algorithm 1: Local Search Algorithm

```
   /* Generate Initial Solution                                   */
1  currentPath = ConstructionHeuristic()
   /* Make Local Improvements                                     */
2  bestPath = currentPath
3  numIterNoImprove = 0
4  currentMetric = random metric
5  T = starting temperature
6  for iterations = 1 to maxIterations do
7  |  T = T · ΔT
8  |  Z = numIterNoImprove / (2 · maxIterNoImprove)
   |  /* Perform 2-Opt Operation on currentPath                  */
9  |  currentPath = 2-Opt(currentPath)
   |  /* Remove Vertices from currentPath                        */
10 |  while currentPath is infeasible OR rand() ≤ Z do
11 |  |  remove the second last vertex from currentPath
12 |  end
   |  /* Insert Vertices to currentPath                          */
13 |  neighborPath = Insert(currentPath, currentMetric)
   |  /* Update currentPath and bestPath                         */
14 |  ΔR = neighborPath.reward − currentPath.reward
15 |  if ΔR > 0 OR rand() ≤ e^(ΔR/T) then
16 |  |  currentPath = neighborPath
17 |  end
18 |  if currentPath.reward > bestPath.reward then
19 |  |  bestPath = currentPath
20 |  |  numIterNoImprove = 0
21 |  else
22 |  |  numIterNoImprove = numIterNoImprove + 1
23 |  |  if numIterNoImprove > maxIterNoImprove then
24 |  |  |  currentMetric = new random metric
25 |  |  |  numIterNoImprove = 0
26 |  |  end
27 |  end
28 end
29 return bestPath
```

6 DSOP Algorithms

In this section, we describe a branch-and-bound algorithm and a local search algorithm that solves DSOPs.

6.1 Branch-and-Bound Algorithm

We provide a depth-first branch-and-bound algorithm, where the root of the search tree is the start vertex and the children of a vertex are all the unvisited vertices minus the exit vertex. The branch of an arbitrary vertex thus represents the path from the start vertex to that vertex. The value of a vertex is the sum of rewards of all vertices along its branch. The algorithm prunes the subtree of a vertex if it fails to satisfy our risk-sensitive criterion. For example, assume that a vertex v_k is on the branch $\pi = \langle v_1, v_2, \ldots, v_k \rangle$, where vertex v_i is on the i-th position on the branch. The algorithm prunes the subtree rooted at vertex v_k if the condition in Equation 1 is not satisfied if one appends the exit vertex to the end of the path. The algorithm returns the vertex with the largest value and the branch of that vertex with the exit vertex appended at the end of the path as the best solution that satisfies the risk-sensitive criterion.

6.2 Local Search Algorithm

Unfortunately, the branch-and-bound algorithm suffers from scalability issues as the size of the search tree is exponential in the number of vertices in the graph. We thus introduce a local search algorithm that is based on the standard two-phase approach – a construction heuristic to generate an initial solution followed by local improvements on that solution.

6.2.1 Construction Heuristic

The construction heuristic is a greedy insertion algorithm that greedily inserts the best unvisited vertex at the best position in the current path according to a given metric. The algorithm begins with the path that starts at the start vertex and immediately exits at exit vertex, and it terminates when it can no longer insert any attraction at any position without violating the condition in Equation 1.

In this paper, we use the following metric to evaluate the value of inserting vertex v_i at position p: $\frac{\Delta R}{1+\Delta P}$, where ΔR and ΔP is the gain in reward and probability, respectively, for inserting vertex v_i at position p. Thus, $\Delta R = R(v_i)$, which is the reward of vertex v_i, and $\Delta P = \hat{P}(a_n \leq H) - \hat{P}'(a_n \leq H)$, where $\hat{P}'(a_n \leq H)$ and $\hat{P}(a_n \leq H)$ is the probability of arriving at the exit vertex before and after insertion, respectively. We use the same approach of multiplying transition matrices described in Section 6.1 to speed up the computation of the probabilities. Finally, we add 1 to the gain in probabilities such that the denominator is greater than 0.

This metric is motivated by similar metrics in knapsack problems, namely the utility of an item is the ratio between the reward and size of that item [23]. We have also tried 4 other variants of the above metric, namely (1) $\frac{1}{1+\Delta P}$, (2) ΔR, (3) $\frac{(\Delta R)^2}{1+\Delta P}$, (4) $\frac{\Delta R}{\sqrt{1+\Delta P}}$, where we ignored the effects of rewards in (1) and probabilities in (2), and we amplified the effects of rewards in (3) and probabilities in (4). However, our chosen metric was shown to outperform these 4 variants empirically.

6.2.2 Local Improvements

We use a hybrid approach that consists of a variable neighborhood search combined with simulated annealing to locally improve our initial solution found by the construction heuristic. Algorithm 1 shows the pseudocode of this algorithm. After constructing the ini-

tial solution (line 1), the algorithm iteratively runs the following four phases until the maximum number of iterations is reached (line 6):

Phase 1: If the path contains at least two vertices (not including the start and end vertices), then the algorithm performs a 2-Opt operation, that is, it randomly swaps two of these vertices (line 9).

Phase 2: If the path is not feasible, that is, it does not satisfy Equation 1, then the algorithm repeatedly removes the second last vertex until the path is feasible. (The algorithm does not remove the last vertex because it is the exit vertex.) Once the path is feasible, the algorithm repeatedly removes the second last vertex probabilistically (lines 10-12).[2]

Phase 3: The algorithm repeatedly inserts unvisited vertices greedily similar to the construction heuristic (line 13). The difference here is that the metric used can be one of five different metrics, either the metric chosen for the construction heuristics or one of its four variants described above. The algorithm starts by choosing one of the five metrics randomly (line 4). If there are no improvements in *maxIterNoImprove* iterations, the algorithm chooses a new different metric randomly (lines 25-26). These different metrics correspond to the different "neighborhoods" in our variable neighborhood search.

Phase 4: The algorithm then updates the current path to the new neighboring path, which is a result from inserting unvisited vertices in Phase 3, if the new path is a better path or with a probability that depends on the simulated annealing temperature (lines 14-17).

Reusing Matrix Computations: We re-compute the completion probability $\hat{P}(a_n \leq H)$ whenever we make a local move during the search, that is, when (a) two vertices are swapped, (b) a vertex is removed, and/or (c) a vertex is inserted. To compute these probabilities efficiently, we store the results of the products of transition matrices for subsets of vertices. For example, in a path $\pi = \langle v_1, v_2, \ldots, v_i, \ldots, v_j, \ldots \rangle$, if we swap vertices v_i and v_j, then the product of transition matrices for the vertices before v_i, the product of matrices for the vertices between v_{i+1} and v_{j-1}, and the product of matrices for the vertices after v_{j+1} remain unchanged. By storing all of these intermediate results, it is possible to make the computation of probabilities very efficient. However, it requires a significant amount memory for larger problems. In this paper, we store only the products of matrices for the vertices between the start vertex and every subsequent vertex in the path except for the exit vertex. For example, for a path $\pi = \langle v_1, v_2, v_3, v_n \rangle$, we store the product of matrices for vertices v_1 and v_2, which is $\hat{P}(a_2 \leq H)$, and the product of matrices for vertices v_1, v_2 and v_3, which is $\hat{P}(a_3 \leq H)$. While this is not the most efficient approach, it provides a good tradeoff between memory requirement and efficiency.

[2]The rand() function returns a random number in [0,1].

7 Experimental Results

We now empirically demonstrate the scalability of our approaches on synthetic datasets employed in the literature as well as a real-world dataset for a theme park navigation problem. We run our experiments on a 3.2GHz Intel i5 dual-core CPU with 12GB memory, and we set the parameters for the local search algorithm as follows: we set *maxIterNoImprove* to 50, *maxIterations* to 1500, T to 0.1 and ΔT to 0.99. We divide each travel time distribution to 100 ranges for the matrix-based computations and use 1000 samples for the sampling-based computations.

7.1 Synthetic Dataset Results

Our synthetic dataset is based on the dataset provided in [34] with 32 vertices. We assume that the total travel time of each edge is the sum of the travel time between the two vertices connected by that edge and the queueing time at the target vertex of that edge. As in [6], we assume that the total travel time of each edge is a gamma distribution, whose mean is the Euclidean distance between the vertices connected by that edge. We vary the shape parameter $2 \leq k \leq 9$ and scale parameter $1 \leq \theta \leq 4$ such that the mean of the values $\mu \approx k\theta$ is approximately equal to the product of the shape parameter k and the scale parameter θ. A gamma distribution with $k = 1$ is an exponential distribution. As we wanted a more normal-like distribution, we did not include this value of k in our experiments. We also bound the possible values of k such that shape of the distributions across time ranges do not vary significantly, and we use the same bound on the possible values of θ as in [6]. Lastly, we set rewards for each vertex to a random number between 1 and 100.

Table 1 shows our results for the construction heuristic algorithm (labeled CH) and local search algorithm (labeled LS), where we calculate the completion probability of a path (see Equation 1) using both the matrix-based approach (see Equations 5 and 6) and the sampling-based approach (see Equation 3). We report the completion probability of the best path found by the local search algorithm using the matrix-based approach (labeled P_M) and the sampling-based approach (labeled P_S). We also report the the percentage of improvement in the reward of the path found by the local search algorithm compared to the path found by the construction heuristic algorithm (denoted

(a) Results averaged across all deadlines H and risk parameters ϵ

	Matrix-based Approach					Sampling-based Approach				
	Rewards			Runtimes (s)		Rewards			Runtimes (s)	
	CH	LS		CH	LS	CH	LS		CH	LS
$\theta = 1$	87	88	(0.50)	0.5	568	876	1033	(18.75)	5.3	2443
$\theta = 2$	129	134	(1.65)	0.8	987	695	792	(17.03)	2.7	1477
$\theta = 3$	123	133	(3.97)	0.8	904	533	569	(6.63)	1.3	716
$\theta = 4$	140	140	(0.00)	0.8	864	406	428	(6.39)	0.7	388

(b) Results averaged across all scale parameters θ and risk parameters ϵ

	Matrix-based Approach					Sampling-based Approach				
	Rewards			Runtimes (s)		Rewards			Runtimes (s)	
	CH	LS		CH	LS	CH	LS		CH	LS
$H = 20$	28	28	(0.00)	0.2	238	193	220	(12.71)	0.2	221
$H = 40$	94	94	(0.11)	0.5	520	432	498	(15.00)	0.8	690
$H = 60$	138	141	(1.43)	0.8	862	657	732	(11.10)	2.0	1303
$H = 80$	155	160	(2.00)	0.9	1050	847	952	(11.35)	3.7	1858
$H = 100$	185	196	(4.12)	1.3	1485	1008	1126	(10.84)	5.7	2208

(c) Results averaged across all deadlines H and scale parameters θ

	Matrix-based Approach							Sampling-based Approach						
	Rewards			Runtimes (s)		P_M	P_S	Rewards			Runtimes (s)		P_M	P_S
	CH	LS		CH	LS			CH	LS		CH	LS		
$\epsilon = 0.1$	1	1	(0.00)	0.1	168	1.00	1.00	507	605	(18.48)	1.7	1077	0.17	0.90
$\epsilon = 0.2$	46	46	(0.00)	0.2	332	0.90	0.99	585	669	(13.98)	2.1	1186	0.15	0.81
$\epsilon = 0.3$	113	119	(3.38)	0.6	768	0.79	0.99	643	711	(10.03)	2.6	1270	0.13	0.73
$\epsilon = 0.4$	194	197	(1.23)	1.1	1248	0.66	0.97	679	757	(10.81)	2.8	1376	0.09	0.63
$\epsilon = 0.5$	246	256	(3.05)	1.6	1640	0.54	0.95	725	785	(7.71)	3.3	1371	0.07	0.55

Table 1: Experimental Results for Simpler Synthetic Datasets

in parentheses beside the local search rewards). The branch-and-bound algorithm successfully terminated for problems with small deadlines H and risk parameters ϵ only. Therefore, we did not tabulate its results as it is unfair to only consider successful runs in computing them. (We do not know the rewards and completion probabilities for unsuccessful runs.) We make the following observations:

- Table 1(a) shows that for the matrix-based approach, the solution rewards increase between $\theta = 1$ and $\theta = 2$, and remain relatively unchanged for larger values of θ. As θ increases, the variance of the gamma distributions increases as well. When $\theta = 1$, only very few ranges have non-zero transition probabilities computed with Equation 5. As a result, adding an additional edge to a solution can result in a significant decrease in completion probability. With larger values of θ, more ranges have non-zero transition probabilities, but the number of ranges and transition probabilities do not change much with increasing values of θ. Thus, the path length and, consequently, reward and runtime, typically increases as θ increases from 1 to 2, but remains relatively unchanged for larger values of θ. The runtime depends on the path length because the number of positions to check to find the best position to insert a vertex, which is done by the construction heuristic algorithm and phase 3 in the local improvement phase, depends on the path length.

  On the other hand, for the sampling-based approach, the solution rewards decrease as θ increases. Since the sampling probabilities are relatively accurate representations of the true probabilities, as the variance increases, adding an additional edge to a solution can result in a significant decrease in completion probability. Thus, the path length and, consequently, reward and runtime, typically decreases as θ increases.

- Table 1(a) also shows that as θ increases, for the sampling-based approach, the improvement of the local search algorithm over the construction heuristic algorithm decreases. The reason is that as the variance of the gamma distributions increases, there is less distinction between the different gamma distributions. Thus, many of the neighboring solutions are very similar to the solution found by the construction heuristic algorithm. For the matrix-based approach, the improvements are all negligible. The path lengths are short (with 1-3 vertices excluding the start and end vertices), and thus there is no much room

(a) Results averaged across all risk parameters ϵ

	Matrix-based Approach					Sampling-based Approach				
	Rewards			Runtimes (s)		Rewards			Runtimes (s)	
	CH	LS		CH	LS	CH	LS		CH	LS
$H = 20$	29	29	(0.00)	0.2	262	430	537	(30.35)	0.9	1115
$H = 40$	102	103	(0.43)	0.5	571	864	1052	(22.27)	3.8	4526
$H = 60$	147	160	(6.77)	0.7	915	1156	1394	(20.92)	7.5	8255
$H = 80$	181	183	(0.44)	1.1	1326	1503	1588	(5.73)	13.6	9339
$H = 100$	247	263	(5.11)	1.8	2004	1579	1644	(4.14)	15.3	7716

(b) Results averaged across all deadlines H

	Matrix-based Approach							Sampling-based Approach						
	Rewards			Runtimes (s)		P_M	P_S	Rewards			Runtimes (s)		P_M	P_S
	CH	LS		CH	LS			CH	LS		CH	LS		
$\epsilon = 0.1$	16	16	(0.00)	0.1	215	0.98	1.00	1004	1162	(24.68)	6.9	6165	0.00	0.90
$\epsilon = 0.2$	83	83	(0.00)	0.4	500	0.87	0.97	1071	1222	(19.87)	8.2	6237	0.00	0.81
$\epsilon = 0.3$	141	144	(1.37)	0.7	975	0.78	0.97	1109	1255	(18.38)	8.1	6561	0.00	0.73
$\epsilon = 0.4$	204	224	(9.17)	1.2	1448	0.61	0.95	1144	1264	(11.25)	8.5	6054	0.00	0.64
$\epsilon = 0.5$	264	272	(2.23)	1.8	1940	0.55	0.87	1205	1311	(9.24)	9.6	5934	0.00	0.56

Table 2: Experimental Results for More Difficult Synthetic Datasets

	Rewards (Peak Days)					Rewards (Non-Peak Days)				
	$H = 2$	$H = 4$	$H = 6$	$H = 8$	$H = 10$	$H = 2$	$H = 4$	$H = 6$	$H = 8$	$H = 10$
$\epsilon = 0.1$	474	485	488	508	558	579	579	579	579	579
$\epsilon = 0.2$	474	485	488	508	558	579	579	579	579	578
$\epsilon = 0.3$	474	485	507	508	557	620	620	620	621	624
$\epsilon = 0.4$	474	485	505	549	558	620	627	627	634	634
$\epsilon = 0.5$	474	485	507	549	557	646	649	646	644	646

Table 3: Experimental Results for Real-World Theme Park Dataset

for improvement. We confirm this result with the branch-and-bound algorithm, where it found similar paths to those found by the local search algorithm for problems where it successfully terminated.

- Tables 1(b) and 1(c) show that as H or ϵ increases, the solution reward increases for both matrix- and sampling-based approaches, which is to be expected. Similarly, the runtime also increases since the number of positions to check to find the best position to insert a vertex also increases.

- Table 1(c) shows that the completion probabilities P_M and P_S are all no less than $1 - \epsilon$ for the matrix- and sampling-based approaches, respectively, which is to be expected.

We observe that the problems in this dataset are relatively easy as all gamma distributions have the same scale parameter and their means satisfy the triangle inequality. Thus, we modified the dataset to increase its difficulties in the following ways: (a) we choose the scale parameter θ of the gamma distributions for each edge randomly between 1 and 4 such that not all edges have distributions with the same scale parameter, and (b) we change the shape parameter k of the gamma distributions for some subset of edges such that their means no longer satisfy the triangle inequality. Table 2 shows our results for this more difficult synthetic dataset. We make the same observations here as in the simpler dataset with the exception that the improvements of the local search algorithm over the construction heuristic algorithm is now up to 30% as opposed to 18% earlier.

Overall, using the sampling-based approach, the local search algorithm provides reasonably better solutions compared to the construction heuristic algorithm. However, it is not guaranteed that these solutions are feasible, that is, they satisfy Equation 1. However, the feasibility likelihood increases with the number of samples. Thus, this approach is better suited for users without strict feasibility requirements. On the other hand, solution feasibility is guaranteed for algorithms using the matrix-based approach. Unfortunately, the local search algorithm fails to reasonably improve on the solutions found by the construction heuristic algorithm. Thus, the construction heuristic algorithm using the matrix-based approach is better suited for users with strict feasibility requirements.

7.2 Real-World Dataset Results

For our real-world theme park dataset, the total travel time of each edge is also a gamma distribution. We choose the scale parameter θ and shape parameter k such $\mu \approx k\theta$ and $\sigma^2 \approx k\theta^2$, where μ and σ^2 is the mean and variance, respectively, of our data points (across several months) for the sum of travel time and queueing time. Similar to the synthetic datasets, we also set rewards for each vertex to a random number between 1 and 100. However, we do not bound the possible values of the scale and shape parameters θ and k such that the gamma distributions approximate the data points as accurately as possible. Lastly, we segment our data points into two categories, peak days and non-peak days,[3] and present results for both categories.

Table 3 shows our results for the local search algorithm using the sampling-based approach to compute the completion probabilities; the deadline H is measured in hours. We only show the solution rewards as the other trends are similar to those observed for the synthetic datasets. Similar to the synthetic datasets, as H and ϵ increase, the solution reward also typically increases. Additionally, the solution rewards for non-peak days are larger than those for peak days. The reason is that the queueing time at an attraction is smaller on non-peak days than on peak days. Thus, it is possible to visit more attractions, and thus accrue more rewards, on non-peak days than on peak days.

8 Related Work

OPs have a long history and has been known by a variety of other names including selective TSPs [18], maximum collection problems [16] and bank robber problems [1]. Vansteenwegen et al. recently presented a broad overview of the problem, its variants and associated solution methods [36]. In contrast, to the best of our knowledge, there has not been much work in stochastic variants of OP thus far. Aside from the work on SOPs [6], two closely related problems with stochastic travel times are time-constrained TSPs with stochastic travel and service times [33] and stochastic selective TSPs [31]. These problems assume that the traveling time distributions are time independent, unlike our dynamic SOPs. Lastly, aside from travel time, researchers have also investigated stochasticity in the reward values of vertices [15].

SOPs also bear some similarity with Markov random fields (MRFs) [37] and Bayesian networks [28]. They are both graphical models, where nodes in a graph correspond to random variables and edges in a graph correspond to potential functions between pairs of random variables. While MRF graphs can be cyclic, Bayesian network graphs are strictly acyclic. The goal in these two models is to compute the maximum a posteriori (MAP) assignment, which is the most probable assignment to all the random variables of the underlying graph in MRFs [37, 17, 30] and Bayesian networks [24, 14, 39]. Thus, the main difference between MAP assignment problems and SOPs is that MAP assignment problems are *inference* problems while SOPs are *planning* problems. DSOPs can potentially be represented using the TiMDP model [5]. However, the objective in TiMDPs is to maximize expected reward, which is unlike in DSOPs, where we also consider the robustness criterion.

With respect to modeling and accounting for different risk preferences, there are generally the following three approaches: (1) Stochastic dominance, whose theory was developed in statistics and economics [19, 13]. Stochastic dominance defines partial orders on the space of random variables and allow for pairwise comparison of different random variables. (2) Mean-risk analysis, whose models originate from finance. They include the well known mean-variance optimization model in portfolio optimization, where the variance of the return is used as the risk functional [21]. (3) Chance constraints or percentile optimization, whose models were initiated and developed in operations research [22, 25]. Recently, researchers have provided a thorough overview of the state-of-the-art of the optimization theory with chance constraints [26]. Our approach of defining a risk-sensitive measure that allows the user to specify a level of risk (failure tolerance) is along the lines of using chance constraints to model and account for different risk preferences. While it has been applied to solve planning and scheduling problems [3, 8, 10], to the best of our knowledge, it has yet to be applied to solve OPs.

9 Conclusions

Researchers have used OPs to model vehicle routing and tourist trip design problems. However, OPs assume that the travel times are independent of the time of day and the route returned is independent of the risk preference of the user. Therefore, we make the following contributions in this paper: (1) we introduce a dynamic and stochastic OP (DSOP) model that allow for time-dependent travel times; (2) we propose a risk-sensitive criterion that allow for different risk preferences; and (3) we develop a local search algorithm to solve DSOPs with this risk-sensitive criterion. We also empirically show that this approach is applicable on a real-world theme park navigation problem. We anticipate that such user-centric route guidance will be more widely used as users carry smart devices and use mobile applications with increasing intensity.

[3] Peak days are Fridays, Sundays and Mondays according to our theme park operator.

References

[1] E. Arkin, J. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 307–316, 1998.

[2] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[3] J. C. Beck and N. Wilson. Proactive algorithms for job shop scheduling with probabilistic durations. *Journal of Artificial Intelligence Research*, 28(1):183–232, 2007.

[4] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing*, 37(2):653–670, 2007.

[5] J. Boyan and M. Littman. Exact solutions to time dependent MDPs. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1026–1032, 2001.

[6] A. Campbell, M. Gendreau, and B. Thomas. The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, 186(1):61–81, 2011.

[7] I.-M. Chao, B. Golden, and E. Wasil. Theory and methodology – the team orienteering problem. *European Journal of Operational Research*, 88:464–474, 1996.

[8] X. Chen, M. Sim, P. Sun, and J. Zhang. A linear decision-based approximation approach to stochastic programming. *Operations Research*, 56(2):344–357, 2008.

[9] M. Fischetti, J. J. S. Gonzlez, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10:133–148, 1998.

[10] N. Fu, H. C. Lau, P. Varakantham, and F. Xiao. Robust local search for solving RCPSP/max with durational uncertainty. *Journal of Artificial Intelligence Research*, 28(1):43–86, 2012.

[11] B. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307–318, 1987.

[12] A. Gupta, R. Krishnaswamy, V. Nagarajan, and R. Ravi. Approximation algorithms for stochastic orienteering. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2012.

[13] G. Hanoch and H. Levy. The efficiency analysis of choices involving risk. *Review of Economic Studies*, 36(3):335–346, 1969.

[14] J. Huang, M. Chavira, and A. Darwiche. Solving MAP exactly by searching on compiled arithmetic circuits. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 143–148, 2006.

[15] T. Ilhan, S. Iravani, and M. Daskin. The orienteering problem with stochastic profits. *IIE Transactions*, 40:406–421, 2008.

[16] S. Kataoka and S. Morito. An algorithm for the single constraint maximum collection problem. *Journal of the Operations Research Society of Japan*, 31(4):515–530, 1988.

[17] A. Kumar and S. Zilberstein. MAP estimation for graphical models by likelihood maximization. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1180–1188, 2010.

[18] G. Laporte and S. Martello. The selective traveling salesman problem. *Discrete Applied Mathematics*, 26:193–207, 1990.

[19] E. Lehmann. Ordered families of distributions. *Annals of Mathematical Statistics*, 26(3):399–419, 1955.

[20] A. Leifer and M. Rosenwein. Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research*, 73:517–523, 1994.

[21] H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[22] B. Miller and H. Wagner. Chance constrained programming with joint constraints. *Operations Research*, 13(6):930–945, 1965.

[23] R. Nauss. An efficient algorithm for the 0-1 knapsack problem. *Management Science*, 23(1):27–31, 1976.

[24] J. Park and A. Darwiche. Solving MAP exactly using systematic search. In *Proceedings of the Conference on Uncertainty in Articial Intelligence (UAI)*, pages 459–468, 2003.

[25] A. Prekopa. Contributions to the theory of stochastic programming. *Mathematical Programming*, 4(1):202–221, 1973.

[26] A. Prekopa. Probabilistic programming. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*. Elsevier, 2003.

[27] R. Ramesh, Y.-S. Yoon, and M. Karwan. An optimal algorithm for the orienteering tour problem. *INFORMS Journal on Computing*, 4(2):155–165, 1992.

[28] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[29] M. Schilde, K. Doerner, R. Hartl, and G. Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201, 2009.

[30] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2011.

[31] H. Tang and E. Miller-Hooks. Algorithms for a stochastic selective travelling salesperson problem. *Journal of the Operational Research Society*, 56:439–452, 2005.

[32] M. F. Tasgetiren. A genetic algorithm with an adaptive penalty function for the orienteering problem. *Journal of Economic and Social Research*, 4(2):1–26, 2001.

[33] S. Teng, H. L. Ong, and H. C. Huang. An integer l-shaped algorithm for the time-constrained traveling salesman problem with stochastic travel times and service times. *Asia-Pacific Journal of Operational Research*, 21:241–257, 2004.

[34] T. Tsiligrides. Heuristic methods applied to orienteering. *Journal of Operation Research Society*, 35(9):797–809, 1984.

[35] P. Vansteenwegen and D. V. Oudheusden. The mobile tourist guide: An OR opportunity. *OR Insights*, 20(3):21–27, 2007.

[36] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209:1–10, 2011.

[37] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

[38] Q. Wang, X. Sun, B. L. Golden, and J. Jia. Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research*, 61:111–120, 1995.

[39] C. Yuan and E. Hansen. Efcient computation of jointree bounds for systematic MAP search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1982–1989, 2009.

Computing Optimal Security Strategies for Interdependent Assets

Joshua Letchford
Duke University
Durham, NC
jcl@cs.duke.edu

Yevgeniy Vorobeychik
Sandia National Laboratories*
Livermore, CA
yvorobe@sandia.gov

Abstract

We introduce a novel framework for computing optimal randomized security policies in networked domains which extends previous approaches in several ways. First, we extend previous linear programming techniques for Stackelberg security games to incorporate benefits and costs of arbitrary security configurations on individual assets. Second, we offer a principled model of failure cascades that allows us to capture both the direct and indirect value of assets, and extend this model to capture uncertainty about the structure of the interdependency network. Third, we extend the linear programming formulation to account for exogenous (random) failures in addition to targeted attacks. The goal of our work is two-fold. First, we aim to develop techniques for computing optimal security strategies in realistic settings involving interdependent security. To this end, we evaluate the value of our technical contributions in comparison with previous approaches, and show that our approach yields much better defense policies and scales to realistic graphs. Second, our computational framework enables us to attain theoretical insights about security on networks. As an example, we study how allowing security to be endogenous impacts the relative resilience of different network topologies.

*Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

1 Introduction

Game theoretic approaches to security have received much attention in recent years. There have been numerous attempts to distill various aspects of the problem into a model that could be solved in closed form, particularly accounting for interdependencies of security decisions (e.g., Kunreuther and Heal [2003], Grossklags et al. [2008]). Numerous others offer techniques based on mathematical programming to solve actual instances of security problems. One important such class of problems is network interdiction [Cormican et al., 1998], which models zero-sum encounters between an interdictor, who attempts to destroy a portion of a network, and a smuggler, whose goal typically involves some variant of a network flow problem (e.g., maximizing flow or computing a shortest path).

Our point of departure is another class of optimization-based approaches in security settings: Stackelberg security games [Paruchuri et al., 2008]. These are two-player games in which a *defender* aims to protect a set of targets using a fixed set of limited defense resources, while an *attacker* aims to assail a target that maximizes his expected utility. A central assumption in the literature on Stackelberg security games is that the defender can commit to a probabilistic defense (equivalently, the attacker observes the probabilities with which each target is covered by the defender, but not the actual defense realization).

Much of the work on Stackelberg security games focuses on building fast, scalable algorithms, often in restricted settings [Kiekintveld et al., 2009, Jain et al., 2010, Shieh et al., 2012]. One important such restriction is to assume that targets exhibit *independence*: that is, the defender's utility only depends on which target is attacked and the security configuration at that target. Short of that restriction, one must, in principle, consider all possible combinations of security decisions jointly for all targets, making scalable computation elusive. Many important settings, how-

ever, exhibit interdependencies between potential targets of attack. These may be explicit, as in IT and supply chain network security, or implicit, as in defending critical infrastructure (where, for example, successful delivery of transportation services depends on a highly functional energy sector, and vice versa), or in securing complex software systems (with failures at some modules having potential to adversely affect other modules). While in such settings the assumption of independence seems superficially violated, we demonstrate below that under realistic assumptions about the nature of interdependencies, we can nevertheless leverage the highly scalable optimization techniques which assume independence.

In all, we offer the following contributions. (1) We modify and extend the previous linear programming techniques for Stackelberg security games to allow for an arbitrary set of security configurations (rather than merely to cover a target, or not), as well as to account for both random and targeted failures, and replace hard constraints on defense resources with costs associated with specific security configurations (Section 3). (2) We present and justify a crucial assumption on the nature of interdependencies that allows us to use our LP formulations which fundamentally assume independence between targets (Section 4.1). We then offer a simple model of interdependencies based on probabilistic failure cascades satisfying this assumption. Our model makes an explicit distinction between an intrinsic and indirect value of assets, the latter being due entirely to interdependencies. This allows an economically meaningful extension of a well-known independent cascade approach to modeling the spread of infectious diseases or ideas (Section 4). (3) We demonstrate that for trees we can compute expected utilities for all targets in linear time (Section 4). (4) We extend our model to capture uncertainty about network structure, and experimentally study the impact of such uncertainty (Sections 4.5 and 6.1). (5) We show that our approach is both scalable to realistic security settings and offers much better solutions than state-of-the-art alternatives (Sections 5.2 and 5.2). (6) We experimentally study the properties of optimal defense configurations in real and generated networks (Section 6).

2 Stackelberg Security Games

A Stackelberg security game consists of two players, the leader (defender) and the follower (attacker), and a set of possible targets. The leader can decide upon a randomized policy of defending the targets, possibly with limited defense resources. The follower (attacker) is assumed to observe the randomized policy of the leader, but not the realized defense actions. Upon observing the leader's strategy, the follower chooses a target so as to maximize its expected utility.

In past work, Stackelberg security game formulations focused on defense policies that were costless, but resource bounded. Specifically, it had been assumed that the defender has K fixed resources available with which to cover targets. Additionally, security decisions amounted to covering a set of targets, or not. While in numerous settings to which such work has been applied (e.g., airport security, federal air marshal scheduling) this formulation is very reasonable, in other settings one may choose among many *security configurations* for each valued asset, and, additionally, security resources are only available at some cost. For example, in cybersecurity, protecting computing nodes could involve configuring anti-virus and/or firewall settings, with stronger settings carrying a benefit of better protection, but at a cost of added inconvenience, lost productivity, as well as possible licensing costs. Indeed, costs on resources may usefully replace resource constraints, since such constraints are often not hard, but rather channel an implicit cost of adding further resources.

While security games as described above naturally entail an attacker, most systems exhibit failures that are not at all a deliberate act of sabotage, but are due entirely to inadvertent errors. Even though such failures are generally far more common than attacks, the vast majority of work in security games posits an attacker, but ignores such failures entirely; essentially the lone exception is a paper by Zhuang and Bier [2007] which offers an analytic treatment of a simple model making explicit the distinction between attacks and natural disasters. Our formulation below is, to our knowledge, the first to explicitly model both attacks and random failures in the Stackelberg security game literature.

To formalize, suppose that the defender can choose from a finite set O of security configurations for each target $t \in T$, with $|T| = n$. A configuration $o \in O$ for target $t \in T$ incurs a cost $c_{o,t}$ to the defender. If the attacker happens to attack t while configuration o is in place, the expected value to the defender is denoted by $U_{o,t}$, while the attacker's value is $V_{o,t}$. A key assumption in Stackelberg security games is that the targets are completely independent: that is, player utilities only depend on the target attacked and its security configuration [Kiekintveld et al., 2009]. We revisit this assumption below when we turn to networked (interdependent) settings. We denote by $q_{o,t}$ the probability that the defender chooses o at target t. Finally, let r be the prior probability of the defender that a failure will happen due to a deliberate attack. If no attack is involved, any target can fail; the defender's belief that target t randomly fails (conditional on the event that no attack is involved) is g_t, with $\sum_t g_t = 1$.

3 Computing Optimal Randomized Security Configurations

Previous formulations of Stackelberg security games involved a fixed collection of defender resources, and in most cases a binary decision to be made for each target: to cover it, or not. To adapt these to our domains of interest, we first modify the well-known multiple linear program (henceforth, multiple-LP) formulation that assumes target independence to incorporate an arbitrary set of security configurations, together with their corresponding costs of deployment. In the multiple-LP formulation, each linear program solves for an optimal randomized defense strategy *given that the attacker attacks a fixed target* $\hat{t}$, with the constraint that $\hat{t}$ is an optimal choice for the attacker. The defender then chooses the best solution from all feasible LPs as his optimal randomized defense configuration. The independence assumption becomes operational here because we treat defense configurations $q_{o,t}$ for each target in isolation, as this assumption obviates the need to randomize over joint defense schedules for all targets. The LP formulation for a representative target $\hat{t}$ is shown in Equations 1a-1d.

$$\max r\left(\sum_o U_{o,\hat{t}} q_{o,\hat{t}}^{\hat{t}}\right) + (1-r)\left(\sum_{t,o} g_t U_{o,t} q_{o,t}^{\hat{t}}\right) - \sum_t \sum_o c_{o,t} q_{o,t}^{\hat{t}}. \quad (1a)$$

s.t.

$$\forall_{o,t}\ q_{o,t}^{\hat{t}} \in [0,1] \quad (1b)$$

$$\forall_t \sum_o q_{o,t}^{\hat{t}} = 1 \quad (1c)$$

$$\forall_t \sum_o V_{o,t} q_{o,t}^{\hat{t}} \leq \sum_o V_{o,\hat{t}} q_{o,\hat{t}}^{\hat{t}} \quad (1d)$$

The intuition behind the multiple-LP formulation is that in an optimal defense configuration, the attacker must (weakly) prefer to attack *some* target, and, consequently, one of these LPs must correspond to an optimal defense policy.

Notice that we can easily incorporate additional linear constraints. For example, it is often useful to add a budget constraint of the form:

$$\forall_{\hat{t},t} \quad \sum_o c_{o,t} q_{o,t}^{\hat{t}} \leq B.$$

4 Incorporating Network Structure

4.1 A General Model of Interdependencies

Thus far, a key assumption has been that the utility of the defender and the attacker for each target depends only on the defense configuration for that target, as well as whether it is attacked or not. In many domains, such as cybersecurity and supply chain security, assets are fundamentally interdependent, with an attack on one target having potential consequences for others. In this section, we show how to transform certain important classes of problems with interdependent assets into a formulation in which targets become effectively independent, for the purposes of our solution techniques.

Below we focus on the defender's utilities; attacker is treated identically. Let w_t be an *intrinsic worth* of a target to the defender, that is, how much loss the defender would suffer if this target were to be compromised with no other target affected (i.e., not accounting for indirect effects). In doing so, we assume that these worths are independent for different targets. Let $s = \{o_1, \ldots, o_n\}$ be the security configuration on all nodes. The probability that a given t' is affected depends on s and the target t chosen by the attacker. Let $z_{s,t'}(t)$ be the marginal probability that target t' is affected when the attacker attacks target t. Assuming that the utility function is additive in target-specific worths and the attacker can only attack a single target, the defender's expected utility from choosing s when t is attacked is

$$U_t(s) = E\left[\sum_{t'} w_{t'} 1(\mathrm{t}' \text{ affected} \mid \mathrm{s}, \mathrm{t})\right] = \sum_{t'} w_{t'} z_{s,t'}(t),$$

where $1(\cdot)$ is an indicator function. This expression makes apparent that in general $U_t(s)$ depends on defense configurations at all targets, making the problem intractable. We now make the crucial assumption that enables fast computation of defender policies by recovering inter-target independence.

Assumption 1. *For all t and t', $z_{s,t'}(t) = z_{o_t,t'}(t)$.*

In words, the probability that a target t' is affected when t is attacked only depends on the security configuration at the attacked target t. Below, we use o instead of o_t where t is clear from context.

A way to interpret our assumption is that security against external threats is not very efficacious once an attack has found a way into the system. Alternatively, if the utility of nodes is derived from their contribution to overall connectivity (e.g., in communication networks, where removing a node can, for example, increase latency), it is quite natural to assume that removal of a node impacts global connectivity *regardless of security policies on other nodes.* Our assumption was also operational in other work on interdependent security [Kunreuther and Heal, 2003], where a justification is through a story about airline baggage screening: baggage that is transferred between

airlines is rarely thoroughly screened, perhaps due to the expense. Thus, even while an airline may have very strong screening policies, it is poorly protected from luggage entering its planes via transfers. Cybersecurity has similar shortcomings: defense is often focused on external threats, with little attention paid to threats coming from computers internal to the network. Thus, once a computer on a network is compromised, the attacker may find it much easier to compromise others on the same network.

Under the above assumption, the defender utility when t is attacked under security configuration o is:

$$U_{o,t} = z_{o,t}(t)w_t + \sum_{t' \neq t} z_{o,t'}(t)w_{t'}.$$

By a similar argument and an analogous assumption for the attacker's utility, we thereby recover target independence required by the linear programming formulations above.

4.2 Cascading Failures Model

In general, one may use an arbitrary model to compute or estimate $z_{o,t'}(t)$. Here, we offer a specific model of interdependence between targets that is simple, natural, and applies across a wide variety of settings.

Suppose that dependencies between targets are represented by a graph (T, E), with T the set of targets (nodes) as above, and E the set of edges (t, t'), where an edge from t to t' (or an undirected edge between them) means that target t' depends on target t (and, thus, a successful attack on t may have impact on t'). Each target has associated with it a worth, w_t as above, although in this context this worth is incurred only if t is affected (e.g., compromised, broken). The security configuration determines the probability $z_{o,t}(t)$ that target t is affected if the attacker attacks it *directly* and the defense configuration is o. We model the interdependencies between the nodes as independent cascade contagion, which has previously been used primarily to model diffusion of product adoption and infectious disease [Kempe et al., 2003, Dodds and Watts, 2005]; Mounzer et al. [2010] is a rare exception (a similar model was also proposed by Tsai et al. [2012], but involves both a defender and an attacker maximizing impact of information diffusion through cascades). The contagion proceeds starting at an attacked node t, affecting its network neighbors t' each with probability $p_{t,t'}$; the contagion then spreads from the newly affected nodes t' to their neighbors, and so on. The contagion can only occur once along any network edge, and once a node is affected, it stays affected through the diffusion process. An equivalent way to model this process is to start with the network (T, E) and remove each edge (t, t') with probability $(1-p_{t,t'})$. The entire connected component of an attacked node is then deemed affected.

4.3 Computing Expected Utilities

Given the independent cascade model of interdependencies between targets, we must compute expected utilities, $U_{o,t}$ and $V_{o,t}$, of the defender and the attacker respectively (note that these are expectations only over the cascades, but not the defender's mixed strategy). In general, we can do so by simulating cascades starting at every node t (using breadth-first search), with expected utility of defender/attacker estimated as a sample average over K simulated cascades (expectation in this case is with respect to random realizations of attack success for specific targets as well as edges that become a part of the failure contagion). In several special cases, however, we can either compute these exactly and efficiently, or speed up utility estimation. We now address these special cases.

4.3.1 Cascades on Trees

It is intuitive that when the dependency graph is a tree, expected utilities can be computed efficiently. A naive algorithm can do it in linear time *for each target* t, yielding quadratic time in total (since we must repeat the process for all targets). In fact, we can do it in linear time *for all targets*, as the following theorem asserts.

Theorem 1. *If (T, E) is an undirected tree we can compute expected utilities for at targets in $O(|T|)$ time.*

The proofs of this and other results can be found in the online supplement.

4.3.2 Cascades on Undirected Graphs

In general undirected graphs, we can apply a very simple optimization in the way we sample cascades to obtain substantial speedups when the graph is dense. First, observe that rather than determining live edges as the cascade unfolds, we can instead flip the biased coin for each edge to determine whether it is live or not during a particular cascade *prior* to propagating the failure. The resulting graph contains a subset of edges from the original graph. At this point, observe that each potential target in a given connected component will result in the same defender/attacker utility. We therefore only need to compute the expected loss once for each connected component. When the size of the largest connected component is $O(|T|)$, a likely scenario in dense graphs, this optimization results in an $O(|T|)$ speedup.

4.4 The Significance of Capturing Interdependence

An obvious question that may arise upon pondering the complexities of our framework is whether they are worthwhile: it may well be that previous approaches which assume target independence offer satisfactory approximation. We now show theoretically, and later experimentally, that our approach improves dramatically as compared to one which assumes independence.

Proposition 1. *There exists a family of problem instances for which the independence assumption yields a solution that is a factor of $O(n)$ worse than optimal.*

4.5 Incorporating Uncertainty about the Network

Applying our framework in real-world networked security settings requires an accurate understanding of the interdependencies. Thus far, we assumed that the actual network over which cascading failures would spread is perfectly known. A natural question is: what if our network model is inaccurate?

Formally, we model the uncertainty about the network as a parameter ϵ which represents the probability of incorrectly estimating the relationship between a pair of targets. Thus, if there is an edge between t and t', we now let this edge be present with probability $1-\epsilon$. On the other hand, if t and t' are not connected in the graph given to us, we propose that they are, in fact, connected with probability ϵ. Thus, when the graph is large, even a small amount noise will cause us to err about a substantial number of edges.[1]

Note that there is a natural way to incorporate this model of uncertainty into our framework. Let us interpret $p_{t,t'}$ as the probability of a cascade from t to t' *conditional on an edge from t to t'*. Then, if t and t' are connected, we modify cascade probabilities to be $\hat{p}_{t,t'} = p_{t,t'}(1-\epsilon)$, whereas if they are not connected, the cascade probability is $\hat{p}_{t,t'} = p_{t,t'}\epsilon$.

5 Experiments

The goal of this section is to illustrate the value of our framework as a computational tool for designing security in interdependent settings. Specifically, we aim to demonstrate that our approach clearly improves on state-of-the-art alternatives, and offers a scalable solution for realistic security problems. We pursue this aim by randomly constructing dependency graphs using Erdos-Renyi (ER) and Preferential Attachment (PA) generative models [Newman, 2010], as well as using a graph representing a snapshot of Autonomous System (AS) interconnections generated using Oregon route-views [of Oregon Route Views Project]; this graph contains 6474 targets and 13233 edges and thus offers a reasonable test of scalability. In the ER model every directed link is made with a specified and fixed probability p; we refer to it as ER(p). The PA model adds nodes in a fixed sequence, starting from an arbitrary seed graph with at least two vertices. Each node i is attached to m others stochastically (unless $i \leq m$, in which case it is connected to all preceding nodes), with probability of connecting to a node j proportional to the degree of j, d_j.

For the randomly generated networks, all data presented is averaged over 100 graph samples. Since we generate graphs that may include undirected cycles, we obtain expected utilities for all nodes on a given graph using 10,000 simulated cascades (below we show that this is more than sufficient). Intrinsic worths w_t are generated uniformly randomly on $[0, 1]$. Cascade probabilities $p_{t,t'}$ were set to 0.5 unless otherwise specified. In the sequel, we restrict the defender to two security configurations at every target, one with a cost of 0 which stops attacks with probability 0 and one with a cost of c which prevents attacks with probability 1.

5.1 Sampling Efficiency

Throughout our experiments we use 10,000 samples to evaluate the expected utilities of players. A natural question is: are we taking enough samples? To answer this, we systematically varied the number of samples between 0 (i.e., letting $U_{o,t} = -w_t$) and 100,000. Our results offer strong evidence that 10,000 samples is more than enough: the expected utility (evaluated using 100,000 samples) of the resulting defense configurations becomes flat already when the number of samples is 1000.

5.2 Scalability

An important question given the complexity of our framework is whether it can scale to realistic defense scenarios. To test this, we ran our framework on the AS graph consisting of 6474 targets and 13233 edges. Since this is a large undirected graph containing cycles, a sampling approach was required, but the total running time (including both sampling and solving linear programs) amounted to less than 1 hour. Given the importance of security, and the fact that *distributions* of security settings are computed once (or at least infrequently, as long as significant changes to the

[1] We assume here that both the defender and attacker share the same uncertainty about the network. An alternative model could consider an attacker that has more (or exact) information about the network. The resulting defender problem would become a Bayesian Stackelberg game.

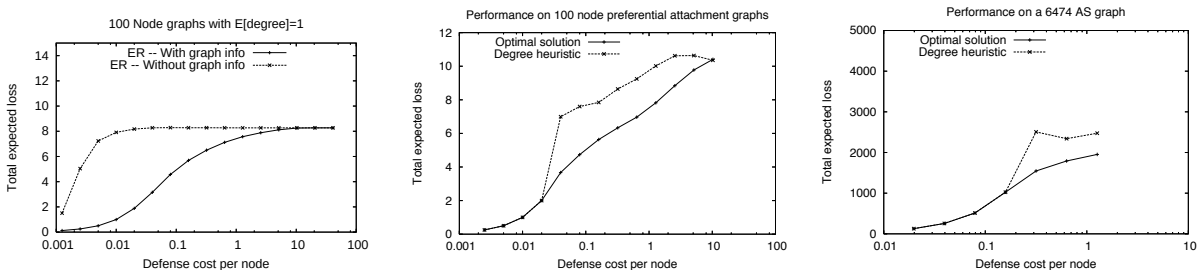


Figure 1: Left: Comparison between our approach ("with graph info") and one assuming independence ("without graph info") using the ER(0.1) generative model. Middle: Comparison to the degree-based heuristic on PA graphs. Right: Comparison to the degree-based heuristic on the AS graph.

interdependency structure are not very frequent), this seems a relatively small computational burden.

5.3 Comparison to State-of-the-Art Alternatives

There are two prime computational alternatives to our framework. The first is to assume that targets are independent. While we showed above that in the worst case this can be quite a poor approximation, we offer empirical support to the added value of our approach below. The second is to use a well-known heuristic developed in the context of vaccination strategies on networks. This latter heuristic would in our case defend nodes in order of their connectivity (degree), until the defense budget is exhausted. Figure 1 compares our approach first to the former (left) and then to the latter (right). In both cases, computing optimal defense strategies using our framework yields much higher utility to the defender than the alternatives.

Aside from interdependencies, two other important aspects of our model are the fact that it allows an arbitrary number of security configurations, instead of simply allowing the defender to defend, or not, each target, and its ability to optimize with respect to both intelligent attackers and inadvertent failures. We now show that both of these can add substantial value. Figure 2 (top) shows a comparison between a solution which only allows two configurations (defend and do not defend) and two solutions which also allow for a third configuration, which is less effective than full defense, but also less costly. We consider two potential third options, one providing 50% defense at 12.5% of the cost of full defense (1/2 – 1/8) and one providing 75% defense at 12.5% cost (3/4 – 1/8). It is clear from this graph that considering the third configuration adds considerable value. Figure 2 (bottom) assumes that all (or nearly all) failures arise randomly, and compares a solution which posits an attacker to an optimal solution. Again, the value of solving the problem optimally is clear. This plot actually shows an interesting pattern, as the expected utility of the defender is non-monotonic in cost when the solution is suboptimal. This is because the differences between the two solutions are most important when costs are intermediate; with low costs, nearly everything is fully defended, while high costs imply almost no defense.

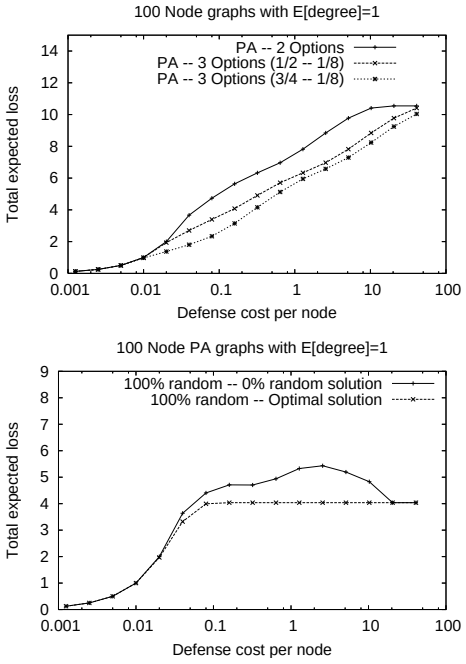


Figure 2: Top: Comparison between assuming only two configurations, and allowing the defender to consider three alternatives. Bottom: Comparison between a solution which assumes that failures are only due to attacks, and an optimal solution, when failures are actually random. Comparisons use PA graphs.

6 Applications to Interdependent Security Analysis

In this section we apply our framework to several network security domains. For simplicity, we restrict at-

tention to zero-sum security games, unless otherwise specified. As above, we consider ER and PA generative models, although we utilize a generalized version of PA. In a generalized PA model, connection probabilities are $\frac{(d_i)^\mu}{\sum_j (d_j)^\mu}$, such that when $\mu = 0$ the degree distribution is relatively homogeneous, just as in ER, $\mu = 1$ recovers the "standard" PA model, and large values of μ correspond to highly inhomogeneous degree distributions. Throughout, we use $\mu = 1$ unless otherwise specified. All parameters are set as in the experiments section, unless otherwise specified.

In addition to generative models of networks, we explore two networks derived from real security settings: one with 18 nodes that models dependencies among critical infrastructure and key resource sectors (CIKR), as inferred from the DHS and FEMA websites, and the second with 66 nodes that captures payments between banks in the core of the Fedwire network [Soramaki et al., 2007].

For the CIKR network, each node was assigned a low, medium, or high worth of 0.2, 0.5, or 1, respectively, based on perceived importance (for example, the energy sector was assigned a high worth, while the national monuments and icons sector a low worth). Each edge was categorized based on the importance of the dependency (gleaned from the DHS and FEMA websites) as "highly" or "moderately" significant, with cascade probabilities of 0.5 or 0.1 respectively. For the Fedwire network, all nodes were assigned an equal worth of 0.5, and cascade probabilities were discretely chosen between 0.05 and 0.5 in 0.05 increments depending on the weight of the corresponding edges shown in Soramaki et al. [2007].

6.1 The Impact of Uncertainty

Our framework offers a natural way to incorporate uncertainty about the network into the analysis. An important question is: how much impact on defender decision does uncertainty about the network have? Figure 3 quantifies the impact of uncertainty on the quality of defense if the observed graph is the PA network with average degree of 2. When cascade probabilities are relatively high ($p_{t,t'} = 0.5$ for all edges, top plot), even if the amount of noise is relatively small ($\epsilon = 0.01$), the resulting increase in the number of possible cascade paths in the network makes the defender much more vulnerable. With smaller cascade probabilities ($p_{t,t'} = 0.1$, bottom plot), however, noise has relatively little impact. It can thus be vital for the defender to obtain an accurate portrait of the true network over which failures may cascade when the interdependencies among the components are strong.

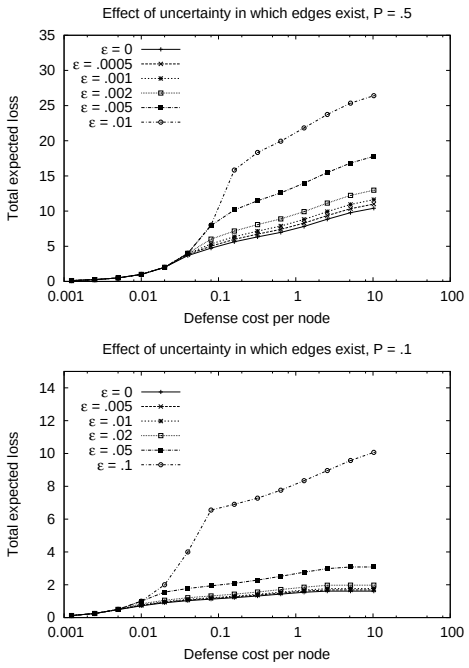


Figure 3: The impact of noise on PA networks. Top: when $p_{t,t'} = 0.5$; Bottom: when $p_{t,t'} = 0.1$.

6.2 The Impact of Marginal Defense Cost

Our next analysis deals with the impact of marginal defense cost c on defender expected losses, his total costs, and the sum of these (i.e., negative expected utility). The results for ER and BA (both with 100 nodes and average degree of 2), as well as CIKR and Fedwire networks are shown in Figure 4. All the plots feature a clear pattern: expected loss and (negative) utility are monotonically increasing, as expected, while total costs start at zero, initially rise, and ultimately fall (back to zero in 3 of the 4 cases). It may at first be surprising that total costs eventually fall even as marginal costs continue to increase, but this clearly must be the case: when c is high enough, the defender will not wish to invest in security at all, and total costs will be zero. What is much more surprising is the presence of two peaks in PA and Fedwire networks. Both of these networks share the property that there is a non-negligible fraction of nodes with very high connectivity [Newman, 2010, Soramaki et al., 2007]. When the initial peak is reached, the network is fully defended, and as marginal costs rise further, the defender begins to reduce the defense resources expended on the less important targets. At a certain point, only the most connected targets are protected, and since these are so vital to protect, total costs begin increasing again. After the second peak is reached, c is finally large enough to discourage the defender from fully protecting even the most important targets, and the subsequent fall of

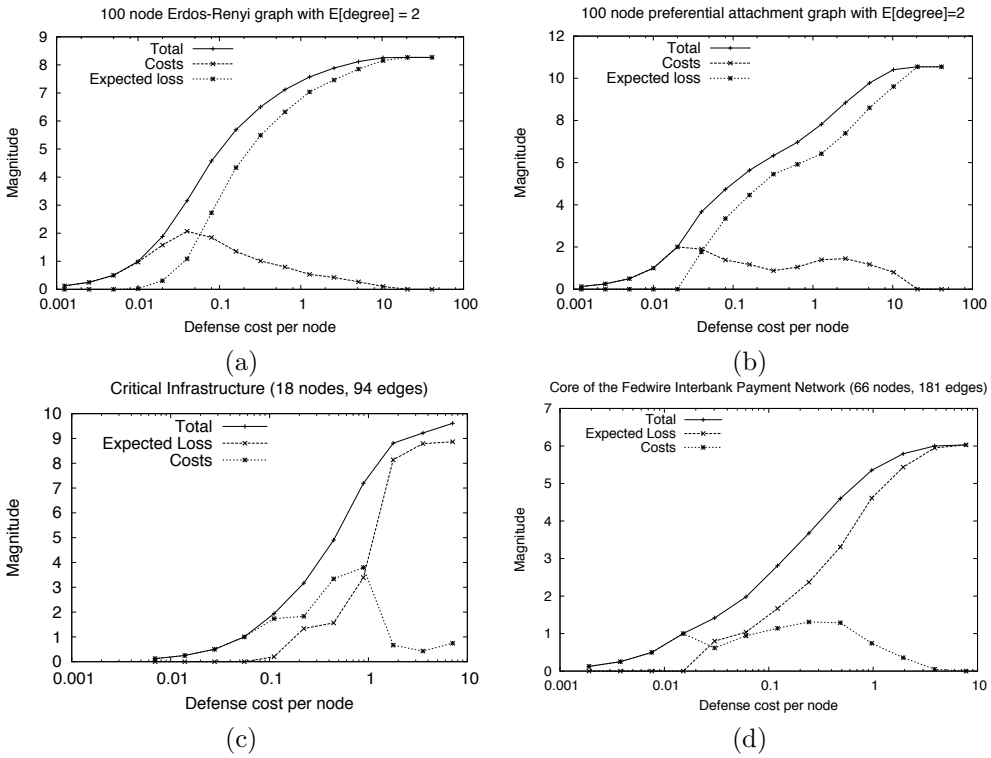


Figure 4: Expected loss, cost, and their sum in (a) 100-node ER(0.2), (b) 100-node PA, (c) 18-node critical infrastructure, and (d) 66-node core of the Fedwire networks as defense cost increases. The results for ER and PA are averages over 100 stochastic realizations of these networks.

total costs is no longer reversed.

6.3 Resilience to Targeted Attacks: Impact of Network Structure

One of the important streams in the network science literature is the question of relative resilience of different network topologies to failures, random or targeted. A central result, replicated in a number of contexts, is that network topology is a vital factor in determining resilience [Albert et al., 2000, Newman, 2010]. Of particular interest to us is the observation that scale-free networks such as PA exhibit poor tolerance to targeted attacks as compared to ER [Albert et al., 2000], which is precisely the context that we consider.

In Figure 5 (top) we show the defender's utility for three different network topologies, PA, ER, and Fedwire as a function of cost c. Remarkably, there is essentially no difference between PA and ER (and not much between these and Fedwire) until c is quite high, at which point they begin to diverge. This seems to contradict essentially all the previous findings in that network topology seems to play little role in resilience in our case! A superficial difference here is that we consider a cascading failure model, while most of the previous work on the subject focused on diminished connectivity due to attacks. We contend that the most important distinction, however, is that previous work studying resilience did not account for a simple observation that most important targets are also most heavily defended; indeed, there was no notion of endogenous defense at all. In scale-free graphs, there are well connected nodes whose failure has global consequences. These are the nodes which are most important, and are heavily defended in optimal decisions prescribed by our framework. Once the defense decision becomes endogenous, differences in network topology disappear. Naturally, once c is high enough, defense of important targets weakens, and eventually we recover the standard result: for high c, PA is considerably more vulnerable than ER.

To investigate the impact of network topology on resilience further, we consider the generalized PA model in which we systematically vary the homogeneity of the degree distribution by way of the parameter μ. The results are shown in Figure 5 (bottom). In this graph, we do observe clear variation in resilience as a function of network topology, but the operational factor in this variation is homogeneity in the distribution of expected utilities, rather than degrees: increasing *homogeneity* of the utility distribution *lowers network resilience*. This seems precisely the opposite

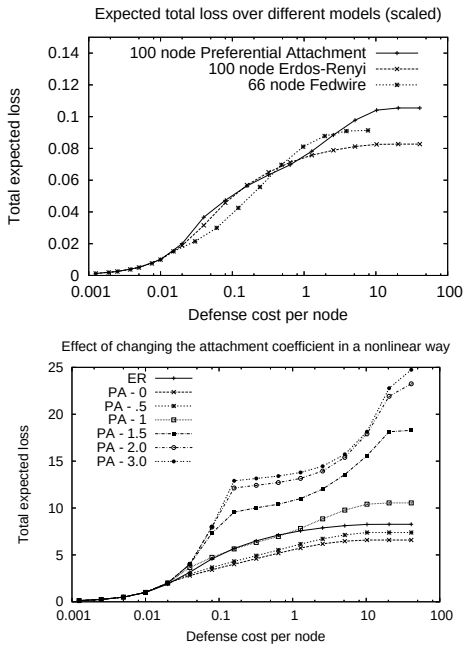


Figure 5: Top: Expected total loss: comparison across different network structures. Bottom: Expected defender disutility in the generalized PA model as we vary μ (keeping average degree fixed at 2). ER is also shown for comparison.

of the standard results in network resilience, but the two are in fact closely related, as we now demonstrate. Superficially, the trend in the figure seems to follow the common intuition in the resilience literature: as the degree distribution becomes more inhomogeneous (more star-like), it becomes more difficult to defend. Observe, however, that ER is actually more difficult to defend than PA with $\mu = 0$. The lone difference of the latter from ER is the fact that nodes that enter earlier are more connected and, therefore, the degree distribution in the PA variant should actually be more *inhomogeneous* than ER! The answer is that random connectivity combined with inhomogeneity of degrees actually makes the distribution of *utilities* less homogeneous in PA with $\mu = 0$, and, as a result, fewer nodes on which defense can focus as compared to ER. On the other hand, as the graph becomes more star-like, the utilities of all nodes become quite similar; in the limiting case, all nodes are only two hops apart, and attacking any one of them yields a loss of many as a result of cascades.

There is another aspect of network topology that has an important impact on resilience: network density. Figure 6 shows a plot of an Erdos-Renyi network with the probability of an edge varying between 0.0025 to 0.08 (average degree between .25 and 8) and cost c fixed at 0.04. Clearly, expected utility and loss of the defender are increasing in density, but it is rather sur-

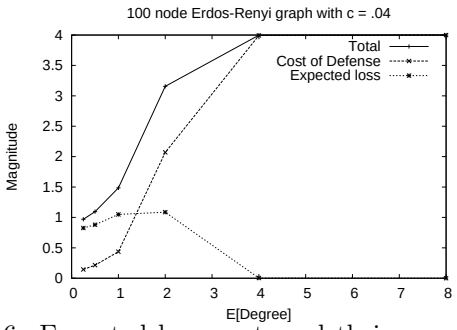


Figure 6: Expected loss, cost, and their sum in 100-node Erdos-Renyi networks as a function of network density (equivalently, expected degree).

prising to observe how sharply they jump once average degree exceeds 1 (the ER network threshold for a large connected component); in any case, network density has an unmistakable impact. The reason is intuitive: increased density means more paths between targets, and, consequently, greater likelihood of large cascades in the event that a target is compromised. Total cost initially increases in response to increased density, in part to compensate for the increased vulnerability to attacks, but eventually falls, since it is too expensive to protect everything, and anything short of that is largely ineffective.

7 Conclusion

We presented a framework for computing optimal randomized security policies in network domains, extending previous linear programming approaches to Stackelberg security games in several ways. First, we extended previous linear programming techniques to incorporate benefits and costs of arbitrary security configurations on individual assets. Second, we offered a principled model of failure cascades that allows us to capture both the direct and indirect value of assets, and showed how to extend this model to capture uncertainty about the structure of the interdependency network. Third, we allowed the defender to account for failures due to actual attacks, as well as those that are a result of exogenous failures. Our results demonstrate the value of our approach as compared to alternatives, and show that it is scalable to realistic security settings. Furthermore, we used our framework to analyze four models of interdependencies: two based on random graph generation models, a simple model of interdependence between critical infrastructure and key resource sectors, and a model of the Fedwire interbank payment network.

References

Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.

Kelly J. Cormican, David P. Morton, and R. Kevin Wood. Stochastic network interdiction. *Operations Research*, 46(2):184–197, 1998.

Peter S. Dodds and Duncan J. Watts. A generalized model of social and biological contagion. *Journal of Theoretical Biology*, 232:587–604, 2005.

Jens Grossklags, Nicolas Christin, and John Chuang. Secure or insure? A game-theoretic analysis of information security games. In *Seventeenth International World Wide Web Conference*, pages 209–218, 2008.

Manish Jain, Erim Kardes, Christopher Kiekintveld, Milind Tambe, and Fernando Ordonez. Security games with arbitrary schedules: A branch and price approach. In *Twenty-Fourth National Conference on Artificial Intelligence*, 2010.

David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence in a social network. In *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, 2009.

Howard Kunreuther and Geoffrey Heal. Interdependent security. *Journal of Risk and Uncertainty*, 26 (2-3):231–249, 2003.

Jeffrey Mounzer, Tansu Alpcan, and Nicholas Bambos. Integrated security risk management for IT-intensive organizations. In *Sixth International Conference on Information Assurance and Security*, pages 329–334, 2010.

Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010.

U. of Oregon Route Views Project. Online data and reports. http://www.routeviews.org.

Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems*, pages 895–902, 2008.

Eric Shieh, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrrett Meyer. PROTECT: A deployed game theoretic system to protect the ports of the United States. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems*, pages 13–20, 2012.

Kimmo Soramaki, Morten L. Bech, Jeffrey Arnold, Robert J. Glass, and Walter Beyeler. The topology of interbank payment flows. *Physica A*, 379: 317–333, 2007.

Jason Tsai, Thanh H. Nguyen, and Milind Tambe. Security games for controlling contagion. In *Twenty-Sixth National Conference in Artificial Intelligence*, 2012. to appear.

J. Zhuang and V.M. Bier. Balancing terrorism and natural disasters—Defensive strategy with endogenous attacker effort. *Operations Research*, 55(5): 976–991, 2007.

Nested Dictionary Learning for Hierarchical Organization of Imagery and Text

Lingbo Li
Duke University
Durham, NC 27708
ll83@duke.edu

XianXing Zhang
Duke University
Durham, NC 27708
xz60@duke.edu

Mingyuan Zhou
Duke University
Durham, NC 27708
mz31@duke.edu

Lawrence Carin
Duke University
Durham, NC 27708
lcarin@duke.edu

Abstract

A tree-based dictionary learning model is developed for joint analysis of imagery and associated text. The dictionary learning may be applied directly to the imagery from patches, or to general feature vectors extracted from patches or superpixels (using any existing method for image feature extraction). Each image is associated with a path through the tree (from root to a leaf), and each of the multiple patches in a given image is associated with one node in that path. Nodes near the tree root are shared between multiple paths, representing image characteristics that are common among different types of images. Moving toward the leaves, nodes become specialized, representing details in image classes. If available, words (text) are also jointly modeled, with a path-dependent probability over words. The tree structure is inferred via a nested Dirichlet process, and a retrospective stick-breaking sampler is used to infer the tree depth and width.

1 Introduction

Statistical topic models, such as latent Dirichlet allocation (LDA) (Blei et al., 2003b), were originally developed for text analysis, but they have been recently transitioned successfully for the analysis of imagery. In most such topic models, each image is associated with a distribution over topics, and each topic is characterized by a distribution over observed features in the image. In this setting researchers typically represent an image as a bag of visual words (Fei-Fei & Perona, 2005; Li & Fei-Fei, 2007). These methods have been applied to perform unsupervised clustering, classification and annotation of images, using image features as well as auxiliary data such as image annotations (Barnard et al., 2003; Blei & Jordan, 2003; Blei & MaAuliffe, 2007; Wang et al., 2009; Li et al., 2009).

In such work feature extraction is performed as a pre-processing step, and local image descriptors, *e.g.*, scale-invariant feature transform (SIFT) (Lowe, 1999), and other types of features (Arbelaez & Cohen, 2008), are commonly used to extract features from local patches (Fei-Fei & Perona, 2005; Li & Fei-Fei, 2007; Bart et al., 2008; Sivic et al., 2008; Wang et al., 2009), segments (Li et al., 2009; Yang et al., 2010), or superpixels (Du et al., 2009). Different images are related to one another by their corresponding distributions over topics.

There are several limitations with most of this previous work. First, vector quantization (VQ) is typically applied to the image features (Fei-Fei & Perona, 2005), and the codes play the role of "words" in traditional topic modeling. There is a loss of information in this quantization step, and one must tune the number of codes (the proper codebook may change with different types of images, and as new imagery are observed). Secondly, feature design is typically performed separately from the subsequent image topic modeling. Finally, most of the image-based topic modeling is performed at a single scale or level (Wang et al., 2009; Li et al., 2009; Du et al., 2009), thereby not accounting for the hierarchical characteristics of most natural imagery.

In recent work, there have been papers that have addressed particular aspects of the above limitations, but none that has addressed all. For example, in Li et al. (2011) the authors employed a dictionary-learning framework, eliminating the need to perform VQ. This dictionary learning could be applied to traditional features pre-computed from the image, or it could be applied directly to patches of raw imagery, thereby ameliorating the requirement of separating the feature-design and topic-modeling steps. However, Li et al. (2011) did not consider the hierarchical charac-

ter of imagery. Recently Li et al. (2010) employed a nested Chinese restaurant process (nCRP) to infer a hierarchical tree representation for a corpus of images and (if available) accompanying text; however, in that work the VQ step was still employed, and therefore a precomputation of features was as well. Further, in Li et al. (2010), while the tree width was inferred, the depth was set. Finally, the nCRP construction in Li et al. (2010) has the disadvantage of only updating parent-child-transition parameters from one node of the tree at a time, in a sampler, yielding poor mixing relative to a stick-breaking Dirichlet process (DP) implementation (Ishwaran & James, 2001). Related but distinct dictionary learning with the nCRP was considered in Zhang et al. (2011).

Motivated by these recent contributions, and the limitations of most existing topic models of imagery and text, this paper makes the following contributions:

- A nested DP (nDP) model is developed to learn a hierarchical tree structure for a corpus of imagery and text, with a stick-breaking construction employed; we infer both the tree depth and width, using a retrospective stick-breaking construction (Papaspiliopoulos & Roberts, 2008).
- A beta-Bernoulli dictionary learning framework (Zhou et al., 2011b) is adapted to such a hierarchical model, removing the VQ step, and allowing one to perform topic modeling directly on image patches, thereby integrating feature design and topic modeling. However, if desired, the dictionary learning may also be applied to features pre-computed from the image, using *any* existing method for feature design, and again removing the limitations of VQ.

2 Modeling Image Patches

We wish to build a hierarchical model to arrange M images and their associated annotations (when available); the vocabulary of such annotations is assumed to be of dimension N_v. The vector $\boldsymbol{x}_{mi}$ represents the pixels or features associated with the ith patch in image m, and $\boldsymbol{y}_m = (y_{m1}, \ldots, y_{mN_v})^T$ represents a vector of word counts associated with that image, when available (y_{mn} represents the number of times word $n \in \{1, \ldots, N_v\}$ is present in the annotation).

The mth image is divided into N_m patches (or superpixels (Li et al., 2010)), and the data for the ith patch is denoted $\boldsymbol{x}_{mi} \in \mathbb{R}^P$ with $i = 1, \ldots, N_m$. The vector $\boldsymbol{x}_{mi}$ may represent raw pixel values, or a feature vector extracted from the pixels (using any available method of image feature extraction, *e.g.*, SIFT (Lowe, 1999)). Each $\boldsymbol{x}_{mi}$ is represented as a sparse linear combination of learned dictionary atoms. Further, each patch is assumed associated with a "topic"; the probability of which dictionary atoms are employed for a given patch is dictated by the topic it is associated with.

Specifically, each patch is represented as $\boldsymbol{x}_{mi} = \mathbf{D}(\boldsymbol{z}_{mi} \odot \boldsymbol{s}_{mi}) + \boldsymbol{e}_{mi}$, where $\odot$ represents the element-wise/Hadamard product, $\mathbf{D} = [\boldsymbol{d}_1, \cdots, \boldsymbol{d}_K] \in \mathbb{R}^{P \times K}$, K is the truncation level on the possible number of dictionary atoms, $\boldsymbol{z}_{mi} = [z_{mi1}, \cdots, z_{miK}]^T$, $\boldsymbol{s}_{mi} = [s_{mi1}, \cdots, s_{miK}]^T$, $z_{mik} \in \{0, 1\}$ indicates whether the kth atom is *active* within patch i in image m, $s_{mik} \in \mathbb{R}^+$, and $\boldsymbol{e}_{mi}$ is the residual. Note that $\boldsymbol{z}_{mi}$ represents the specific sparseness pattern of dictionary usage for $\boldsymbol{x}_{mi}$. The hierarchical form of the model is

$$\begin{aligned} \boldsymbol{x}_{mi} &\sim \mathcal{N}(\mathbf{D}(\boldsymbol{z}_{mi} \odot \boldsymbol{s}_{mi}), \gamma_e^{-1}\mathbf{I}_P) \\ \boldsymbol{d}_k &\sim \mathcal{N}(0, \frac{1}{P}\mathbf{I}_P) \\ \boldsymbol{s}_{mi} &\sim \mathcal{N}_+(0, \gamma_s^{-1}\mathbf{I}_K) \\ \boldsymbol{z}_{mi} &\sim \prod_{k=1}^{K} \text{Bernoulli}(\pi_{h_{mi}k}) \end{aligned} \quad (1)$$

where gamma priors are placed on both γ_e and γ_s. Positive weights $\boldsymbol{s}_{mi}$ (truncated normal, $\mathcal{N}_+(\cdot)$) are imposed, which we have found to yield improved results.

The indicator variable h_{mi} defines the topic associated with $\boldsymbol{x}_{mi}$. The K-dimensional vector $\boldsymbol{\pi}_h$ defines the probability that each of the K columns of $\mathbf{D}$ is employed to represent topic h, where the kth component of $\boldsymbol{\pi}_h$ is π_{hk}. These probability vectors are drawn

$$\boldsymbol{\pi}_h \sim G_0, \quad G_0 = \prod_{k=1}^{K} \text{Beta}(a_0/K, b_0(K-1)/K) \quad (2)$$

where π_{hk} represents the probability of using $\boldsymbol{d}_k$ for object type h, and the introduction of G_0 is for discussions below. This representation for $\boldsymbol{\pi}_h$ corresponds to an approximation to the beta-Bernoulli process (Thibaux & Jordan, 2007; Paisley & Carin, 2009; Zhou et al., 2011a,b), which also yields an approximation to the Indian buffet process (IBP) (Griffiths & Ghahramani, 2005; Teh et al., 2007).

3 Tree Structure via nDP

The nested Dirichlet process (nDP) tree construction developed below is an alternative means of constituting the same type of tree manifested by the nested Chinese restaurant process (Blei et al., 2003a; Li et al., 2010). We emphasize the nDP construction because of the stick-breaking implementation we employ, which

allows block updates, and therefore often manifests better mixing than the nCRP-type implementation (Ishwaran & James, 2001). Related work was considered in Wang & Blei (2009), but VB inference was employed and the tree size was therefore not inferred (a fixed truncation was imposed). The retrospective sampler developed below allows inference of both the tree depth and width (Papaspiliopoulos & Roberts, 2008).

Consider a draw from a DP, $G \sim \mathrm{DP}(\gamma, G_0)$, where $\gamma > 0$ is an "innovation" parameter, with G_0 defined in (2). Then the DP draw (Ishwaran & James, 2001) may be expressed as $G = \sum_{n=1}^{\infty} \lambda_n \delta_{\phi_n}$, where $\lambda_n = \nu_n \prod_{l<n}(1-\nu_l)$, $\nu_l \sim \mathrm{Beta}(1,\gamma)$, and $\phi_n \sim G_0$; each ϕ_n corresponds to a topic, as in (2). Letting $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots)^T$, we denote the draw of $\boldsymbol{\lambda}$ as $\boldsymbol{\lambda} \sim \mathrm{Stick}(\gamma)$.

3.1 Tree width

Using notation from Adams et al. (2010), let $\boldsymbol{\epsilon}$ represent a path through the tree, characterized by a sequence of parent-child nodes, and let $|\boldsymbol{\epsilon}|$ be the length of this path (total number of layers traversed). In addition to representing a path through the tree, $\boldsymbol{\epsilon}$ identifies a node at layer $|\boldsymbol{\epsilon}|$, *i.e.*, the node at the end of path $\boldsymbol{\epsilon}$. For node $\boldsymbol{\epsilon}$, let $\boldsymbol{\epsilon}\epsilon_i$, $i = 1, 2, \dots$, denote the *children* of $\boldsymbol{\epsilon}$, at level $|\boldsymbol{\epsilon}| + 1$. To constitute a distribution over the children nodes, we draw $G_{\boldsymbol{\epsilon}} \sim \mathrm{DP}(\gamma, G_0)$, yielding $G_{\boldsymbol{\epsilon}} = \sum_{i=1}^{\infty} \lambda_{\boldsymbol{\epsilon}\epsilon_i} \delta_{\phi_{\boldsymbol{\epsilon}\epsilon_i}}$, where $\lambda_{\boldsymbol{\epsilon}\epsilon_i} = \nu_{\boldsymbol{\epsilon}\epsilon_i} \prod_{j=1}^{i-1}(1-\nu_{\boldsymbol{\epsilon}\epsilon_j})$, $\nu_{\boldsymbol{\epsilon}\epsilon_j} \sim \mathrm{Beta}(1,\gamma)$, and $\phi_{\boldsymbol{\epsilon}\epsilon_i} \sim G_0$, with G_0 defined in (2); $\boldsymbol{\lambda}_{\boldsymbol{\epsilon}} = (\lambda_{\boldsymbol{\epsilon}\epsilon_1}, \lambda_{\boldsymbol{\epsilon}\epsilon_2}, \dots)^T$ is denoted as drawn $\boldsymbol{\lambda}_{\boldsymbol{\epsilon}} \sim \mathrm{Stick}(\gamma)$. The probability measure $G_{\boldsymbol{\epsilon}}$ constitutes in principle an infinite set of children nodes, with $\lambda_{\boldsymbol{\epsilon}\epsilon_i}$ defining the probability of transiting from node $\boldsymbol{\epsilon}$ to child ϵ_i; $\phi_{\boldsymbol{\epsilon}\epsilon_i}$ constitutes the topic-dependent probability of dictionary usage at that child node.

The process continues *in principle* to an infinite number of levels, with each child node spawning an infinite set of subsequent children nodes, manifesting a tree of infinite depth and width. However, note that a draw $\boldsymbol{\lambda}_{\boldsymbol{\epsilon}}$ will typically only have a relatively small number of components with appreciable amplitude. This means that while $G_{\boldsymbol{\epsilon}}$ constitutes in principle an infinite number of children nodes, only a small fraction will be visited with appreciable probability.

Let $\boldsymbol{c}_m = (c_m^1, c_m^2, \dots)^T$ represent the path associated with image m, where c_m^l corresponds to the node selected at level l. For conciseness we write $\boldsymbol{c}_m \sim \mathrm{nCRP}(\gamma)$ (Blei et al., 2003a), emphasizing that the underlying transition probabilities $\boldsymbol{\lambda}_{\boldsymbol{\epsilon}}$, controlling the probabilistic path through the tree, are a function of parameter γ.

3.2 Tree depth

We also draw an associated probability vector $\boldsymbol{\theta}_m \sim \mathrm{Stick}(\alpha)$. Patch $\boldsymbol{x}_{mi}$ is associated with level l_{mi} in path $\boldsymbol{c}_m$, where $l_{mi} \sim \sum_{l=1}^{\infty} \theta_{ml} \delta_l$. Since $\boldsymbol{\theta}_m$ typically only has a small number of components with appreciable amplitude, the tree depth is also constrained.

3.3 Modeling words

In Section 2 we developed topic (node) dependent probabilities of atom usage; we now extend this to words (annotations), when available. A distribution over words may be associated with each topic (tree node) h. For topic h we may draw (Blei et al., 2003b)

$$\boldsymbol{\psi}_h \sim \mathrm{Dir}(\frac{\eta}{N_v}, \dots, \frac{\eta}{N_v}) \tag{3}$$

where $\boldsymbol{\psi}_h$ is the distribution over words for topic h.

Recall that each image/annotation is associated with a path $\boldsymbol{c}_m$ through a tree, and $\boldsymbol{\theta}_m$ controls the probability of employing each node (topic) on that path. Let θ_{mh} represent the probability that node h is utilized, with $h \in \boldsymbol{c}_m$. Then the "collapsed" probability of word usage on this path, marginalizing out the probability of node selection, may be expressed as

$$\boldsymbol{\psi}_{\boldsymbol{c}_m} = \sum_{h \in \boldsymbol{c}_m} \theta_{mh} \boldsymbol{\psi}_h \tag{4}$$

A probability over words $\boldsymbol{\psi}_{\boldsymbol{c}_m}$ is therefore associated with each path $\boldsymbol{c}_m$. One may argue that the $\boldsymbol{\theta}_m$ used to control node usage for image patches should be different from that used to represent words; this is irrelevant in the final model, as a path-dependent $\boldsymbol{\psi}_{\boldsymbol{c}_m}$ is drawn directly from a Dirichlet distribution (discussed below), and therefore (4) is only illustrative/motivating.

3.4 Retrospective sampling

The above discussion indicated that while the width and depth of the tree is infinite in principle, a finite tree is manifested given finite data, which motivates adaptive inference of the tree size. In a retrospective implementation of a stick-breaking process, we constitute a *truncated* stick-breaking process, denoted $\boldsymbol{w} \sim \mathrm{Stick}_L(\gamma)$, with $w_n = V_n \prod_{l<n}(1-V_l)$, $V_n \sim \mathrm{Beta}(1,\gamma)$ for $n < L$, and $V_L = 1$; here there is an L-stick truncation, yielding $\boldsymbol{w} = (w_1, \dots, w_L)^T$, with w_L representing the probability of selecting a stick *other* than sticks 1 through $L-1$.

In a retrospective sampler (Papaspiliopoulos & Roberts, 2008), each of the aforementioned sticks is truncated as above. When drawing children nodes and levels, if the last stick (the Lth above) is selected, this

implies that a new child/level must be added, since the first $L-1$ sticks are not enough to capture how the data are clustered. If stick L is selected, then a new node/level is constituted (a new child is added), by drawing a new $V_L \sim \text{Beta}(1,\gamma)$, and then $V_{L+1}=1$, thereby now constituting an $(L+1)$-dimensional stick representation; the associated node-dependent statistics are constituted as discussed in Section 2 (drawing new probabilities over dictionary elements). The model therefore infers "in retrospect" that the L-level truncation was too small, and expands adaptively. The model also has the ability to shrink the number of sticks used at any component of the model, if less than the associated truncated level is needed to define the number of children/levels are actually utilized.

3.5 Generative Process

The generative process for the model is summarized as follows:

1. Draw dictionary $\mathbf{D} \sim \prod_{k=1}^{K} \mathcal{N}(0, \frac{1}{P}\mathbf{I}_P)$
2. Draw γ, α, γ_e and γ_s from respective gamma distributions
3. For each image $m \in \{1,2,...,M\}$
   (a) Draw $\boldsymbol{c}_m \sim \text{nCRP}(\gamma)$
   (b) For each *newly utilized* node $\boldsymbol{\epsilon}$ in the tree, draw dictionary usage probabilities $\boldsymbol{\pi}_{\boldsymbol{\epsilon}} \sim \prod_{k=1}^{K} \text{Beta}(a_0/K, b_0(K-1)/K)$
   (c) Draw $\boldsymbol{\theta}_m \sim \text{Stick}(\alpha)$
   (d) For the ith patch or feature vector
      i. Draw level index $l_{mi} \sim \sum_{l=1}^{\infty} \theta_{ml}\delta_l$, which along with $\boldsymbol{c}_{mi}$ defines node h_{mi}
      ii. Draw $\boldsymbol{z}_{mi} \sim \prod_{k=1}^{K} \text{Bernoulli}(\pi_{h_{mi}k})$, and $\boldsymbol{s}_{mi} \sim \mathcal{N}_+(0, \gamma_s^{-1}\mathbf{I}_K)$
      iii. Draw $\boldsymbol{x}_{mi} \sim \mathcal{N}(\mathbf{D}(\boldsymbol{z}_{mi} \odot \boldsymbol{s}_{mi}), \gamma_e^{-1}\mathbf{I}_P)$
4. For each unique tree path p, draw $\boldsymbol{\psi}_p \sim \text{Dir}(\frac{\beta}{N_V}, \ldots, \frac{\beta}{N_V})$
5. If annotations are available for image m, $\boldsymbol{y}_m \sim \text{Mult}(|\boldsymbol{y}_m|, \boldsymbol{\psi}_{\boldsymbol{c}_m})$, where $|\boldsymbol{y}_m|$ is the total number of words in $\boldsymbol{y}_m$

In Step 3(b), new nodes (topics) are added "in retrospect", as discussed in the previous subsection (nodes may also be pruned with this sampler). After completing Step 3, the tree size is constituted, which allows Step 4, imposition of a distribution over words for each path.

Algorithm 1 Retrospective Sampling for l_{mi}

Input: L_m, $\boldsymbol{z}$, $\boldsymbol{l}$, a_0, b_0
Output: l_{mi}, L_m
for $m=1$ **to** M and $i=1$ **to** N_m **do**
  Sample $\mu_{m|\boldsymbol{\epsilon}|}$, $\boldsymbol{\pi}_{\boldsymbol{\epsilon}}$ from the conditional posterior for $|\boldsymbol{\epsilon}| \leq L_m$, and from the prior for $|\boldsymbol{\epsilon}| > L_m$; $\theta_{m|\boldsymbol{\epsilon}|} = \mu_{m|\boldsymbol{\epsilon}|}\prod_{s=1}^{|\boldsymbol{\epsilon}|-1}(1-\mu_{ms})$
  Sample $U_{mi} \sim \text{Uniform}[0,1]$
  if $\sum_{s=1}^{j-1} q(l_{mi}=s) < U_{mi} \leq \sum_{s=1}^{j} q(l_{mi}=s)$ **then**
    Set $l_{mi}=j$ with probability $\kappa_{mi}(j)$, otherwise, leave l_{mi} unchanged
  else
    $L_m = L_m+1$, set $l_{mi}=L_m$ with probability $\kappa_{mi}(L_m)$, otherwise, leave l_{mi} unchanged
  end if
end for

4 Model Inference

A contribution of this paper concerns use of retrospective sampling to infer the tree width and depth. To save space for an extensive set of experimental results, we here only discuss updates associated with inferring the tree depth. A complete set of update equations are provided in Supplementary Material, where one may also find a summary of all notation.

To sample l_{mi} from the conditional posterior, we first need to specify the likelihood that $\{\boldsymbol{\epsilon} \in \boldsymbol{c}_m\}$:

$$p(\boldsymbol{z}_{mi}|\boldsymbol{\pi}_{\boldsymbol{\epsilon}}, \boldsymbol{c}_m) = \prod_{k=1}^{K} \pi_{\boldsymbol{\epsilon}k}^{z_{mik}}(1-\pi_{\boldsymbol{\epsilon}k})^{1-z_{mik}}$$

and the prior distribution, which is specified by a stick-breaking draw $\boldsymbol{\theta}_m$ for each image m. Although l_{mi} can be sampled from a closed form posterior for a fixed L_m, here to learn L_m adaptively we instead use an Metropolis-Hastings step, where the proposal distribution is defined as

$$q(l_{mi}=j) \propto \begin{cases} \theta_{mj}p(\boldsymbol{z}_{mi}|\boldsymbol{\pi}_j, \boldsymbol{c}_m), & j \leq L_m \\ \theta_{mj}\mathcal{M}_{mi}(L_m), & j > L_m \end{cases}$$

where $\mathcal{M}_{mi}(L_m) = \max_{1\leq|\boldsymbol{\epsilon}|\leq L_m}\{p(\boldsymbol{z}_{mi}|\boldsymbol{\pi}_{\boldsymbol{\epsilon}}, \boldsymbol{c}_m)\}$. Note that the sampled value of l_{mi} is allowed to be larger than the truncation level L_m, consequently L_m and the depth of the tree is learned adaptively. The acceptance probability $\kappa_{mi}(j)$ for $l_{mi}=j$ is

$$\begin{cases} 1, & j \leq L_m \;\&\; L'_m = L_m \\ \min\{1, \frac{\tilde{c}_{mi}(L_m)\mathcal{M}_{mi}(L'_m)}{\tilde{c}_{mi}(L'_m)p(\boldsymbol{z}_{mi}|\boldsymbol{\pi}_{h_{mi}}, \boldsymbol{c}_m)}\}, & j \leq L_m \;\&\; L'_m < L_m \\ \min\{1, \frac{\tilde{c}_{mi}(L_m)p(\boldsymbol{z}_{mi}|\boldsymbol{\pi}_j, \boldsymbol{c}_m)}{\tilde{c}_{mi}(L'_m)\mathcal{M}_{mi}(L_m)}\}, & j > L_m \end{cases}$$

where the normalizing constant is defined as $\tilde{c}_{mi}(L_m) = \sum_{|\boldsymbol{\epsilon}|=1}^{L_m}\theta_{m|\boldsymbol{\epsilon}|}p(\boldsymbol{z}_{mi}|\boldsymbol{\pi}_{\boldsymbol{\epsilon}}, \boldsymbol{c}_m) + \mathcal{M}_{mi}(L_m)(1-$

$\sum_{|\boldsymbol{\epsilon}|=1}^{L_m} \theta_{m|\boldsymbol{\epsilon}|}$). The retrospective sampling procedure for l_{mi} is summarized in Algorithm 1.

5 Experiments

We test the proposed model with five datasets: (i) a simulated, illustrative example that examines the ability to learn the tree structure; (ii) a subset of the MNIST digits data, (iii) face data (Tenenbaum et al., 2000); (iv) the Microsoft (MSRC) image database; and (v) the LabelMe data. In the case of (iv) and (v), the images are supplemented by annotations. For (ii)-(v), we process patches from each image. For the MNIST and face data, we randomly select 50 partially overlapping patches in each image, with 15×15 and 40×40 patch sizes, respectively (placement of patches was not tuned, selected uniformly at random, with partial overlap). For the MSRC and LabelMe data, we collect all $32 \times 32 \times 3$ non-overlapping patches from the *color* images (we also consider overlapping patches in this case, but it was found unnecessary). Recall that $\boldsymbol{x}_{mi} \in \mathbb{R}^P$, with P the number of pixels in each patch ($P = 225$ for MNIST, $P = 1600$ for the face data, and $P = 3072$ for MSRC and LabelMe data).

We have examined different methods for initializing the dictionary, including random draws from the prior and various fixed redundant bases, such as the over-complete DCT. Alternatively, we may use existing dictionary-learning methods (independent of the topic model); for this purpose, we use the covariate-dependent hierarchical beta process (with the covariates linked to the relative locations between patches) to learn an initial set of dictionary atoms (Zhou et al., 2011b). Additionally, in examples (ii)-(v), we initialize the tree with 4 levels. In this initialization, four nodes are present beneath the root node, and each subsequent node has two children, down four levels; nested K-means clustering is used to initialize the data among the clusters (nodes) at the respective levels.

In all experiments, the hyperparameters were set $a_0 = b_0 = 1$, $c_0 = d_0 = e_0 = f_0 = 10^{-6}$, $\alpha = 1$ and $\gamma = 1$, and the truncation level (upper bound) on the number of dictionary elements was $K = 400$; many related settings of these parameters yield similar results, and no tuning was performed.

5.1 Inferring the tree: simulated data

We first illustrate that the proposed model is able to infer both the depth and width of the tree, using synthesized data for which the tree that generates the data is known; the data are like (but distinct from) that considered in Figure 2 of Blei et al. (2003a). In this simple example we wish to isolate the component

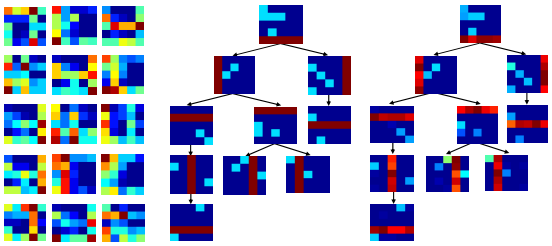


Figure 1: Example with synthesized images. Left: Example images. Middle: The ground truth for the underlying model. Right: The inferred model from the maximum-likelihood collection sample.

of the model that infers the tree, so there is no dictionary learning. The data consists of a 25-element alphabet, arranged as 5×5 blocks on a grid; each topic is characterized by the probability of using each of the 25 elements (there is a probability π_{hk} for using element $k \in \{1, \ldots, 25\}$, for each topic/node h). For each topic h the "truth" (middle in Figure 1) has probabilities 0 (blue), 0.1 (cyan blue), and 0.5 (red). The generative process for image m corresponds to first drawing a path $\boldsymbol{c}_m$ through the tree, and then 1000 times a node on this patch is drawn from $\boldsymbol{\theta}_m$, and finally each of the 25 alphabet members are drawn Bernoulli, with the associated topic-dependent probabilities (this is the proposed model, without dictionary learning). The final data ("image") consists of a count of the number of times each of the 25 elements was used, across the 1000 draws (example data at left in Figure 1). A total of 100 5×5 "images" were drawn in this manner to constitute the data. The right part of Figure 1 corresponds to the *recovered* tree, based upon the maximum-likelihood collection sample. Of course, the order of the branches and children is arbitrary; the inferred tree in Figure 1 (right) was arranged *a posteriori* to align with the "truth" (middle), to clarify presentation.

In this example the tree was initialized with 3 paths, each with three 3 layers (note in truth there are four paths, with variable number of layers). We also initialized the tree with 4 and 5 paths, under the same experimental setting, and similar recovery is achieved. If we initialized with less than 3 paths or more than 5, the recovered tree was still reasonable (close), but not as good. However, the inference of the topic-dependent usage probabilities $\boldsymbol{\pi}_h$ was very robust to numerous different settings. These results are based on 2000 samples, 1000 discarded as burn-in.

5.2 Model fit

For the MNIST handwritten digit database, we randomly choose 100 images per digit (digits 0 through

9), and therefore $M = 1000$ images are considered in total; the images are of size 28×28. The face dataset (Tenenbaum et al., 2000) contains $M = 698$ faces, each of size 64×64. Concerning the inferred trees, for the MNIST data, the maximum-likelihood collection sample had 168 paths and each path was typically 5 layers deep; for the face data 80 paths were inferred, and each was typically 5 layers. To quantitatively compare the ability of the hierarchical dictionary construction to fit the data, we consider reconstruction error for the data, comparing with the single-layer ("flat") model in Li et al. (2011); results are summarized in Table 1, corresponding to $\|\boldsymbol{x}_{mi} - \mathbf{D}(\boldsymbol{z}_{mi} \odot \boldsymbol{s}_{mi})\|_2^2$, averaged across all images m and patches i. In addition to results for MNIST and faces data, we show results for the MSRC and LabelMe data sets (analyzed in this case *without* annotations).

We also performed experiments to investigate how initialization affects the performance. In Table 1, instead of initializing the dictionary via the hierarchical beta process (hBP), they are initialized at random. While there is a slight degradation in performance with random initialization, it is not marked, and the results are still better than those produced by Li et al. (2011). Similar improvements in hBP initialization were observed for the classification task discussed below; hBP helps, but random initialization is still good.

Table 1: Reconstruction error comparison (mean square error multiplied by 10^3, and $\pm$ one standard deviation) on MNIST, Face, MSRC and LabelMe datasets. 'nDP+hBP' and 'nDP+random' correspond to the proposed model with the dictionary initialized by hBP and randomly, respectively, while the "flat" (single layer) model corresponds to Li et al. (2011).

	MNIST	Face	MSRC	LabelMe
flat model	11.10 ± 0.32	8.18 ± 0.17	10.27 ± 0.54	12.16 ± 0.30
nDP+hBP	10.42 ± 0.21	7.64 ± 0.12	8.64 ± 0.36	10.21 ± 0.27
nDP+random	10.85 ± 0.35	7.91 ± 0.20	9.25 ± 0.41	11.53 ± 0.50

The proposed model fits the data better than the "flat" model in Li et al. (2011); the gains are more evident when considering real and sophisticated imagery (the MSRC and LabelMe data). It is important to note that the proposed model is effectively no more complicated than the model in Li et al. (2011). Specifically, in Li et al. (2011) and in the proposed model, each image is put in a cluster, where in Li et al. (2011) a single-layer Dirichlet process was used to perform clustering, where here paths through the tree define clusters. In Li et al. (2011) and here each cluster/path is characterized by a distribution over topics, and in both models each topic is characterized by probabilities of atom usage (and each model has a truncation level on the dictionary of the same size). The difference is that in Li et al. (2011) the probabilities of topic usage for each cluster are drawn independently, where here the tree structure, and shared nodes between different paths, manifest statistical dependencies between the probability of topic usage in different paths with shared nodes.

In these examples a total of 250 Gibbs samples were run, with 150 discarded as burn-in. The results of the model correspond to averaging across the collection samples. In all examples useful results were found with a relatively small number of Gibbs samples.

5.3 Organizing MSRC data

We use the same settings of images and annotations from the MSRC data[1] as considered in Du et al. (2009), to allow a direct comparison. We choose 320 images from 10 categories of images with manual annotations available. The categories are "tree", "building", "cow", "face", "car", "sheep", "flower", "sign", "book" and "chair". The numbers of images are 45 and 35 in the "cow" and "sheep" classes, respectively, and 30 in all the other classes. Each image has size 213×320 or 320×213. For annotations, we remove all words that occur less than 8 times (approximately 1% of words), and obtain 15 unique annotation-words, thus $N_v = 15$.

The full tree structure inferred is shown in Figure 2, with its maximum depth inferred to be 6 (maximum-likelihood collection sample depicted, from 100 collection samples). On the second level, it is clearly observed that images are clustered in several main subgenres, *e.g.*, one with images containing grass, one with flowers, and another with urban construction, including cars and buildings, etc. For each path, we depict up to the 8 most-probable images assigned to it (fewer when less than 8 were assigned).

To further demonstrate the form of the model, two pairs of example images are shown in Figure 3. The pairs of images were assigned to two distinct paths, that shared nodes near the root (meaning the model infers shared types of patches between the images, assigned to these shared nodes). For each node, three example patches that are assigned to it are selected, from each of the two images. For the pair of example images from the "sheep" and "cow" classes, patches of grass and legs are shared on the top nodes, while distinct patches manifesting color and texture are separately assigned to nodes at bottom levels. The two images from the "building" class show the diversity of this category. It is anticipated that the "building" category will be diverse, with common patches shared at nodes near the root, and specialized patches near the leaves

[1] `http://research.microsoft.com/en-us/projects/objectclassrecognition/`

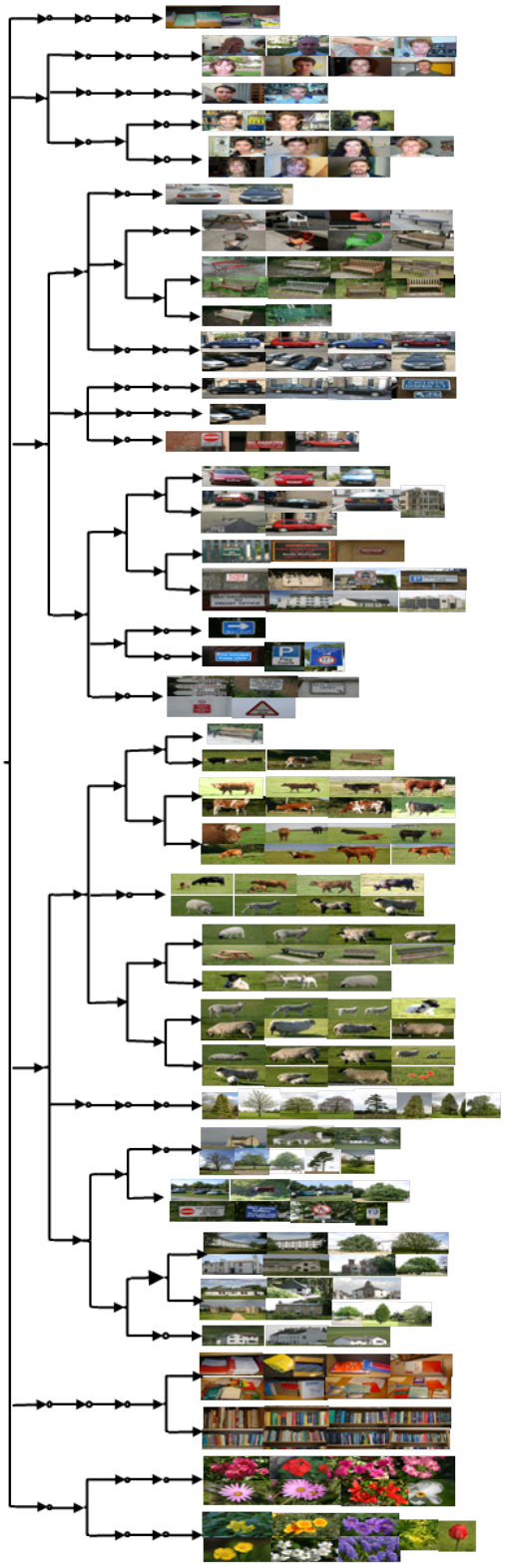


Figure 2: The full tree structure inferred from MSRC data. For each path, up to 8 images assigned to it are shown.

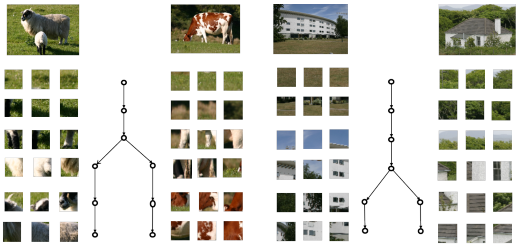


Figure 3: Two pairs of example images, and the paths they were assigned to. Between the images is shown the splitting paths. At top are the example images, and for each image we depict three example patches assigned to a respective node.

(for different types of buildings/structures). These typical examples illustrate that ubiquitous patches are shared at nodes near the root, with nodes toward the leaves describing details associated with specialized classes of images. It is this hierarchical structure that is missed by the model in Li et al. (2011), and that also apparently manifests the better model fit, as summarized in Table 1.

Another product of the model is a distribution over words for each path in the tree (not shown, for brevity). We illustrate this component of the model for the LabelMe data, considered next.

5.4 Organizing LabelMe data

The LabelMe data[2] contain 8 image classes: "coast", "forest", "highway", "inside city", "mountain", "open country", "street" and "tall building". We use the same settings of images and annotations as Wang et al. (2009): we randomly select 100 images for each class, thus the total number of images is 800. Each image is resized to be 256×256 pixels. For the annotations, we remove terms that occur less than 10 times, and obtain a vocabulary of 99 unique words, thus $N_v = 99$. There are 5 terms per annotation in the LabelMe data on average. Figure 6 visualizes 7 sub-trees of the inferred tree structure; there are 7 nodes inferred on the second level, and each node represents one sub-tree. Class "street" and class "insidecity" share the same root node, labeled 5 in Figure 6.

Based on the learned posterior word distribution $\boldsymbol{\psi}_p$ for the pth image class, we can further infer which words are most probable for each path. Figure 4 shows the $\boldsymbol{\psi}_p$ for 8 example paths (maximum-likelihood sample, from 100 collection samples), with the five largest-probability words displayed; the capital letters associated with each histogram in Figure 4 have associated paths through the tree as indicated in Figure 6. A good connection is manifested between the words and paths (examine the images and words associated with

[2] http://www.cs.princeton.edu/~chongw/

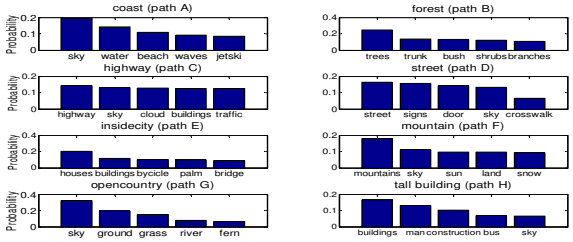
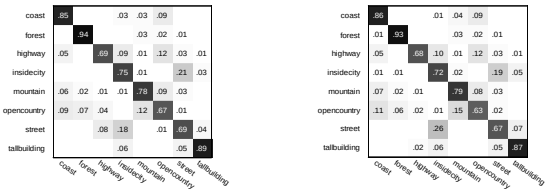


Figure 4: Inferred distributions over words for LabelMe data, as a function of inferred image category. The letters correspond to paths in Figure 6.



(a) (b)

Figure 5: For the 800 testing images from LabelMe data, (a): Confusion Matrix on original patches with the average accuracy of 78.3%. (b): Confusion Matrix on SIFT features with the average accuracy of 76.9%.

each path).

We evaluate the proposed model on the image-classification task, similar to as considered in Li et al. (2010). A set of 800 randomly selected images are held out as testing images from the 8 classes, each class with 100 testing images. Each image is represented by the estimated distribution over all the nodes in the entire hierarchy. Only nodes that are associated to the image have nonzero values in the distribution. We calculate the χ^2-distances between the node distribution of the testing images and those of the training images. The KNN algorithm (K is set to be 50) is then applied to obtain the class label.

Figure 5(a) shows the confusion matrix of classification, with an average classification accuracy of 78.3%, compared with 76% in Li et al. (2011). In all of the above examples the dictionary learning was applied directly to the observed pixel values within a given patch, with no *a priori* feature extraction. Alternatively, the patch-dependent data $\boldsymbol{x}_{mi}$ may correspond to features extracted using *any* image feature extraction algorithm. To illustrate this, we now let $\boldsymbol{x}_{mi}$ correspond to SIFT features (Lowe, 1999) on the same patches; in this experiment the dictionary learning replaces the VQ step in models like Wang et al. (2009). In Figure 5(b) we show the confusion matrix of the model based on SIFT features, with an average accuracy of 76.9%, slightly better than the results reported in Wang et al. (2009) (but here there is no need to tune the number of VQ codes). This also demonstrates that performing dictionary learning directly on the patches, rather than via a state-of-the-art feature extraction method, yields highly competitive results.

We now compare the proposed hierarchical model with the hierarchical model in Li et al. (2010), in which offline SIFT feature extraction and VQ are employed. Based on related work in Wang et al. (2009), we used a codebook of size 240, and achieved an average classification accuracy of 77.4%, compared with 79.6% reported above for our algorithm. Note, however, that we found the model in Li et al. (2010) to be very sensitive to the codebook size, with serious degradation in performance manifested with 150 or 400 codes, for example. To further test the proposed model, we considered the same classification experiment on MSRC data, which is characterized by 10 classes. Five images per class were randomly chosen as testing data, and the remaining images are treated as training data to learn a hierarchical structure. An average accuracy of 64% is obtained with the proposed model, compared with 60% using that in Li et al. (2010), where the codebook size is set to be 200. These experiments indicate that the proposed model typically does better than that in Li et al. (2010) for the classification task, even when we optimize the latter with respect to the number of codes.

The experiments above have been performed in 64-bit Matlab on a machine with 2.27 GHz CPU and 4 Gbyte RAM. One MCMC sample of the proposed model takes approximately 4, 2, 8 and 10 minutes respectively for the MNIST, Face, MSRC and LabelMe experiments (in which we simultaneously analyzed respectively 1000, 698, 320, and 800 total images). Note that while these model *learning* times are relatively expensive, model testing (after the tree and dictionary are learned) is very fast, this employed for the aforementioned classification task. To scale the model up to larger numbers of training images, we may perform variational Bayesian inference rather than sampling, and employ online-learning methods (Hoffman et al., 2010; Carvalho et al., 2010).

6 Conclusions

The nested Dirichlet process has been integrated with dictionary learning to constitute a new hierarchical topic model for imagery. The dictionary learning may be employed on the original image pixels, or on features from any image feature extractor. If words are available, they may be utilized as well, with word-dependent usage probabilities inferred for each path through the tree. The model infers both the tree depth and width. Encouraging qualitative and quantitative results have been demonstrated for analysis of many of the traditional datasets in this field, with comparisons provided to other related published methods.

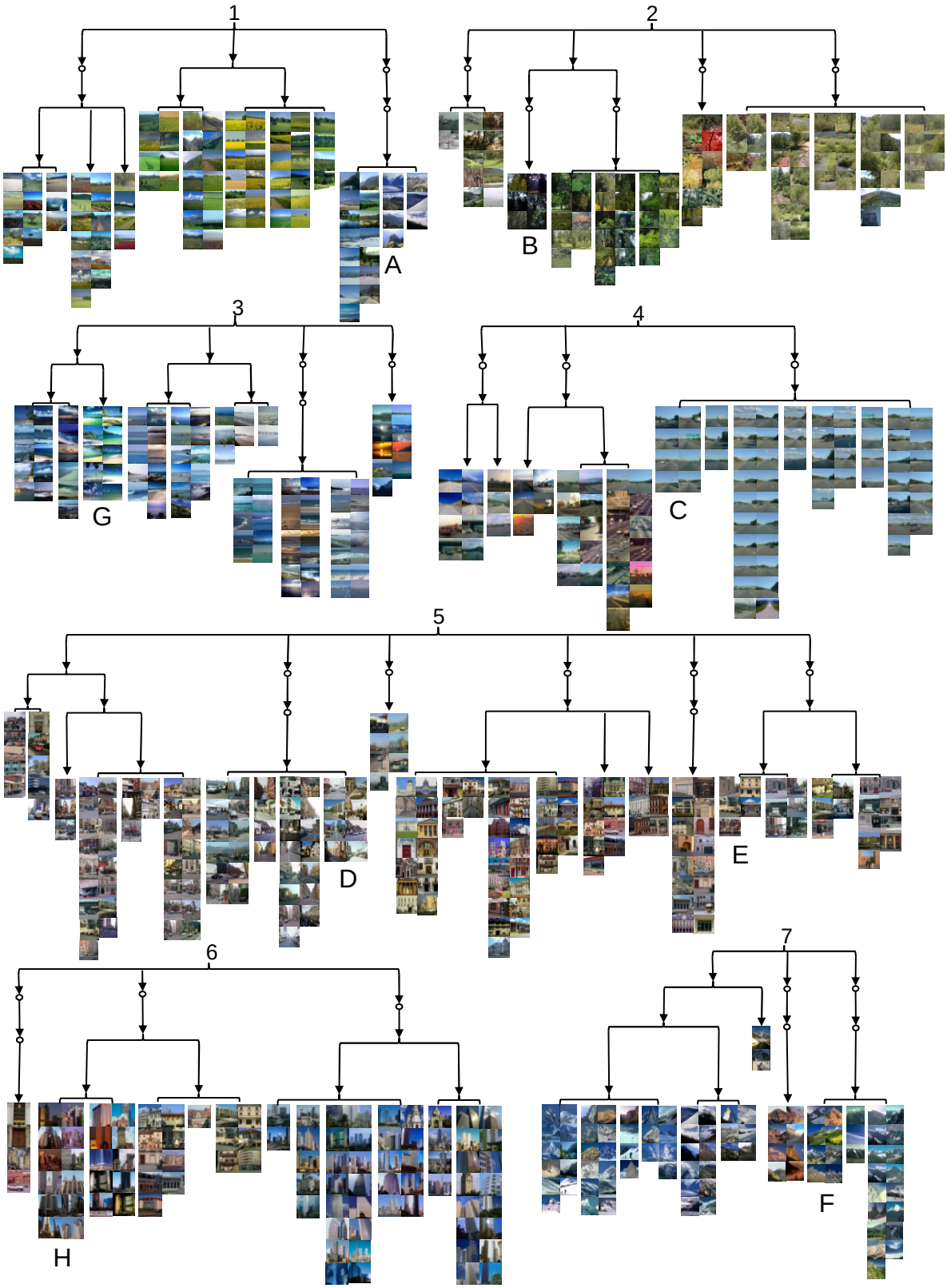


Figure 6: The structure of 7 sub-trees inferred from LabelMe data. The root of each sub-tree is one child node of the root for the entire tree structure. For each path, all images assigned to it are listed with no order importance. The letters refer to paths for which distributions over words are depicted in Figure 4.

Acknowledgement

The work reported here was supported by ARO, DOE, NGA, ONR and DARPA (under the MSEE Program).

References

Adams, R., Ghahramani, Z., and Jordan, M. Tree-structured stick breaking for hierarchical data. In *NIPS*, 2010.

Arbelaez, P. and Cohen, L. Constrained image segmentation from hierarchical boundaries. In *CVPR*, 2008.

Barnard, K., Duygulu, P., Forsyth, D., Freitas, N., Blei, D., and Jordan, M. Matching words and pictures. *JMLR*, 2003.

Bart, E., Porteous, I., Perona, P., and Welling, M. Unsupervised learning of visual taxonomies. In *CVPR*, 2008.

Blei, D. and Jordan, M. Modeling annotated data. In *SIGIR*, 2003.

Blei, D. and MaAuliffe, J. Supervised topic models. In *NIPS*, 2007.

Blei, D., Griffiths, T., Jordan, M., and Tenenbaum, J. Hierarchical topic models and the nested chinese restaurant process. In *NIPS*, 2003a.

Blei, D., Ng, A., and Jordan, M. Latent dirichlet allocation. *JMLR*, 2003b.

Carvalho, C.M., Lopes, H.F., Polson, N.G., and Taddy, M.A. Particle learning for general mixtures. *Bayesian Analysis*, 2010.

Du, L., Ren, L., Dunson, D., and Carin, L. Bayesian model for simultaneous image clustering, annotation and object segmentation. In *NIPS*, 2009.

Fei-Fei, L. and Perona, P. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.

Griffiths, T. L. and Ghahramani, Z. Infinite latent feature models and the indian buffet process. In *NIPS*, 2005.

Hoffman, M.D., Blei, D.M., and Bach, F. Online learning for latent dirichlet allocation. In *NIPS*, 2010.

Ishwaran, H. and James, L. Gibbs sampling methods for stick-breaking priors. *J. Amer. Stat. Assoc*, 2001.

Li, L., Zhou, M., Sapiro, G., and Carin, L. On the integration of topic modeling and dictionary learning. In *ICML*, 2011.

Li, L.-J. and Fei-Fei, L. What, where and who? classifying events by scene and object recognition. In *ICCV*, 2007.

Li, L.-J., Socher, R., and Fei-Fei, L. Towards total scene understanding: classification, annotation and segmentation in an automatic framework. In *CVPR*, 2009.

Li, L.-J., Wang, C., Lim, Y., Blei, D., and Fei-Fei, L. Building and using a semantivisual image hierarchy. In *CVPR*, 2010.

Lowe, D. Object recognition from local scale-invariant features. In *ICCV*, 1999.

Paisley, J. and Carin, L. Nonparametric factor analysis with beta process priors. In *ICML*, 2009.

Papaspiliopoulos, O. and Roberts, G. Retrospective markov chain monte carlo methods for dirichlet process hierarchical models. *Biometrika*, 2008.

Sivic, J., Russell, B., Zisserman, A., Freeman, W., and Efros, A. Unsupervised discovery of visual object class hierarchies. In *CVPR*, 2008.

Teh, Y. W., Görür, D., and Ghahramani, Z. Stick-breaking construction for the Indian buffet process. In *AISTATS*, volume 11, 2007.

Tenenbaum, J.B., Silva, V., and Langford, J.C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319-2323, 2000.

Thibaux, R. and Jordan, M. Hierarchical beta processes and the indian buffet process. In *AISTATS*, 2007.

Wang, C. and Blei, D. Variational infernece for the nested chinese restaurant process. In *NIPS*, 2009.

Wang, C., Blei, D., and Fei-Fei, L. Simultaneous image classification and annotation. In *CVPR*, 2009.

Yang, S.H., Bian, J., and Zha, H. Hybrid generative/discriminative learning for automatic image annotation. In *UAI*, 2010.

Zhang, X., Dunson, D., and Carin, L. Tree-structured infinite sparse fsvtor model. In *ICML*, 2011.

Zhou, M., Chen, H., Paisley, J., Ren, L., Li, L., Xing, Z., Dunson, D., Sapiro, G., and Carin, L. Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images. 2011a.

Zhou, M., Yang, H., Sapiro, G., Dunson, D., and Carin, L. Dependent hierarchical beta process for image interpolation and denoising. In *AISTATS*, 2011b.

Learning Mixtures of Submodular Shells with Application to Document Summarization

Hui Lin
University of Washington
hlin@ee.washington.edu

Jeff Bilmes
University of Washington
bilmes@ee.washington.edu

Abstract

We introduce a method to learn a mixture of submodular "shells" in a large-margin setting. A submodular shell is an abstract submodular function that can be instantiated with a ground set and a set of parameters to produce a submodular function. A mixture of such shells can then also be so instantiated to produce a more complex submodular function. What our algorithm learns are the mixture weights over such shells. We provide a risk bound guarantee when learning in a large-margin structured-prediction setting using a projected subgradient method when only approximate submodular optimization is possible (such as with submodular function maximization). We apply this method to the problem of multi-document summarization and produce the best results reported so far on the widely used NIST DUC-05 through DUC-07 document summarization corpora.

1 Introduction

Submodular functions [10] are those that satisfy the property of *diminishing returns*: given a finite ground set V, for any $A \subseteq B \subseteq V \setminus v$, a submodular function f must satisfy $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$, i.e., the incremental "value" of v decreases as the context in which v is considered grows from A to B. Submodular functions share a number of properties in common with convex and concave functions [29], including their wide applicability, their generality, their multiple options for their representation, and their closure under a number of common operators (including mixtures, truncation, complementation, and certain convolutions). For example, the weighted sum of a collection of submodular functions $\{f_i\}_i$, $f = \sum_i w_i f_i$ where w_i are nonnegative weights, is also submodular. While they have a long history in operations research, game theory, econometrics, and electrical engineering, they are still beginning to be studied in machine learning for a variety of tasks including sensor placement [19, 20], structure learning of graphical models [34], document summarization [27, 28] and social networks [18].

The problem of learning submodular functions has also recently been addressed. For example, in [15], it is asked "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where n is the ground set size and for what function $g : \mathbb{N} \to \mathbb{R}$ is the possible?". Among many results, they show that even with adaptive queries and monotone functions, one cannot learn better than an $\Omega(\sqrt{n}/\log n)$ approximation of a given fixed submodular function. Similarly, [1] addressed the submodular function learning problem from a learning theory perspective, given a distribution on subsets. They provide strong negative results including that one can not approximate in this setting to within a constant factor. In general, therefore, learning submodular functions is hard.

While learning over all possible submodular functions may be hard, this does not preclude learning submodular functions with known forms with unknown parameters. For example, given a finite set of M fixed submodular components $\{f_i\}_{i=1}^M$ where $f_i : 2^V \to \mathbb{R}$ is submodular over V, then learning a conical mixture $\sum_{i=1}^M w_i f_i$, where $(w_1, w_2, \ldots, w_M) = \boldsymbol{w} \in \mathbb{R}_+^M$, has not in the above been ruled out to have approximate guarantees. Such mixtures might span a very large set of submodular functions, depending on the diversity of the component set. We call such a problem "learning submodular mixtures."

In this paper we extend this one step further, to an approach we call learning "submodular shells." An instantiated submodular shell is a function $f_{\alpha,(V,\beta)} : 2^V \to \mathbb{R}$ indexed by a pair of parameter vectors α, β and a

ground set V. The parameters β are associated with a particular ground set V but the parameters α of the shell apply to *any* ground-set vector pair (V,β). A *shell* has the form $f_{\bar{\alpha},(\cdot,\cdot)}$, which does not have (V,β) instantiated. We might learn, for example, that a particular value $\bar{\alpha}$ produces a good shell $f_{\bar{\alpha},(\cdot,\cdot)}$ for any instantiation of that shell with a particular (V,β). In this sense, a submodular shell might be seen as a form of "structured submodularity." Moreover, we might learn $\boldsymbol{w}$ in a mixture of such shells $\sum_i w_i f_{\alpha_i,(\cdot,\cdot)}$ based on data consisting only of training tuples of the form $\{((V^{(t)},\beta^{(t)}),S^{(t)})\}_t$ where $S^{(t)} \subseteq V^{(t)}$ and which can be used in an objective that involves the instantiated mixture of shells. Note that here, training tuples may consist of a sequence of different ground sets, and the goal is from these training tuples, and given a finite and fixed set of shells parameterized by $\{\alpha_i\}$, identify the conical weights $\boldsymbol{w}$ that optimize an objective. Of course, if all training ground sets are identical, then learning submodular shell mixtures reduces to learning submodular mixtures. In practice, however, the training ground sets are not usually identical. E.g., in extractive document summarization, the training data could be a set of documents and the corresponding human summaries, where the ground sets, those sentences that constitute the documents, are not identical between training samples. In particular, for a document t, the ground set $V^{(t)} = \{1,\cdots,n_t\}$ where n_t is the number of sentences in document t, and $\beta^{(t)}$ could be the term-frequency vectors for the sentences (see Section 5.2 for more instances of this).

We introduce a method whereby submodular shell mixtures may be learnt in a max-margin structured-prediction setting, and provide a risk-bound guarantee under approximate submodular maximization (Theorem 1). We apply our method to the extractive document summarization problem and, as mentioned in the abstract, this yields extremely good results.

2 Structured Prediction

A main goal of learning is to identify a function $h : \mathcal{X} \to \mathcal{Y}$, that maps from an input domain $\mathcal{X}$ to an output domain $\mathcal{Y}$. Structured prediction problems [51, 50, 14] are those where these domains may consist of combinatorial structures. Often, given an input $\boldsymbol{x} \in \mathcal{X}$, only a subset of $\mathcal{Y}$, say $\mathcal{Y}_{\boldsymbol{x}}$, is valid. For example, if $\boldsymbol{x}$ is a sentence, and $\mathcal{Y}$ is the set of all parse trees, than only a subset of possible parse trees $\mathcal{Y}_{\boldsymbol{x}}$ might be valid. This offers little solace, however, as $\mathcal{Y}_{\boldsymbol{x}}$ is typically still exponentially large. Usually, one forms a score function $s : \mathcal{X} \times \mathcal{Y}$ that measures how good an output $\boldsymbol{y}$ is for a given input $\boldsymbol{x}$. The decision problem is, given an $\boldsymbol{x}$, find one of the outputs with the highest score, i.e., $\boldsymbol{y}^* \in \operatorname{argmax}_{\boldsymbol{y}\in\mathcal{Y}_{\boldsymbol{x}}} s(\boldsymbol{x},\boldsymbol{y})$.

One of the primary challenges of structured prediction is that one needs to handle $\mathcal{Y}_{\boldsymbol{x}}$, whose cardinality is finite but exponentially large. A common way to address this issue is to make assumptions on both $\mathcal{Y}$ and s. For instance, if s decomposes over the parts of $\mathcal{Y}$ and moreover only depends on "local" parts, one could employ dynamic programming or integer programming algorithm to find the optimal solution. Another approach is to explore combinatorial structures upon which efficient algorithms are available. Examples of this sort include the Hungarian method [21] for maximum weighted bipartite matching and the Chu-Liu-Edmonds algorithm [6, 9] for optimal branchings. Alternatively, as we do in the sequel, one could resort to approximate solutions with approximation guarantees.

3 Submodular Shell Scores

In this paper, we propose to explore the structured prediction problem using submodular shell score functions. That is, given $\boldsymbol{x}$, $s(\boldsymbol{x},\cdot) : \mathcal{Y}_{\boldsymbol{x}} \to \mathbb{R}$ is a submodular function where $\mathcal{Y}_{\boldsymbol{x}}$ is considered to be the finite ground set associated with $\boldsymbol{x}$ — there is a finite ground set $V_{\boldsymbol{x}}$ associated with $\boldsymbol{x}$ where $\mathcal{Y}_{\boldsymbol{x}} \subseteq 2^{V_{\boldsymbol{x}}}$. There are at least two benefits of using such submodular score functions. First, submodular functions are natural and expressive with the capability of modeling decisions beyond local or linear interactions among parts. That is, submodular functions may allow for global direct interactions over a structured object unlike score functions that must decompose in some way. Second, this expressive power does require prohibitive computation as would an arbitrary score function. The reason is that the submodular maximization problem, even under many constrained settings, can be solved efficiently and near-optimally with rigorous performance guarantees [36, 13, 25, 2, 11].

The hypothesis function we consider takes the following linear discriminant form:

$$h(\boldsymbol{x};\boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{y}\in\mathcal{Y}_{\boldsymbol{x}}} s(\boldsymbol{x},\boldsymbol{y}) = \operatorname*{argmax}_{Y\in\mathcal{Y}_{\boldsymbol{x}}} \boldsymbol{w}^\top \boldsymbol{f}_{\boldsymbol{x}}(Y)$$
$$= \operatorname*{argmax}_{Y\in\mathcal{Y}_{\boldsymbol{x}}} \sum_i w_i f_{\alpha_i,(V_{\boldsymbol{x}},\beta_{\boldsymbol{x}})}(Y)$$

where $\boldsymbol{w} \in \mathbb{R}_+$ is a non-negative weight vector, and $f_{\alpha,(V,\beta)} : 2^V \to \mathbb{R}$ is submodular over subsets of V for every valid value of α and (V,β). Since the weights are non-negative, the score function is also submodular. We call $f_{\alpha,(V,\beta)}$ a *submodular shell*, abstracting a set of submodular functions characterized by some α,β and a ground set V. The score function is called a *submodular shell mixture*, and each $f_{\alpha_i,(V,\beta)}$ is a *shell component* of the mixture. Note that the parameters β are associated with a particular ground set V but the parameters α of the shell apply to *any* pair of ground set V and β value.

While we discuss our learning process in Section 4, we introduce the learning problem here to improve clarity. We are given a set of training instances $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ drawn independently from a distribution D over pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ restricted in such a way that $\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}$ for any pair $(\boldsymbol{x}, \boldsymbol{y})$. $\boldsymbol{x}^{(t)}$ corresponds to a finite ground set $V_{\boldsymbol{x}^{(t)}}$ and a set of parameters $\beta_{\boldsymbol{x}^{(t)}}$ and so we say that $\boldsymbol{x}^{(t)} = (V_{\boldsymbol{x}^{(t)}}, \beta_{\boldsymbol{x}^{(t)}})$. We also have a finite collection of M submodular shells $\{f_{\alpha_i,(\cdot,\cdot)}\}_{i=1}^{M}$ each of which is instantiated into a submodular function for any $\boldsymbol{x} \in \mathcal{X}$. That is, $f_{\alpha_i,\boldsymbol{x}^{(t)}} : 2^{V_{\boldsymbol{x}^{(t)}}} \to \mathbb{R}$ is a submodular function. We are also given a loss function and $\ell_{\boldsymbol{x},\boldsymbol{y}}(\hat{\boldsymbol{y}})$ that measures the loss of predicting $\hat{\boldsymbol{y}} \in \mathcal{Y}_{\boldsymbol{x}}$ when the true label is $\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}$. The empirical risk minimization problem then is to find a conical mixture $\boldsymbol{w} \in \mathbb{R}_{+}^{M}$ such that the risk $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{x},\boldsymbol{y}}(h(\boldsymbol{x};\boldsymbol{w}))]$ is minimized. It is important to realize that what is learnt is the mixture coefficients $\boldsymbol{w}$ over a set of shells that may be instantiated into a weighted sum of submodular functions.

Before we discuss learning in detail, we consider the expressive power of mixtures of such submodular shells.

3.1 Submodular Shell Mixtures

A fairly rich subclass of submodular functions can be instantiated from submodular shell mixtures with components being simpler submodular shells. Consider, for example, truncation functions, mixtures of which can represent many submodular functions of interest. Given a modular function[1] $c : 2^V \to \mathbb{R}$, define a truncation function $t_{\alpha,(V,c)} : 2^V \to \mathbb{R}$ as

$$t_{\alpha,(V,c)}(A) = \min \left\{ c(A), \alpha \max_{B \subseteq V} c(B) \right\},$$

which is submodular, where $\alpha \in \mathbb{R}$ is the truncation threshold. A modular function can also be written as a truncation function with $\alpha = \infty$. We note that (V, β) above takes the form (V, c) here, and we might have a collection of such functions $\{t_{\alpha_i,(V,c)}\}_i$, and learning a mixture $\boldsymbol{w}$ would mean that w_i indicates the importance of truncation threshold α_i.

In fact, many submodular functions can be written as mixtures of truncation functions. For example, coverage type functions, canonical examples of submodular functions, can be written as submodular mixtures. Precisely, let a collection of sets be $\{A_i\}_{i \in \{1,\cdots,|V|\}}$ on a ground set E. Then the set cover function becomes

$$f : 2^V \to \mathbb{R}_+, \quad f(B) = \left| \bigcup_{i \in B} A_i \right|.$$

We can define costs $c_{i,j} = 1$ if A_i contains $j \in E$ and $c_{i,j} = 0$ otherwise. We then have

$$f(B) = \left| \bigcup_{i \in B} A_i \right| = \sum_{j=1}^{|E|} \min \left\{ \sum_{i \in B} c_{i,j}, 1 \right\},$$

which is a mixture of truncation functions. Moreover sums of concave functions applied to cardinality functions [47] can be represented as mixtures of truncation functions — any sum of such functions can be expressed as a sum of a modular function and nonnegative linear combinations of truncation functions.

Another interesting class is weighted matroid rank functions [32]. Given matroids [56, 38] $\mathcal{M}_i = (V, \mathcal{I}_i)$, and modular functions $m_i : 2^V \to \mathbb{R}_+$, we get:

$$\sum_{i=1}^{M} w_i f^i_{m_i,(V,\mathcal{M}_i)} = \sum_{i=1}^{M} w_i \max\{m_i(I) : I \subseteq S, I \in \mathcal{I}_i\}$$

which includes cover-like functions and many others.

The submodular functions introduced in [28] for document summarization can also been seen as a mixture of submodular shells, with components being either the coverage or the diversity shell that can be instantiated for a particular document.

In general, by using rich enough families of shell components, a submodular shell mixture could be very expressive, representing a very large family of submodular functions. Moreover, another advantage of using submodular shell mixture representations is that, since we assume each component is given and only the component weights are unknown, the learning problem can be addressed by using well-established methods. And by learning shell mixtures, we can then apply the learnt mixture to structured problems even over different underlying ground sets.

4 Learning Submodular Shell Mixtures

While there might be many ways of learning shell mixtures, in this paper we take a large-margin approach, standard in structured prediction. In other words, we want to minimize the risk of making predictions (decisions) when using the submodular shell mixture as a score function. Of course, the standard maximization in learning structured prediction is only approximate in this case, since maximizing submodular functions in NP-hard (although constant-factor approximable). Moreover, we need to identify a valid loss function that dose not violate submodularity. We wish also to ensure that the learnt parameters have quality guarantees (e.g., a risk bound). All of this is done in the below.

[1] f is modular if both f and $-f$ are submodular.

4.1 Large Margin Learning

We consider the learning problem defined in Section 3. For concision, we denote

$$\mathcal{Y}_t \triangleq \mathcal{Y}_{\boldsymbol{x}^{(t)}}, \quad \boldsymbol{f}_t(\boldsymbol{y}) \triangleq \boldsymbol{f}_{\boldsymbol{x}^{(t)}}(\boldsymbol{y}), \text{ and } \ell_t(\boldsymbol{y}) \triangleq \ell_{\boldsymbol{x}^{(t)},\boldsymbol{y}^{(t)}}(\boldsymbol{y}).$$

We follow the maximum margin approach [51] to learn the component weights $\boldsymbol{w}$, where the goal is to find a score function that scores $\boldsymbol{y}^{(t)}$ higher than all other $\boldsymbol{y} \in \mathcal{Y} \setminus \boldsymbol{y}^{(t)}$ by some margin. Formally, the learning problem is as follows:

$$\min_{\boldsymbol{w} \geq 0} \frac{1}{T} \sum_{t=1}^{T} \hat{\ell}_t(\boldsymbol{w}) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2, \tag{1}$$

where the generalized hinge loss

$$\hat{\ell}_t(\boldsymbol{w}) \triangleq \max_{\boldsymbol{y} \in \mathcal{Y}_t} \left(\boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y}) \right) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) \tag{2}$$

and a quadratic regularizer is minimized.

Algorithm 1: Projected subgradient descent for learning submodular shell mixtures.

Input : $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ and a learning rate sequence $\{\eta_t\}_{t=1}^{T}$.

$\boldsymbol{w}_0 = 0$;

for $t = 1, \cdots, T$ **do**

  Approximate loss augmented inference:

  $\hat{\boldsymbol{y}}_t \approx \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{Y}_t} \boldsymbol{w}_{t-1}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y})$;

  Compute the subgradient:

  $\boldsymbol{g}_t = \lambda \boldsymbol{w}_{t-1} + \boldsymbol{f}_t(\hat{\boldsymbol{y}}_t) - \boldsymbol{f}_t(\boldsymbol{y}^{(t)})$;

  Update the weights with projection:

  $\boldsymbol{w}_t = \max(\boldsymbol{0}, \boldsymbol{w}_{t-1} - \eta_t \boldsymbol{g}_t)$;

Return : the averaged parameters $\frac{1}{T} \sum_t \boldsymbol{w}_t$.

Many algorithms have been proposed for the large margin learning problem, including those based on the exponentiated gradient method [7], the dual extragadient method [52], the cutting-plane algorithm [54], and the subgradient descent method [42, 43]. We adopt a subgradient descent algorithm to learn submodular shell mixtures, as illustrated in Algorithm 1, where max of two vectors takes dimension-wise maximum, i.e. $\max\{\boldsymbol{a}, \boldsymbol{b}\} = (\max(a_1, b_1), \cdots, \max(a_n, b_n))$ where $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$.

Note, to preserve submodularity, the component weights must be non-negative. We thus simply project the weights to the non-negative orthant whenever doing the updates, and it is easy to show that updates followed by projection onto a non-negative region do not affect the convergence or correctness of the algorithm as when a point is projected back into the convex set, it is moved closer to every point in the set including the optimal points.

Second, whether the learning can be done efficiently depends on whether the so called *loss augmented inference* (LAI) problem,

$$\max_{\boldsymbol{y} \in \mathcal{Y}_t} \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y}), \tag{3}$$

can be solved efficiently. The LAI problem has a term that precisely matches the prediction problem whose parameters we are trying to learn but also has an additional term corresponding to the loss. Tractability of LAI therefore not only depends on the tractability of the prediction problem but also on the form of loss functions. When using submodular shell mixtures as the score function, we use approximate inference with a performance guarantee. Therefore, we must perform approximate inference for the LAI problem as well. We thus in general refer to this as approximate learning.

4.2 Approximate learning

Since some form of inference is a dominant subroutine in many learning algorithms for structured prediction, it is natural to use good approximate inference techniques to make the learning problem tractable. When learning submodular shell mixtures, we inevitably must use approximate learning since, by using a more expressive class of score functions that need not decompose, the inference problem is intractable to do exactly. Using approximate inference as a drop-in replacement for exact inference in learning, however, could mislead the learning algorithm and result in poorly learnt models. This is analyzed in [24], where it is pointed out that approximate learning could fail even with an approximate inference method having approximation guarantees. In general, therefore, it is problematic to assume that an arbitrary choice of approximate inference method will lead to useful results when the learning method expects exact feedback. Choosing compatible inference and learning procedures is therefore crucial.

In the following, we aim to leverage the approximation guarantees of submodular optimization such that the performance of approximate learning of submodular shell mixtures can be bounded in some way. One possible way of bounding is to investigate the degree to which we can approximate the parameters $\boldsymbol{w}$ that would be obtained by exact learning, since the parameters themselves offer little utility if good prediction cannot be made from them. Alternatively, we can focus on the quality of prediction obtained from an approximately learned model. In particular, we seek to bound the risk gap. That is, the difference between the expected loss of predictions from an approximate (but efficient) scheme and from exact (but intractable) methods.

4.3 Analysis

Hence, we must further analyze whether using good approximate inference will lead us to good approximate learning in our case. Before doing so, we note that there are two types of approximation inference algorithms, namely undergenerating and overgenerating approximations.

Consider a maximization problem

$$\max_{\boldsymbol{y}\in\mathcal{Y}} f(\boldsymbol{y}) \triangleq f^*.$$

Undergenerating approximation algorithm always find a solution $\boldsymbol{y} \in \mathcal{Y}$ such that $f(\boldsymbol{y}) \leq f^*$, while overgenerating approximation algorithm always find a solution $\boldsymbol{y} \in \bar{\mathcal{Y}} \supseteq \mathcal{Y}$ such that $f(\boldsymbol{y}) \geq f^*$. A greedy algorithm or the loopy belief propagation [40] are instances of undergenerating approximation algorithms. Relaxation methods, e.g. linear programming relaxation, are overgenerating approximation algorithms. Note that undergenerating algorithms usually produce solutions that are within the feasible region of the problem. Overgenerating algorithms, on the other hand, generate solutions that might lie outside this feasible region. Therefore, solutions found by overgenerating algorithms sometimes need to be mapped back to the feasible region (e.g., rounding of linear programming produced solutions) in order to produce a feasible solution, during which the approximation guarantee no longer holds in some cases [44]. For learning submodular shell mixtures, we are particularly interested in undergenerating algorithms since the greedy algorithm, one of the undergenerating algorithms, offers near-optimal solutions for submodular maximization under certain (e.g., cardinality, budget, and matroid) constraints [36, 13].

Generalization bounds for approximate learning with cutting-plane algorithms, with either undergenerating or overgenerating inferences, have been shown in [12]. For subgradient descent methods, generalization analysis is available, but only for overgenerating cases, in [30, 22]. As far as we know, no generalization analyses are available for approximate learning with undergenerating subgradient methods. We fill this gap and offer risk bounds for approximate learning with undergenerating subgradient methods.

We need two definitions before moving on.

Definition 1 (ρ-approximate algorithm)**.** *Given $f : \mathcal{Y} \to \mathbb{R}^+$ and a maximization problem $\max_{\boldsymbol{y}\in\mathcal{Y}} f(\boldsymbol{y}) \triangleq f^*$, we call an (undergenerating) algorithm a ρ-approximate algorithm if it finds a solution $\boldsymbol{y} \in \mathcal{Y}$ such that $f(\boldsymbol{y}) \geq \rho f^*$, where $0 \leq \rho \leq 1$.*

Definition 2 (γ-approximate subgradient)**.** *Given $f : \mathbb{R}^M \to \mathbb{R}$, a vector $\boldsymbol{g} \in \mathbb{R}^M$ is called a γ-subgradient of f at $\boldsymbol{w}$ if for all $\boldsymbol{w}'$, $f(\boldsymbol{w}') \geq f(\boldsymbol{w}) + \boldsymbol{g}^\top(\boldsymbol{w}' - \boldsymbol{w}) - \gamma f(\boldsymbol{w})$ where $0 \leq \gamma \leq 1$ and $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^M$.*

In Algorithm 1, one needs to solve the LAI in Eqn (3). Note that if ℓ is modular (e.g. hamming loss) or submodular (e.g. see Section 5.3), a ρ-approximate inference algorithm can also apply to this loss augmented inference to find a near-optimal solution efficiently. However, as mentioned above, when an approximate inference algorithm is used in a learning algorithm, a good approximation of the score might not be sufficient, and it is possible that the learning can fail even with rigorous approximate guarantees [24]. On the other hand, Ratliff et al. [43] show that the subgradient algorithm is robust under approximate settings, and the risk experienced during training with γ-approximate subgradients can be bounded.

Note that a ρ-approximate LAI does not necessary imply any γ-subgradient because the approximate ratio does not apply to the term $-\boldsymbol{w}^\top f_t(\boldsymbol{y}^{(t)})$. To analyze the actual impact of ρ-approximate LAI in the learning procedure when compared with the exact formulation, then, we provide risk bounds for the approximate learner in Theorem 1.

Theorem 1. *Assume $w_i, f_i, i = 1, \cdots, M$ are all upper-bounded by 1, $\hat{\ell}_t(\boldsymbol{w}) \leq B$, and $\|\boldsymbol{g}_t\| \leq G$. Let $\boldsymbol{w}^*$ and $\hat{\boldsymbol{w}}$ be the solutions returned by Algorithm 1 using exact and ρ-approximate LAI, respectively, with learning rate $\eta_t = \frac{2}{\lambda t}$ and $\lambda = \frac{G}{M}\sqrt{\frac{2(1+\log T)}{T}}$. Then for any $\delta > 0$ with probability at least $1 - \delta$,*

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{y}}(h(\boldsymbol{x};\hat{\boldsymbol{w}}))] \leq \frac{1}{\rho}\left(\frac{1}{T}\sum_{t=1}^{T}\hat{\ell}_t(\boldsymbol{w}^*)\right) + S(T),$$

where

$$S(T) = \frac{MG}{\rho}\sqrt{\frac{2(1+\log T)}{T}} + B\sqrt{\frac{2}{T}\log\frac{1}{\delta}} + \frac{1-\rho}{\rho}M.$$

The proof is given in the supplementary material. Note that there are three terms in $S(T)$. While the first two terms vanish as $T \to \infty$, the third term does not. Therefore, the additional risk incurred due to the use of ρ-approximate LAI for learning is $(R^* + (1-\rho)M)/\rho$, where $R^* = \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\hat{\ell}_t(\boldsymbol{w}^*)$ and can be seen as the risk of predictions using models learnt with exact inference. Thus, the better (larger ρ) the approximation, the less the additional risk there will be when using an approximate LAI. And when exact inference is used ($\rho = 1$), the additional risk shrinks to zero. Note that Theorem 1 applies to any loss function and any score function which is not necessary to be submodular. When addition assumptions are made (e.g., assumption that the loss function is linearly realizable as in [57]), a better bound might be possible where the additional risk shrinks to zero as T grows. On the other hand, when the LAI objective is monotone submodular, a

simple and efficient greedy algorithm performs near-optimally with approximation factor $1 - 1/e$ [36]. In practice, moreover, as the approximation factor of the greedy algorithm on submodular maximization is usually very close to 1 [27], one could expect very little additional risk when using Algorithm 1 with approximate inference on learning submodular shell mixtures.

To the best of our knowledge, Theorem 1 is the first approximate learning bound for subgradient algorithms with undergenerating (greedy) inference.

5 Application to Document Summarization

Submodular mixtures could be applied to many structured prediction problems of practical interest. In this paper, we apply submodular shell mixture learning to extractive document summarization as a case study.

5.1 Submodularity in Document Summarization

Extractive document summarization can be seen as a subset selection problem [27]. Given a ground set of sentences V, the task of extractive document summarization is selecting a subset of sentences, say S, that best represents the whole document. In other words, we want to find $A \subseteq V$ such that

$$A \in \operatorname*{argmax}_{B \subseteq V} f(B) \text{ subject to: } \sum_{i \in B} c_i \leq b, \qquad (4)$$

where $c_i \in \mathbb{R}^+$ is the cost of sentence i (e.g., it could be the number of words in the sentence), $b \in \mathbb{R}^+$ is the total budget (e.g., it could the largest number of words allowed in a summary), and $f : 2^V \to \mathbb{R}$ is a set function that models the quality of a summary. Eqn (4) is known as the problem of submodular maximization subject to knapsack constraints [31] which NP-complete [35]. However, when f is *monotone submodular*, Eqn (4) can be solved efficiently and near-optimally with a theoretical guarantee via greedy algorithms [48, 27].

One can always force f to be submodular, leading to an objective function that can be optimized well but might on the other hand poorly represent a given problem. One attractive property of submodularity, like convexity in continuous domain, is that it arises naturally in many applications. One such applications is document summarization. As pointed out in [28], many well-established methods, including the widely used maximum margin relevance method [3], actually correspond to submodular optimization. Moreover, it is shown that the commonly used ROUGE score [26] for automatic summarization evaluation is monotone submodular [28], giving further evidence that submodular functions are natural for document summarization.

In this paper, we further show that not only is the ROUGE score submodular, the score used in the Pyramid method [37], one of the manual evaluation metrics that has been used in recent TAC summarization track[2], is also monotone submodular.

Theorem 2. *The modified score in Pyramid method is monotone submodular.*

The proof is in Appendix C in the supplement.

The remaining question is how to design (or ideally learn) a good submodular function for summarization. Lin and Bilmes [28] proposed a class of submodular functions that models the coverage as well as the diversity of summary. In this paper, we further generalize their class of submodular functions and propose to use *submodular shell mixtures* for document summarization.

5.2 Submodular shells for summarization

Diversity shell components

We define a diversity shell component as

$$f^{\text{diversity}}_{(a,K,\mathcal{A}),(V,\boldsymbol{r})}(S) = \frac{\sum_{k=1}^{K} \left(\sum_{i \in S \cap P_k} r_i\right)^a}{\sum_{k=1}^{K} \left(\sum_{i \in P_k} r_i\right)^a}, \qquad (5)$$

where $0 \leq a \leq 1$ is the curvature, $K \in \mathbb{Z}^+$ is the number of clusters (partitions), $\mathcal{A}$ is a clustering algorithm, and $\{P_k\}_{k=1,\cdots,K}$ is a partition of the ground set V generated by $\mathcal{A}$, and $\boldsymbol{r} = \{r_i\}_{i=1}^{|V|}$ with $r_i \in [0, 1]$ is the vector of singleton reward of element $i \in V$. The diversity component models the diversity of a summary set S, by diminishing the benefit of choosing elements from the same cluster.

Note that the α parameter of a submodular shell here takes the form $(a, K, \mathcal{A})$. By using different values of a and K, and different clustering algorithms $\mathcal{A}$, we can produce a variety of submodular shells. The (V, β) parameter of a submodular shell takes the form of $(V, \boldsymbol{r})$. When a document (ground set) is given, rewards of each sentence (i.e., r_i) can be computed, and the diversity shell component is then instantiated into a submodular function that measures the diversity of a summary for this particular document.

Clustered facility location shell components

We define clustered facility-location like components as

$$f^{\text{c-facility}}_{(K,\mathcal{A}),(V,\boldsymbol{r})}(S) = \frac{1}{K} \sum_{k=1}^{K} \max_{i \in S \cap P_k} r_i, \qquad (6)$$

[2] http://www.nist.gov/tac/2011/Summarization/

where $K \in \mathbb{Z}^+$ is the number of clusters (partitions), $\mathcal{A}$ is a clustering algorithm, and $r_i \in [0,1]$ is the singleton reward of element $i \in V$. This function has a similar form to the well known submodular facility location function, but defined on a partition of the ground set. We thus call it clustered facility location. If a summary contains multiple elements from a same cluster, the element with largest singleton reward will be regarded as the "representative" of this cluster, and only the reward of this representative will be counted into the final score. This again diminishes returns of choosing elements from the same cluster and therefore $f^{\text{c-facility}}$ is submodular.

Fidelity shell components

Given a ground set V, we define fidelity components as

$$f^{\text{fidelity}}_{\alpha,\left(V,\{C_i\}_{i=1}^{|V|}\right)}(S) = \frac{1}{|V|}\sum_{i\in V}\min\left\{\frac{\mathcal{C}_i(S)}{\mathcal{C}_i(V)},\alpha\right\}, \quad (7)$$

where $0 < \alpha \leq 1$ is a saturation threshold and $\mathcal{C}_i : 2^V \to \mathbb{R}$ is a *monotone* submodular function modeling how S covers the information contained in i. This function is the normalized version of the coverage function defined in [28]. Basically, the saturation threshold controls how much of a given element $i \in V$ should be covered; once $\mathcal{C}_i(S)$ is large enough such that the ratio of it over its largest possible value ($\mathcal{C}_i(V)$) is above threshold, covering more of i does not further increase the function value. Therefore, a larger value of f^{fidelity} tends to have more $i \in V$ well covered. When a document is given, we can instantiate different submodular shells using a variety of C_i.

5.3 A Submodular Loss Function

The most widely used evaluation criteria for summarization is the ROUGE score, which is basically a submodular function that counts n-gram recall rate over human summaries. Let S be the candidate summary (a set of sentences extracted from the ground set V), $c_e : 2^V \to \mathbb{Z}_+$ be the number of times n-gram e occurs in summary S, and R_i be the set of n-grams contained in the reference summary i (suppose we have K reference summaries, i.e., $i = 1, \cdots, K$). Then ROUGE-N [26] can be written as the following set function:

$$f_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K}\sum_{e\in R_i}\min(c_e(S), r_{e,i})}{\sum_{i=1}^{K}\sum_{e\in R_i} r_{e,i}},$$

where $r_{e,i}$ is the number of times n-gram e occurs in reference summary i. $f_{\text{ROUGE-N}}(S)$ is submodular, as shown in [28], but cannot be used as a loss function since it basically measures "accuracy" rather than loss.

An alternative is to use $1 - f_{\text{ROUGE-N}}(S)$ as a loss function, but this is supermodular. Note that in order to have the risk of the approximated learned model bounded, performance guarantees are required for the approximation algorithms used in loss augmented inference. When using $1 - f_{\text{ROUGE-N}}$, which is supermodular as a loss function, in the objective function for loss augmented inference (Eqn. (3)) along with a submodular shell mixture as the score function, the resulting objective function for LAI is then a submodular function plus a supermodular function. While an algorithm (e.g., submodular-supermodular procedure [33, 17]) is available to approximately optimize the sum of a submodular function and a supermodular function, performance guarantees usually do not exist for these algorithms (although this strategy might work well in practice and should ultimately be tested). Therefore, when using one-minus-ROUGE as the loss function, the greedy algorithm no longer provides a near-optimal solution when applied to the non-submodular objective, and the risk bound shown in Theorem 1 no longer holds.

To address this issue, we propose a ROUGE-like loss function that measures the "complement recall":

$$\ell_{\text{ROUGE}}(S) \triangleq \frac{\sum_{e\in\bar{R}}\omega_e c_e(S)}{\sum_{e\in\bar{R}}\omega_e r_e}, \quad (8)$$

where $\bar{R} = N \backslash \bigcup_i R_i$, and N is the set of all the n-grams occur in the set of documents, and $r_e = c_e(V)$ is the number of times n-gram e occurs in all the documents, ω_e is a non-negative weight for e, and $\bar{R}$ is the set of n-grams that are *not* covered by any human reference summary. Instead of counting with respect to a reference summary, ℓ_{ROUGE} counts the n-grams of a candidate summary S w.r.t. the *complement* of reference summaries.

Intuitively, we want a summary S to cover as many reference n-grams as possible so that it will get a high ROUGE-score; this is similar to having S be large and overlapping as little as possible with the n-grams that are *not* in human references. In this sense, ℓ_{ROUGE} measures the portion of how many n-grams in the complement of the reference n-grams set are covered, and when comparing summaries with the same size, the smaller ℓ_{ROUGE} is, the better. The best case, i.e., the human reference itself, will have ℓ_{ROUGE} equal to 0.

Obviously, a poor summary that would also have ℓ_{ROUGE} equal to 0 is an empty summary. It is worth noting that ℓ_{ROUGE} only makes sense when comparing summaries that are close to the same budget. Fortunately, most summarization algorithms try to consume every bit of the budget in order to consume as much information as possible under the budget constraint. For summaries produced in this way, ℓ_{ROUGE} offers a fair indicator of their quality: the smaller the loss value, the larger the number of reference n-gram overlaps there are, and therefore the better the summary. We

use the greedy algorithm for solving the summarization problems (Eqn. (4)). Due to its greedy nature, the algorithm always outputs summary candidates whose costs are close to the budget, and therefore ℓ_{ROUGE} can serve as a reasonable surrogate loss function for learning submodular shell mixtures for summarization tasks.

The major advantage of using ℓ_{ROUGE} as a loss function, of course, is algorithmic. The submodular learning analysis in Theorem 1 relies on the fact that a ρ-approximation algorithm is available for the loss augmented information. Since we use a submodular score function for summarization, the above objective will be submodular if the loss function is submodular. Fortunately, similar to $f_{\text{ROUGE-N}}$, the proposed loss for summarization, ℓ_{ROUGE}, is also monotone submodular (in fact modular). Therefore, the LAI in submodular shell mixture learning for summarization is exactly the budgeted submodular maximization problem (Eqn (4)), and efficient and near-optimal algorithms are then available. Consequently, all the theoretical analyses in Section 4.3 apply.

We will soon see (Section 7) that using the ROUGE-like loss function proposed here empirically outperforms using $1 - f_{\text{ROUGE}}$ as a loss functions.

6 Related work

Recently, Yue and Guestrin [57] studied a linear submodular bandits problem in an online learning setting for optimizing a general class of feature-rich submodular utility models in diversified retrieval, and their theoretical result is based on the assumption that the award function has a special (linear) form. Raman et al. [41] also propose an online learning model and algorithm for learning rankings that exploiting feedback to maximize any submodular utility measure. While our approach and analysis could also be applied to the online setting, the focus and analysis in this paper are more on the batch learning. The submodular utility models in [57, 41], however, can both be seen as special instances of submodular shell mixtures.

More closely related to our work is that of [46] where large-margin learning of submodular score functions for extractive document summarization is studied. The representation of a submodular score function in [46], again, turns out to be a special case of a submodular shell mixture. Moreover, our submodular shell mixture framework is more general and flexible than the framework proposed in [46]. Although the authors claim that their approach applies to all submodular summarization models, there are many submodular functions useful for summarization that are not linear on either pairwise similarity or singleton importance. For example, in the diversity reward function, we have a concave function over the sums of singleton rewards, and even if using linear model for singleton rewards, the score function is non-linear over parameters since the weights can not be linearly extracted, and thus the algorithms in [46] do not apply. Viewing each feature as input to a submodular component, on the other hand, preserves the linearity on the parameters, whereas Algorithm 1 applies. Moreover, representing a score function using a submodular shell mixture is expressive, as we have shown in Section 3.1.

In [46], a cutting plane algorithm with one-minus-ROUGE F-measure as loss function was used to learn model weights. When doing the loss augmented inference, they use the greedy algorithm introduced in [27] to approximately optimize the objective. Their objective function, however, is not submodular, due to the non-submodular loss function they use. Therefore, the LAI inference is no longer guaranteed to be near-optimal, and the performance of approximate learning with their cutting plane algorithm is no longer guaranteed [12]. Also, Sipos et al. [46] apparently neglect the fact that the learned similarity (or importance) should be non-negative, which is a necessary ingredient to preserve the submodularity of the learned score function. One simple way to ensure this is to constrain the weights to be non-negative, as we do for submodular shell mixtures by project the weights to the non-negative orthant (Algorithm 1).

7 Experiments

We evaluated our approach on NIST's DUC[3] data 2003-2007, and demonstrate results on both generic and query-focused summarization. DUC data were created by national institution of standard technology (NIST). The DUC evaluation is one of the standardized benchmark evaluations for document summarization and researchers continue to publish results on these data sets.

7.1 Query-independent summarization

Table 1: ROUGE-1 recall (R) and F-measure (F) results (%) on DUC-04. DUC-03 was used as development set.

DUC-04	R	F
Takamura and Okumura [49]	38.50	-
Wang et al. [55]	39.07	-
Lin and Bilmes [27]	-	38.39
Lin and Bilmes [28]	39.35	38.90
Kulesza and Taskar [23]	38.71	38.27
Best system in DUC-04 (peer 65)	38.28	37.94
Submodular Shell Mixture	**40.43**	**39.78**

The summarization tasks in DUC-03 and DUC-04 are

[3] http://duc.nist.gov/

generic summarization tasks. We used DUC-03 as the training set for submodular shell mixture learning. There are in total 60 document clusters in the DUC-03 task, therefore we have 60 training examples in total. We used a submodular shell mixture with 15 fidelity components for this task. In particular, the $\mathcal{C}_i$ function we used is $\mathcal{C}_i(S) = \sum_{j \in V} \delta_{i,j}$, where $\delta_{i,j}$ is the pairwise sentence similarity between sentence i and j. We used three types of sentence similarities. The first two are cosine similarities with unigram and bigram TF-IDF vectors respectively. The third similarity is again cosine similarity but on the vector generated by latent semantic analysis. For each of the similarity measures, we use five different saturation thresholds ($\alpha = 0.01, \cdots, 0.05$), and thus we have 15 fidelity components in total. We used Algorithm 1 with the ROUGE-like loss (Eqn. (8)) with $\omega_e = 1$ for all e to learn this submodular shell mixture. The ROUGE-1 results are shown in Table 1. As we can see, the result of the learned submodular shell mixture significantly outperforms all other previous reported results. Note that the experiment in [46] used a non-standard setup where DUC-04 data were divided into training, development, and test sets with only 5 documents in their test set. Therefore, results reported in [46] are not directly comparable to other results (e.g., our results) that follow standard DUC evaluation setup.

7.2 Query-focused summarization

Table 2: ROUGE-2 recall (R) and F-measure (F) results on DUC-05 (%). We used DUC-06 and DUC-07 as training sets.

DUC-05	R	F
Daumé III and Marcu [8]	6.98	-
Lin and Bilmes [28]	7.82	7.72
Best system in DUC-05 (peer 15)	7.44	7.43
Submodular Shell Mixture	**8.44**	**8.39**

Table 3: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-06, where DUC-05 and DUC-07 were used as training sets.

DUC-06	R	F
Celikyilmaz and Hakkani-tür [4]	9.10	-
Shen and Li [45]	9.30	-
Lin and Bilmes [28]	9.75	9.77
Best system in DUC-06 (peer 24)	9.51	9.51
Submodular Shell Mixture	**9.92**	**9.93**

Since 2004, DUC summarization evaluations have concentrated on query-focused summarization. We tested our approach for summarization on DUC-05, DUC-06 and DUC-07 data. In particular, we used DUC-06,07 as the training set for the DUC-05 task, DUC-05,07 as the training set for the DUC-06 task, and DUC-05,06 as the training set for the DUC-07 task.

Table 4: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-07. DUC-05 and DUC-06 were used as training sets. Note that the peer 15 system in DUC-07 used commercial web search engine to expand the queries.

DUC-07	R	F
Toutanova et al. [53]	11.89	11.89
Haghighi and Vanderwende [16]	11.80	-
Celikyilmaz and Hakkani-tür [4]	11.40	-
Lin and Bilmes [28]	12.38	12.33
Best system in DUC-07 (peer 15)	12.45	12.29
Submodular Shell Mixture	**12.51**	**12.40**

We used diversity components, clustered facility location components, and fidelity components to form the submodular shell mixture for query-focused summarization. Three clusterings with different numbers of clusters were created ($K = 0.1|V|, 0.2|V|, 0.3|V|$). As for the singleton rewards, we used both query-independent and query-dependent singleton rewards. The query-independent reward for i is simply the summation of the pairwise similarities of other elements in V to i. For the query-dependent reward, we simply used the number of terms (up to a bi-gram) that sentence j overlaps the query Q, where the IDF weighting is not used (i.e., every term in the query, after stop word removal, was treated as equally important). Therefore, in total, we have 6 clustered facility location components. We further used three curvatures in diversity components ($\alpha = 0.5, 0.6, 0.7$), which gives 18 diversity components in total. With one additional fidelity component, we have 25 components in total. Compared to other results reported in the literature (Table 2, Table 3 and Table 4), as far as we know, our approach achieves the best results reported so far on DUC-05, DUC-06, and DUC-07.

8 Conclusions

In this paper, we propose the notion of learning submodular shells, which abstract a set of submodular functions that can be instantiated into submodular functions given the input of a structured prediction task. Given a set of training instances, we use subgradient descent to learn the mixture coefficients over a set of submodular shells that may be instantiated into a weighted sum of submodular functions. We show that the submodular shell mixture is very expressive, and the risk of learning can be bounded when only approximate inference in possible. When applied to the task of document summarization, our approach achieves the best results reported so far on standardized benchmark takes for query-focused extractive summarization tasks.

References

[1] M.F. Balcan and N.J.A. Harvey. Learning submodular functions. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 793–802. ACM, 2011.

[2] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *Proc. of 12th IPCO*, pages 182–196. Citeseer, 2007.

[3] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. of SIGIR*, 1998.

[4] A. Celikyilmaz and D. Hakkani-tür. A hybrid hierarchical model for multi-document summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 815–824, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[5] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. In *NIPS*, pages 359–366, 2001.

[6] Y.J. Chu and T.H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14(1396-1400):270, 1965.

[7] P.L.B.M. Collins and B.T.D. McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in neural information processing systems*, volume 17, page 113, 2004.

[8] H. Daumé III and D. Marcu. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, page 312, 2006.

[9] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 1967.

[10] J. Edmonds. *Combinatorial Structures and their Applications*, chapter Submodular functions, matroids and certain polyhedra, pages 69–87. Gordon and Breach, 1970.

[11] U. Feige, V. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. In *Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.

[12] T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *Proceedings of the 25th international conference on Machine learning*, pages 304–311. ACM, 2008.

[13] ML Fisher, GL Nemhauser, and LA Wolsey. An analysis of approximations for maximizing submodular set functions—II. *Polyhedral combinatorics*, pages 73–87, 1978.

[14] L. Getoor and B. Taskar. *Introduction to statistical relational learning*. The MIT Press, 2007.

[15] M.X. Goemans, N.J.A. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 535–544. Society for Industrial and Applied Mathematics, 2009.

[16] A. Haghighi and L. Vanderwende. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[17] Rishabh Iyer and Jeff Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. In *uai*, Catalina Island, USA, July 2012. AUAI.

[18] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th Conference on SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.

[19] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proc. of Uncertainty in AI*, 2005.

[20] A. Krause, H.B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9: 2761–2801, 2008.

[21] H.W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[22] A. Kulesza. Approximate learning for structured prediction problems. *UPenn WPE-II Report*, 2009.

[23] A. Kulesza and B. Taskar. Learning determinantal point processes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

[24] A. Kulesza, F. Pereira, et al. Structured learning with approximate inference. *Advances in neural information processing systems*, 20:785–792, 2007.

[25] J. Lee, M. Sviridenko, and J. Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Approximation, Ran-*

domization, and Combinatorial Optimization. Algorithms and Techniques, pages 244–257, 2009.

[26] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 2004.

[27] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *North American chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL/HLT-2010)*, Los Angeles, CA, June 2010.

[28] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, Portland, OR, June 2011.

[29] L. Lovász. Submodular functions and convexity. *Mathematical programming-The state of the art,(eds. A. Bachem, M. Grotschel and B. Korte) Springer*, pages 235–257, 1983.

[30] A.F.T. Martins, N.A. Smith, and E.P. Xing. Polyhedral outer approximations with application to natural language parsing. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009.

[31] R. McDonald. A study of global inference algorithms in multi-document summarization. *Lecture Notes in Computer Science*, 4425:557, 2007.

[32] K. Murota. Recent developments in discrete convex analysis. *Research Trends in Combinatorial Optimization*, pages 219–260, 2009.

[33] M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Proc. Conf. Uncertainty in Artifical Intelligence, Edinburgh, Scotland, July. Morgan Kaufmann Publishers*, 2005.

[34] Mukund Narasimhan and Jeff Bilmes. PAC-learning bounded tree-width graphical models. In *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI-2004)*. Morgan Kaufmann Publishers, July 2004.

[35] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

[36] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, 14(1):265–294, 1978.

[37] A. Nenkova and R. Passonneau. Evaluating content selection in summarization: The pyramid method. In *Proceedings of HLT-NAACL*, volume 2004, pages 145–152, 2004.

[38] J. Oxley. *Matroid theory*, volume 21. Oxford Univ Pr, 2011.

[39] R.J. Passonneau, A. Nenkova, K. McKeown, and S. Sigelman. Applying the pyramid method in duc 2005. In *Proceedings of the Document Understanding Conference (DUC 05), Vancouver, BC, Canada*, 2005.

[40] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[41] K. Raman, P. Shivaswamy, and T. Joachims. Learning to diversify from implicit feedback. In *WSDM Workshop on Diversity in Document Retrieval*, 2012.

[42] N. Ratliff, J.A. Bagnell, and M. Zinkevich. Subgradient methods for maximum margin structured learning. In *ICML Workshop on Learning in Structured Output Spaces*, 2006.

[43] N.D. Ratliff, J.A. Bagnell, and M.A. Zinkevich. (online) subgradient methods for structured prediction. In *AISTATS*, volume 2007. Citeseer, 2006.

[44] P. Ravikumar, A. Agarwal, and M.J. Wainwright. Message-passing for graph-structured linear programs: proximal projections, convergence and rounding schemes. In *Proceedings of the 25th international conference on Machine learning*, pages 800–807. ACM New York, NY, USA, 2008.

[45] C. Shen and T. Li. Multi-document summarization via the minimum dominating set. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 984–992, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[46] R. Sipos, P. Shivaswamy, and T. Joachims. Large-margin learning of submodular summarization methods. In *Proceedings of the 13th conference of the European chapter of the Association for Computational Linguistics*, 2012.

[47] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *NIPS*, 2010.

[48] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[49] H. Takamura and M. Okumura. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781–789. Association for Computational Linguistics, 2009.

[50] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. *Advances in neural information processing systems*, 16:25–32, 2004.

[51] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM, 2005.

[52] B. Taskar, S. Lacoste-Julien, and M.I. Jordan. Structured prediction, dual extragradient and bregman projections. *The Journal of Machine Learning Research*, 7:1627–1653, 2006.

[53] K. Toutanova, C. Brockett, M. Gamon, J. Jagarlamudi, H. Suzuki, and L. Vanderwende. The PYTHY summarization system: Microsoft research at DUC 2007. In *the proceedings of Document Understanding Conference*, 2007.

[54] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2006.

[55] D. Wang, S. Zhu, T. Li, and Y. Gong. Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 297–300, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[56] D.J.A. Welsh. *Matroid theory*, volume 5. Academic Press London, New York, San Francisco, 1976.

[57] Yisong Yue and Carlos Guestrin. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems 24*, pages 2483–2491, 2011.

Crowdsourcing Control: Moving Beyond Multiple Choice

Christopher H. Lin **Mausam** **Daniel S. Weld**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{chrislin,mausam,weld}@cs.washington.edu

Abstract

To ensure quality results from crowdsourced tasks, requesters often aggregate worker responses and use one of a plethora of strategies to infer the correct answer from the set of noisy responses. However, all current models assume prior knowledge of all possible outcomes of the task. While not an unreasonable assumption for tasks that can be posited as multiple-choice questions (*e.g.* n-ary classification), we observe that many tasks do not naturally fit this paradigm, but instead demand a free-response formulation where the outcome space is of infinite size (*e.g.* audio transcription). We model such tasks with a novel probabilistic graphical model, and design and implement LazySusan, a decision-theoretic controller that dynamically requests responses as necessary in order to infer answers to these tasks. We also design an EM algorithm to jointly learn the parameters of our model while inferring the correct answers to multiple tasks at a time. Live experiments on Amazon Mechanical Turk demonstrate the superiority of LazySusan at solving SAT Math questions, eliminating 83.2% of the error and achieving greater net utility compared to the state-of-the-art strategy, majority-voting. We also show in live experiments that our EM algorithm outperforms majority-voting on a visualization task that we design.

1 Introduction

Crowdsourcing marketplaces (*e.g.*, Amazon Mechanical Turk) continue to rise in popularity. Hundreds of thousands of workers produce a steady stream of output for a wide range of jobs, such as product categorization, audio-video transcription and interlingual translation. Unfortunately, these workers also come with hugely varied skill sets and motivation levels. Ensuring high quality results is a serious challenge for all requesters.

Researchers have studied quality control extensively for the case of simple binary choice (or multiple choice) questions. A common practice is to ask multiple workers and aggregate responses by a majority vote [Snow et al., 2008]. Several extensions have been proposed that track the ability of individual workers while estimating the inherent difficulty of questions [Dai et al., 2010, Whitehill et al., 2009]. These methods typically outperform majority vote and achieve a much higher accuracy.

A key drawback of prior decision-theoretic approaches to quality control is the restriction to multiple choice questions, *i.e.*, jobs where every alternative answer is known in advance and the worker has to simply select one. While many tasks can be formulated in a multiple-choice fashion (*e.g.* n-ary classification), there a large number of tasks with an unbounded number of possible answers. A common example is completing a database with workers' help, *e.g.*, asking questions such as "Find the mobile phone number of Acme Corporation's CEO." Since the space of possible number of answers is huge (possibly infinite), the task interface cannot explicitly enumerate them for the worker. We call these tasks *open questions*.

Unfortunately, adapting multiple-choice models for open questions is not straightforward, because of the difficulty with reasoning about unknown answers. Requesters, therefore, must resort to using a majority-vote, a significant hindrance to achieving quality results from these more general open questions.

Our paper tackles this challenging problem of modeling tasks where workers are free to give any answer. As a first step, we restrict these tasks to those which have exactly one correct answer. We make the following contributions:

- We propose a novel, probabilistic model relating the accuracy of workers and task difficulty to worker responses, which are generated from a countably infinite set.
- We design a decision-theoretic controller, LazySusan, to dynamically infer the correct answer to a task by only soliciting more worker responses when necessary. We also design an Expectation-Maximization (EM) algorithm to jointly learn the parameters of our model while inferring the correct answers to multiple tasks at a time.
- We evaluate variations of our approach first in a simulated environment and then with live experiments on Amazon Mechanical Turk. We show that LazySusan outperforms majority-voting, achieving 83.2% error reduction and greater net utility for the task of solving SAT Math questions. We also show in live experiments that our EM algorithm outperforms majority-voting on a visualization task that we design.

2 Background

There exist many models that tackle the problem of inferring a correct answer for a task with a finite number of possible answers. Our work is based on the model of of Dai *et al.* [Dai et al., 2010], which we now review. Their model assumes that there are exactly 2 possible answers (binary classification). Let $d \in [0, 1]$ denote the inherent *difficulty* of completing a given task, and let $\gamma_w \in [0, \infty)$ be worker w's innate proneness to *error*. The accuracy of a worker on a task, $a(d, \gamma_w)$, is defined to be the probability that she produces the correct answer using the following model: $a(d, \gamma_w) = \frac{1}{2}\left(1 + (1 - d)^{\gamma_w}\right)$. As d and γ increase, the probability that the worker produces the correct answer approaches 0.5, suggesting that she is guessing randomly. On the other hand, as d and γ decrease, a approaches 1, when the worker will deterministically produce the correct answer.

Dai *et al.* couple this model with a utility function, which describes the worth of a correct answer, to define a Partially Observable Markov Decision Process (POMDP) that outputs a policy for TurKontrol, their decision-theoretic controller for crowdsourced tasks. A significant limitation of this model is its inability to handle tasks with an infinite number of possible answers.

In order to address this limitation, we use the Chinese Restaurant Process [Aldous, 1985], a discrete-time stochastic process that generates an infinite number of labels ("tables"). Intuitively, the process may be thought of as modeling sociable customers who, upon entering the restaurant, decide between joining other diners at a table or starting a new table. The greater the number of customers sitting at a table, the more likely new customers will join that table.

Formally, a Chinese Restaurant $R = (T, f, \theta)$ is a set of occupied tables $T = \{t_1, \ldots, t_n | t_i \in \mathbb{N}\}$, a function $f : T \to \mathbb{N}$ that denotes the number of customers at each table t_i, and a parameter $\theta \in \mathbb{R}^+$. Imagine that a customer arrives at the restaurant. He can either choose to sit at one of the occupied tables, or at a new empty table. The probability that he chooses to sit at an occupied table $t \in T$ is

$$C_R(t) \quad = \quad \frac{f(t)}{N + \theta}$$

where $N = \sum_{t \in T} f(t)$ is the total number of customers in the restaurant. The probability that he chooses to begin a new table, or, equivalently, the probability that he chooses not to sit at an occupied table is

$$NT_R \quad = \quad \frac{\theta}{N + \theta}$$

Note that this probability is not equivalent to the probability that the new customer chooses to sit at a *specific* unoccupied table $t \in \mathbb{N} \setminus T$. Since there are an infinite number of unoccupied tables, the total sum of these probabilities, should they be defined this way, would be unbounded.

θ is a parameter that defines how attractive unoccupied tables are at this restaurant. As θ grows, a new customer becomes more likely to sit by himself at a new empty table.

3 Probabilistic Model

We seek to develop a probabilistic model of workers on tasks that have a countably infinite solution space. As a first step, we focus on tasks that have exactly one correct answer (*e.g.* the examples mentioned in the introduction). We also assume that given the correct answer and problem difficulty, the worker's capability to produce the correct answer is independent of all the previous workers' responses. However, even when conditioning on v and d, wrong answers *are* dependent on previous workers' responses (because common mistakes are often repeated). Finally, we assume that workers are not adversarial and do not collaborate with each other.

Dai *et al.*'s model is unable to solve this problem. When one tries to extend their model to tasks with an infinite number of possible solutions, several issues arise from the difficulty of assigning an infinite number of probabilities. For instance, since their model assigns a probability to each possible solution, the naive extension of attempting to place a uniform distribution over the space of solutions is impossible.

Additionally, a good model must consider correlated errors [Grier, 2011]. For instance, consider a task that asks a worker to find the mobile phone number of a company's CEO. We can reasonably guess that the worker might Google the company name, and if one examined a histogram of worker responses, it would likely be correlated with the search results. A common error might be to return the company's main number rather than the CEO's mobile. Not all possible answers are equally likely, and a good model must address this fact.

The Chinese Restaurant Process meets our desiderata. Let tables correspond to possible incorrect solutions to the task (Chinese restaurant); a new worker (diner) is more likely to return a common solution (sit at a table with more people) than a less common solution. We now formally define our extension of Dai *et al.*'s model to the case of unbounded possible answers.

We redefine the accuracy of a worker for a given task, $a(d, \gamma_w)$, to be:

$$a(d, \gamma_w) \quad = \quad (1-d)^{\gamma_w}$$

As a worker's error parameter and/or the task's difficulty increases, the probability the worker produces the correct answer approaches 0. On the other hand, as the stated parameters decrease, a approaches 1, meaning the worker always produces the correct answer.

In addition to the difficulty d and the worker error γ_w, let $\theta \in \mathbb{R}^+$ denote the task's *bandwagon coefficient*. The parameter θ encodes the concept of the "tendency towards a common wrong answer." If θ is high, then workers who answer incorrectly will tend to provide new, unseen, incorrect answers, suggesting that the task does not have "common" wrong answers. Contrastingly, if θ is low, workers who answer incorrectly will tend toward the same incorrect answer, suggesting that the task lends itself to the same mistakes.

Figure 1 illustrates our generative model, which encodes a Bayes Net for responses made by W workers on a given task. x_i is a binary random variable that indicates whether or not the i^{th} worker answers correctly. It is influenced by the correct answer v, the difficulty parameter d, and the error parameter γ_i. b_i, the answer that is provided by the i^{th} worker, is determined by x_i and all previous responses $b_1, \ldots, b_{i-1}$. Only the responses are observable variables.

Let $\mathbf{B_i} = \{b_1, \ldots, b_i\}$ be the multiset of answers that workers $w_1, \ldots, w_i$ provide. Let $\mathbf{A_i} = \{a_1, \ldots, a_k\}$ be the set of unique answers in $\mathbf{B_i}$. The probability that the $i{+}1^{th}$ worker produces the correct answer is simply the worker's accuracy for the given task:

$$\begin{aligned} P(x_i = T|d, v) &= a(d, \gamma_{i+1}) \\ P(x_i = F|d, v) &= 1 - a(d, \gamma_{i+1}) \end{aligned}$$

Then, the probability that the worker's ballot is correct is defined as

$$P(b_{i+1} = v|d, v, \mathbf{B_i}) \quad = \quad P(x_i = T|d, v)$$

To define the probability space of wrong answers we use the Chinese Restaurant Process. Let $f(a) = |\{b \in \mathbf{B_i}|b = a\}|$, and let $R_{i,v} = (\mathbf{A_i} \setminus \{v\}, f, \theta)$ be a Chinese Restaurant Process. Then, the probability that the worker returns a previously seen incorrect answer, $y \in \mathbf{A_i} \setminus \{v\}$ is

$$P(b_{i+1} = y|d, v, \mathbf{B_i}) \quad = \quad P(x_i = F|d, v) C_{R_{i,v}}(y)$$

Finally, the probability that the worker returns an unseen answer is

$$P(b_{i+1} = u|d, v, \mathbf{B_i}) \quad = \quad P(x_i = F|d, v) N T_{R_{i,v}}$$

Here, u represents whatever the worker returns as long as $u \notin \mathbf{A_i}$. More formally, $u \in U$ where U is the singleton set $\{x|b_{i+1} = x \wedge b_{i+1} \notin \mathbf{A_i}\}$. We abuse notation to simplify and elucidate:

$$P(b_{i+1} \notin \mathbf{A_i}|d, v, \mathbf{B_i}) := P(b_{i+1} = u|d, v, \mathbf{B_i})$$

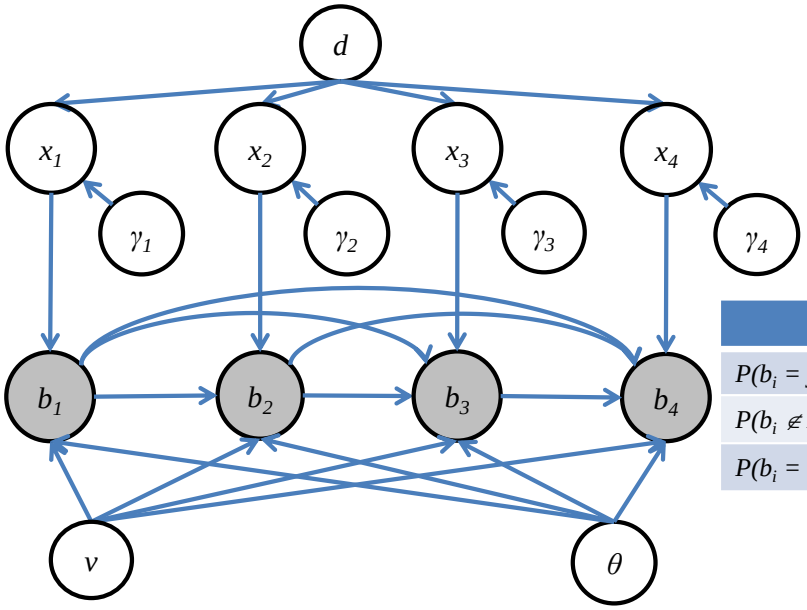
The model cares only about whether it has seen a worker's answer before, not what it actually turns out to be.

3.1 Model Discussion

We now make several subtle and important observations.

First, our model is dynamic in the following sense. As more workers provide answers, the probabilities that govern the generation of an incorrect answer change. In particular, the parameter θ becomes less and less significant as more and more workers provide answers. In other words, as i goes to infinity, the probability that a new worker provides an unseen answer, $\theta/(\theta{+}i)$, goes to 0. As workers provide answers, the probability mass that used to dictate the generation of a new unseen answer is slowly shifted to that which determines the generation of seen answers. See Section 5.4 for a consequence of this behavior. Although we do not believe these model dynamics completely reflect the real-world accurately, we believe our model is a good first approximation with several desirable aspects. In fact, the model dynamics we just described are able to capture the intuition that as more and more answers arrive, we should expect to see fewer and fewer new answers.

Second, certain areas of parameter space cause our model to produce adversarial behavior. In other words, there are settings of d and θ for a task such that the probability a worker produces a particular incorrect answer is greater than the probability a worker



x_i	$P(x_i)$
T	$a(d, \gamma_i)$
F	$1\text{-}a(d, \gamma_i)$

x_i	T	F
$P(b_i = y \in \mathbf{A_i} \mid x_i)$	0	$C_{R_{i,v}}(y)$
$P(b_i \notin \mathbf{A_i} \mid x_i)$	0	$NT_{R_{i,v}}$
$P(b_i = v \mid x_i)$	1	0

Figure 1: Whether or not worker i gets it right, x_i, depends on his error parameter γ_i and the difficulty of the task d. Worker i's answer b_i depends on x_i, the question's true answer v, and all the previous workers' answers $b_1, \ldots, b_{i-1}$. R_i is a Chinese Restaurant Process defined by $\mathbf{B_i}$. This figure shows 4 workers. The b_i are the only observed variables.

provides the correct answer, on average. The following theorem, proven in the supplementary materials, makes this observation.

Theorem. *Suppose the difficulty, d, is fixed and all workers' γ are equal. Then, $\theta < \frac{1-2(1-d)^\gamma}{(1-d)^\gamma}$ if and only if the expected probability the next worker returns the first-seen incorrect answer is greater than the probability the next worker returns the correct answer.*

We note that the theorem only considers the first-seen incorrect answer since the behavior of the Chinese Restaurant Process is such that the first-seen incorrect answer is generated with the highest expected probability. Thus, if the expected probability of generating the correct answer exceeds that of the first-seen incorrect answer, the model will not produce adversarial behavior.

Finally, while the purpose of this paper is to address open questions with infinite answer spaces, we note that Polya's Urn Scheme, the finite version of the Chinese Restaurant Process, applies equally well to finite answer spaces with many answer choices (multiple-choice tasks).

4 A Decision-Theoretic Agent

We now discuss the construction of our decision-theoretic controller, LazySusan. Our control problem is as follows. Given an open question as input, the goal is to infer the correct answer. At each time-step, an agent can choose one of two actions. It can either stop and submit the most likely answer, or it can create another job and receive another response to the task from another crowdsourced worker. The question is: How do we determine the agent's policy?

To solve this problem, first we define the *world state* of LazySusan to be the pair (v, d), where $v \in \mathbb{N}$ is the correct answer of the task and d is the difficulty of the task. The space of world states is infinitely large, and LazySusan cannot directly observe the world state, so it has an *agent state* $\mathbf{S}$, which at time i, is the set of tuples, $\mathbf{S} = \{(v, d) | v \in \mathbf{A_i} \cup \{\perp\} \wedge d \in [0, 1]\}$. Here, $\perp$ represents the case when the true answer has not been seen by the agent so far. In order to keep track of what it believes to be the correct answer v, it maintains a *belief*, which is a probability distribution over $\mathbf{S}$.

The agent state allows us to fold an infinite number of probabilities into one, since to compute the belief, one only needs to calculate $P(v = \perp, d|\mathbf{B_i})$, the probability that no workers have provided the correct answer yet given $\mathbf{B_i}$, instead of $P(v = u, d|\mathbf{B_i})$ for all possible unseen answers $u \in \mathbb{N} \setminus \mathbf{A_i}$.

4.1 Belief Update

We now describe specifically how LazySusan updates its posterior belief $P(v, d|\mathbf{B_i}; i, k)$ after it receives its i^{th} ballot b_i. Here, $k = |\mathbf{A_i}|$ is the number of unique responses it has received. By Bayes' Rule, we have

$$P(v, d|\mathbf{B_i}; i, k) \propto P(\mathbf{B_i}|v, d; i, k)P(v, d; i, k)$$

Symbol	Meaning
$\mathbf{A_i}$	The set of unique responses made by workers $1, \ldots, i$.
$\mathbf{B_i}$	The multiset of responses made by workers $1, \ldots, i$ ($\{b_1, \ldots, b_i\}$)
b_i	Worker i's response to the task
C_C	The value of a correct answer
$C_{R_{i,v}}(y)$	The probability of a worker producing incorrect answer y in restaurant $R_{i,v}$.
C_W	The value of an incorrect answer
d	Difficulty of task
γ_i	Worker i's error parameter
θ	A task's bandwagon coefficient; Chinese Restaurant Process parameter
k	The number of unique worker responses ($\lvert\mathbf{A_i}\rvert$)
i	Number of ballots received so far
$Q(\mathbf{B_i}, \texttt{action})$	The utility of taking `action` with belief $\mathbf{B_i}$
$NT_{R_{i,v}}$	The probability of a worker producing an unseen answer in restaurant $R_{i,v}$.
$R_{i,v}$	An instance of the Chinese Restaurant Process instantiated using $\mathbf{B_i}$ and correct answer v.
$U(\mathbf{B_i})$	The utility of a belief derived from $\mathbf{B_i}$.
$V(a)$	The value of an answer a
v	Correct answer of task
x_i	Did worker i answer the task correctly

Table 1: Summary of notation used in this paper

The likelihood of the worker responses $P(\mathbf{B_i}|v, d; i, k)$ is easily calculated using our generative model:

$$P(\mathbf{B_i}|v, d; i, k) = \prod_{j=1}^{i} P(b_j|v, d, \mathbf{B_{j-1}})$$

The prior on the correct answer and difficulty can be further reduced:

$$P(v, d; i, k) = P(v|d; i, k)P(d; i, k)$$

The prior we must compute describes the joint probability of the correct answer and difficulty given i responses and k distinct responses. Notice that for all $a \in \mathbf{A_i}$, we do not know $P(v = a|d; i, k)$. However, they must be all the same, because knowing the difficulty of the task gives us no information about the correct answer. Therefore, we must only determine the probability the correct answer has yet to be seen given d, i, and k. We propose the following model:

$$P(v = \perp |d; i, k) = d^i$$

This definition is reasonable since intuitively, as the difficulty of the task increases the more likely workers have not yet provided a correct answer. On the other hand, as the number of observations increases, we become more certain that the correct answer is in $\mathbf{A_i}$.

Finally, we model $P(d; i, k)$. Consider for the moment that workers tend to produce answers that LazySusan has seen before (θ is low). Intuitively, as k approaches i, the difficulty should grow, because the agent is seeing a lot of different answers when it shouldn't, and as k approaches 1, the difficulty should become smaller, because everyone is agreeing on the same answer.

We choose to model $P(d; i, k) \sim Beta(\alpha, \beta)$ and define $\alpha \geq 1$ and $\beta \geq 1$ as

$$\begin{aligned} \alpha &= \left((i-1)\frac{k}{i} + 1\right)^{\frac{1}{\theta}} \\ \beta &= \left((1-i)\frac{k}{i} + i\right)^{\theta} \end{aligned}$$

First note that the bulk of a Beta distribution's density moves toward 1 as α increases relative to β, and toward 0 as β increases relative to α. Thus, as β increases, difficulty decreases, and as α increases, difficulty increases. Consider the case when $\theta = 1$. As k approaches i, α approaches i and β approaches 1, causing LazySusan to believe the difficulty is likely to be high, and as k approaches 1, LazySusan believes the difficulty is likely to be low. This behavior is exactly what we desire.

Now we consider the effect of θ. Fix i and k. As θ grows, β increases and α decreases. Therefore, for a fixed multiset of observations, as people become more likely to provide unseen answers, the probability that the difficulty is low becomes greater. In other words, LazySusan needs to see a greater variety of answers to believe that the problem is difficult if θ is high.

Let $\theta_1 > \theta_2$ and consider the following scenarios: Suppose k is close to i, so LazySusan believes, before factoring in θ, that the task is difficult. If $\theta = \theta_2$, LazySusan believes with more certainty that the task is difficult than if $\theta = \theta_1$. This behavior makes sense be-

cause LazySusan should expect a small k if θ is small. If $\theta = \theta_2$, LazySusan should see a smaller k than if $\theta = \theta_1$. If it sees a k that is larger than it expects, it correctly deduces that more people are getting the question wrong, and concludes the task is more difficult. Similarly, if k is close to 1, and $\theta = \theta_1$, then LazySusan believes with more certainty that the task is easy than if $\theta = \theta_2$, since even though workers tend to produce more random answers, k is small.

4.2 Utility Estimation

To determine what actions to take, LazySusan needs to estimate the utility of each action. The first step is to assign utilities to its beliefs. Since $\mathbf{B_i}$ solely determines its belief at time i, we denote $U(\mathbf{B_i})$ to be utility of its current belief. Next, LazySusan computes the utilities of its two possible actions. Let $Q(\mathbf{B_i}, \texttt{submit})$ denote the utility of submitting the most likely answer given its current state, and let $Q(\mathbf{B_i}, \texttt{request})$ denote the utility of requesting another worker to complete the task and then performing optimally. Then

$$\begin{aligned} U(\mathbf{B_i}) &= \max\{Q(\mathbf{B_i}, \texttt{submit}), \\ &\qquad Q(\mathbf{B_i}, \texttt{request})\} \\ Q(\mathbf{B_i}, \texttt{submit}) &= \sum_{a \in \mathbf{A_i}} V(a) \int_d P(v = a, d | \mathbf{B_i}; i, k) \mathrm{d}d \\ Q(\mathbf{B_i}, \texttt{request}) &= c + \sum_{a \in \mathbf{A_i}} P(b_{i+1} = a | \mathbf{B_i}) U(\mathbf{B_{i+1}}) \\ &\quad + P(b_{i+1} \notin \mathbf{A_i} | \mathbf{B_i}) U(\mathbf{B_{i+1}}) \end{aligned}$$

where c is the cost of creating another job and $P(b_{i+1}|\mathbf{B_i}) =$

$$\sum_{a \in \mathbf{A_i}} \int_d P(b_{i+1} | v = a, d, \mathbf{B_i}) P(v = a, d | \mathbf{B_i}) \mathrm{d}d$$

LazySusan takes as inputs C_C, the utility of a correct answer, and C_W, the utility of an incorrect answer. These values are provided by the requester to manage tradeoffs between accuracy and cost. LazySusan uses its own estimate of the correct answer to calculate $V(a)$:

$$\begin{aligned} a* &= \operatorname*{argmax}_{a \in \mathbf{A_i}} \int_d P(v = a, d | \mathbf{B_i}; i, k) \mathrm{d}d \\ V(a) &= \begin{cases} C_C & \text{if } a = a* \\ C_W & \text{otherwise} \end{cases} \end{aligned}$$

4.3 Worker Tracking

After submitting an answer, LazySusan updates its records about all the workers who participated in the task using $a*$. We follow the approach of Dai *et al.* [Dai et al., 2010], and use the following update rules: 1) $\gamma_w \leftarrow \gamma_w - d\epsilon$ should the worker answer correctly, and 2) $\gamma_w \leftarrow \gamma_w + (1-d)\epsilon$, should the worker answer incorrectly, where ϵ is a decreasing learning rate. Any worker that LazySusan has not seen previously begins with some starting $\overline{\gamma}$.

4.4 Decision Making

We note that the agent's state space continues to grow without bound as new answers arrive from crowdsourced workers. This poses a challenge since existing POMDP algorithms do not handle infinite-horizon problems in dynamic state spaces where there is no a-priori bound on the number of states. Indeed, the efficient solution of such problems is an exciting problem for future research. As a first step, LazySusan selects its actions at each time step by computing an l-step lookahead by estimating the utility of each possible sequence of l actions. If the l^{th} action is to request another response, then it will cut off the computation by assuming that it submits an answer on the $l+1^{th}$ action.

In many crowdsourcing platforms, such as Mechanical Turk, we cannot preselect the workers to answer a job. However, in order to conduct a lookahead search, we need to specify future workers' parameters for our generative model. To simplify the computation, we assume that every future worker has $\gamma = \overline{\gamma}$.

4.5 Joint Learning and Inference

We now describe an EM algorithm that can be used as an alternative to the working-tracking scheme from above. Given a set of worker responses from a set of tasks, $\mathbf{b}$, EM jointly learns maximum-likelihood estimates of $\boldsymbol{\gamma}, \mathbf{d}$, and $\boldsymbol{\theta}$, while inferring the correct answers $\mathbf{v}$. Thus, in this approach, after LazySusan submits an answer to a task, it can recompute all model parameters before continuing with the next task.

We treat the variables $\mathbf{d}, \boldsymbol{\gamma}$, and $\boldsymbol{\theta}$ as parameters. In the E-step, we keep parameters fixed to compute the posterior probabilities of the hidden true answers: $p(v_t | \mathbf{b}, \mathbf{d}, \boldsymbol{\gamma}, \boldsymbol{\theta})$ for each task t. The M-step uses these probabilities to maximize the standard expected complete log-likelihood L over $\mathbf{d}, \boldsymbol{\gamma}$, and $\boldsymbol{\theta}$:

$$L(\mathbf{d}, \boldsymbol{\gamma}, \boldsymbol{\theta}) = E[\ln p(\mathbf{v}, \mathbf{b} | \mathbf{d}, \boldsymbol{\gamma}, \boldsymbol{\theta})]$$

where the expectation is taken over $\mathbf{v}$ given the old values of $\boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{\theta}$.

5 Experiments

This section addresses the following three questions: 1) How deeply should the lookahead search traverse? 2)

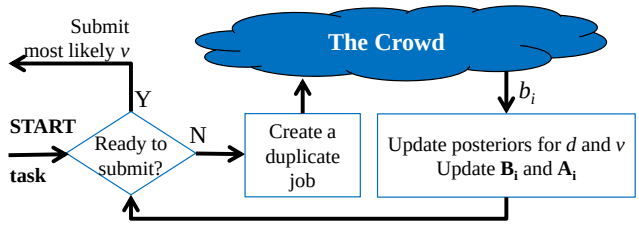


Figure 2: LazySusan's decisions when executing a task.

How robust is LazySusan on different classes of problems? 3) How well does LazySusan work in practice? and 4) How well does our EM algorithm work in practice?

To answer the first question, we compare LazySusan at different settings of lookahead depth. Then, to answer the second question, we test the robustness of LazySusan by applying it to various kinds of problems in simulation. Next, to answer the third question, we compare LazySusan to an agent that uses majority-voting with tasks that test the workers of Amazon Mechanical Turk on their SAT Math skills. Finally, to answer the fourth question, we compare our EM algorithm to a majority-voting strategy on a visualization task.

5.1 Implementation

Since numerical integration can be challenging, we discretize difficulty into nine equally-sized buckets with centers of $0.05, 0.15, \ldots, 0.85$, and 0.95.

For the purposes of simulation only, each response that LazySusan receives also contains perfect information about the respective worker's γ. Thus, LazySusan can update its belief state with no noise, which is the best-case scenario.

In all cases, we set the learning rate $\epsilon = \frac{1}{m_w+1}$, where m_w is the number of questions a worker w has answered so far. We also set the value of a correct answer to be $C_C = 0$. Finally, we set $\overline{\gamma} = 1$, and the bandwagon coefficient $\theta = 1$ for all tasks.

5.2 Best Lookahead Depth

We first determine the best lookahead depth among 2, 3, and 4 with simulated experiments. We evaluate our agents using several different settings of the value of an incorrect answer: $C_W \in \{-10, -50, -100\}$. Each difficulty setting is used 10 times, for a total 90 simulations per utility setting. (9 difficulty settings $\times$ 10 $=$ 90). We set the cost of requesting a job, c, from a (simulated) worker to -1. Our simulated environment draws worker $\gamma \in (0, 2)$ uniformly.

Figure 3 shows the results of our simulation. LazySusan(l) denotes a lookahead depth of l. We also examine an agent that uses a majority-of-7-vote strategy, MV. Expectedly, as the utility of an incorrect answer decreases, the average net utility achieved by each agent drops. We find that for all settings of lookahead depth, LazySusan dramatically outperforms MV. We also see that LazySusan(3) and LazySusan(4) both achieve small, but sure gains over LazySusan(2). However, LazySusan(3) and LazySusan(4) seem to do about the same. Since LazySusan(4) runs significantly slower, we decide to use LazySusan(3) in all future experiments, and refer to it as LazySusan.

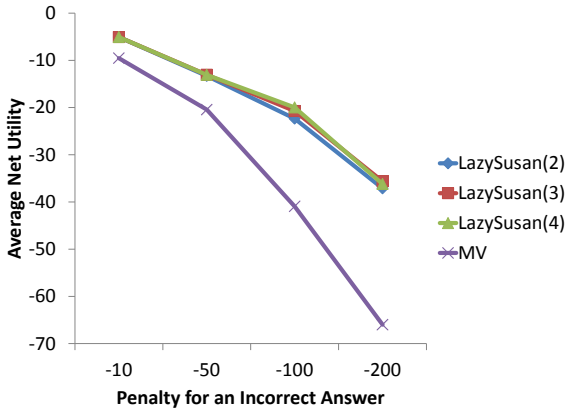


Figure 3: In simulation, our lookahead search does well at all depths.

5.3 Noisy Workers

We examine the effect of poor workers. We compare two simulation environments. In the first, we draw workers' $\gamma \in (0, 1)$ uniformly and in the second, we draw workers' $\gamma \in (0, 2)$ uniformly. Recall that γ is the worker error parameter. Thus, the first environment has workers that are much more competent than those in the second. All other parameters remain as before. Each difficulty setting is simulated 100 times, for a total of 900 simulations per environment. The results of this experiment are shown in Table 2. When the workers are competent, LazySusan makes small gains over MV, reducing the error by 34.3%. However, when there exist noisier workers in the pool, LazySusan more decisively outperforms MV, reducing the error by 48.6%. In both cases, LazySusan spends less money than MV, netting average net utility gains of 55% and 75.9%.

5.4 Tasks with Correlated Errors

Next, we investigate the ability of LazySusan to deal with varying θ and d in the worst case — when all the workers are equally skilled. Recall that a high θ

	$\gamma \in (0,1)$		$\gamma \in (0,2)$	
	LazySusan	MV	LazySusan	MV
Avg Accuracy (%)	88.7	82.8	83.3	67.5
Avg Cost	3.472	5.7	4.946	6.01
Avg Net Utility	-14.772	-22.9	-21.646	-38.51

Table 2: Comparison of average accuracies, costs, and net utilities of LazySusan and MV when workers either have $\gamma \in (0,1)$ or $\gamma \in (0,2)$

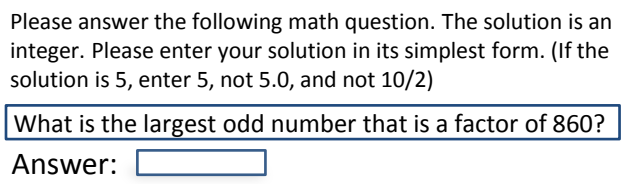


Figure 4: An example of an SAT Math Question task posed to workers for live experiments on Mechanical Turk.

means that workers who answer incorrectly will tend to produce previously unseen answers. We consider the following variations of tasks: 1) Low difficulty, 2) High difficulty, high θ, and 3) High difficulty, low θ. In the first two cases, we see expected behavior. LazySusan is able to use its model to infer correct answers.

However, in the third case, we see some very interesting behavior. Since the difficulty is high, workers more often than not produce the wrong answer. Additionally, since θ is low, they also tend to produce the same wrong answer, making it look like the correct answer. If the ratio is large enough, we find that LazySusan is unable to infer the correct answer, because of the unfortunate ambiguity between high difficulty, low θ problems and low difficulty problems. In fact, as LazySusan observes more ballots, it becomes more convinced that the common wrong answer is the right answer, because of the model dynamics we mention earlier (Section 3.1). This problem only arises, however, if the model produces adversarial 0, and we see in practice that workers on Mechanical Turk generally do not exhibit such behavior.

5.5 Experiments on Mechanical Turk

Next, we compare LazySusan to an agent using majority-vote (MV) using real responses generated by Mechanical Turk workers. We test these agents with 134 math questions with levels of difficulty comparable to those found on the SAT Math section. Figure 4 is an example of one such task and the user interface we provided to workers. We set the utility for an incorrect answer, C_W, to be -100, because with this utility setting, LazySusan requests about 7 jobs on average for each task, and a simple binary search showed

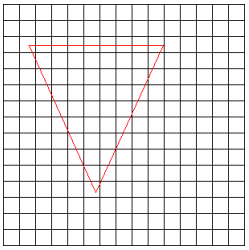


Figure 5: An example of a "triangle" task posed to workers for live experiments on Mechanical Turk. The area of this triangle, rounded down, is 38.

	LazySusan	MV
Avg Accuracy (%)	99.25	95.52
Avg Cost	5.17	5.46
Avg Net Utility	-5.92	-9.94

Table 3: Comparisons of average accuracies, costs, and net utilities of LazySusan and MV when run on Mechanical Turk.

this number to be satisfactorily optimal for MV. We find that the workers on Mechanical Turk are surprisingly capable at solving math problems. As Table 3 shows, LazySusan almost completely eliminates the error made by MV. Since the two agents cost about the same, LazySusan achieves a higher net utility, which we find to be statistically significant using a Student's t-test ($p < 0.0002$).

We examine the sequence of actions LazySusan made to infer the correct answer to the task in Figure 4. In total, it requested 14 ballots, and received the following responses: 215, 43, 43, 43, 5, 215, 43, 3, 55, 43, 215, 215, 215, 215. Since MV takes the majority of 7 votes, it infers the answer incorrectly to be 43. LazySusan on the other hand, uses its knowledge of correlated answers as well as its knowledge from previous tasks that the first three workers who responded with 43 were all relatively poor workers compared to the first two workers who claimed the answer is 215. So even though a clear majority of workers preferred 43, LazySusan was not confident about the answer. While it cost twice as much as MV, the cost was a worthy sacrifice with respect to the utility setting.

Finally, we compare our EM algorithm to MV, using

real responses generated by Mechanical Turk workers. We develop a "triangle" task (Figure 5) that presents workers with a triangle drawn on a grid, and asks them to find the area of the triangle, rounded down. We posted 200 of these tasks and solicited 5 responses for each. These tasks are difficult since many of the responses are off by 1. Our EM algorithm achieves an accuracy of 65.5% while MV achieves an accuracy of 54.1%.

6 Related Work

Modeling repeated labeling in the face of noisy workers when the label is assumed to be drawn from a known *finite* set has received significant attention. Romney *et al.* [Romney et al., 1986] are one of the first to incorporate a worker accuracy model to improve label quality. Sheng *et al.* [Sheng et al., 2008] explore when it is necessary to get another label for the purpose of machine learning. Raykar *et al.* [Raykar et al., 2010] propose a model in which the parameters for worker accuracy depend on the true answer. Whitehill *et al.* [Whitehill et al., 2009] and Dai *et al.* [Dai et al., 2010] address the concern that worker labels should not be modeled as independent of each other unless given problem difficulty. Welinder *et al.* [Welinder et al., 2010] design a multidimensional model for workers that takes into account competence, expertise, and annotator bias. Kamar *et al.* [Kamar et al., 2012] extracts features from the task at hand and use Bayesian Structure Learning to learn the worker response model. Parameswaran *et al.* [Parameswaran et al., 2010] conduct a policy search to find an optimal dynamic control policy with respect to constraints like cost or accuracy. Karger *et al.* [Karger et al., 2011] develop an algorithm based on low-rank matrix approximation to assign tasks to workers and infer correct answers, and analytically prove the optimality of their algorithm at minimizing a budget given a reliability constraint. Snow *et al.* [Snow et al., 2008] show that for labeling tasks, a small number of Mechanical Turk workers can achieve an accuracy comparable to that of an expert labeler. None of these works consider tasks that have an infinite number of possible solutions.

For more complex tasks that have an infinite number of possible answers, innovative workflows have been designed, for example, an iterative improvement workflow for creating complex artifacts [Little et al., 2009], find-fix-verify for an intelligent editor [Bernstein et al., 2010], and others for counting calories on a food plate [Noronha et al., 2011].

An AI agent makes an efficient controller for these crowdsourced workflows. Dai *et al.* [Dai et al., 2010, Dai et al., 2011] create a POMDP-based agent to control an iterative improvement workflow. Shahaf and Horvitz [Shahaf and Horvitz, 2010] develop a planning-based task allocator to assign subtasks to specific humans or computers with known abilities. We [Lin et al., 2012] create a POMDP-based agent to dynamically switch between workflows.

Weld *et al.* [Weld et al., 2011] discuss a broad vision for the use of AI techniques in crowdsourcing that includes workflow optimization, interface optimization, workflow selection and intelligent control for general crowdsourced workflows. Our work provides a more general method for intelligent control.

7 Conclusion & Future Work

This paper introduces LazySusan, an agent that takes a decision-theoretic approach to inferring the correct answer of a task that can have a countably infinite number of possible answers. We extend the probabilistic model of [Dai et al., 2010] using the Chinese Restaurant Process and use l-step lookahead to approximate the optimal number of crowdsourcing jobs to submit. We also design an EM algorithm to jointly learn the parameters of our model while inferring the correct answers to multiple tasks at a time. Live experiments on Mechanical Turk demonstrate the effectiveness of LazySusan. At comparable costs, it yields an 83.2% error reduction compared to majority vote, which is the current state-of-the-art technique for aggregating responses for tasks of this nature. Live experiments also show that that our EM algorithm outperforms majority-voting on "triangle" tasks.

In the future, we would like to address the ambiguity between high difficulty, low θ problems and low difficulty problems. We also hope to develop a generative model that does not change as responses are gathered from workers. We also hope to extend the ability of LazySusan to solving tasks that have multiple correct answers. Indeed, workers oftentimes provide the same answer in different forms (*e.g.*, in different units). Other questions may have several answers (*e.g.*, top executives often carry two mobiles). While multiple correct answers may be reduced with crisply-written instructions, an improved model may also prove useful.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This work was supported by the WRF / TJ Cable Professorship, Office of Naval Research grant N00014-12-1-0211, and National Science Foundation grants IIS 1016713 and IIS 1016465.

References

[Aldous, 1985] Aldous, D. J. (1985). Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII 1983*, volume 1117 of *Lecture Notes in Mathematics*, pages 1–198. Springer Berlin / Heidelberg. 10.1007/BFb0099421.

[Bernstein et al., 2010] Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. (2010). Soylent: A word processor with a crowd inside. In *UIST*.

[Dai et al., 2010] Dai, P., Mausam, and Weld, D. S. (2010). Decision-theoretic control of crowd-sourced workflows. In *AAAI*.

[Dai et al., 2011] Dai, P., Mausam, and Weld, D. S. (2011). Artificial intelligence for artificial intelligence. In *AAAI*.

[Grier, 2011] Grier, D. A. (2011). Error identification and correction in human computation: Lessons from the WPA. In *HCOMP*.

[Kamar et al., 2012] Kamar, E., Hacker, S., and Horvitz, E. (2012). Combining human and machine intelligence in large-scale crowdsourcing. In *AAMAS*.

[Karger et al., 2011] Karger, D. R., Oh, S., , and Shah, D. (2011). Budget-optimal crowdsourcing using low-rank matrix approximations. In *Allerton*.

[Lin et al., 2012] Lin, C. H., Mausam, and Weld, D. S. (2012). Dynamically switching between synergistic workflows for crowdsourcing. In *AAAI*.

[Little et al., 2009] Little, G., Chilton, L. B., Goldman, M., and Miller, R. C. (2009). Turkit: tools for iterative tasks on mechanical turk. In *KDD-HCOMP*, pages 29–30.

[Noronha et al., 2011] Noronha, J., Hysen, E., Zhang, H., and Gajos, K. Z. (2011). Platemate: Crowdsourcing nutrition analysis from food photographs. In *UIST*.

[Parameswaran et al., 2010] Parameswaran, A., Garcia-Molina, H., Park, H., Polyzotis, N., Ramesh, A., and Widom, J. (2010). Crowdscreen: Algorithms for filtering data with humans. In *VLDB*.

[Raykar et al., 2010] Raykar, V. C., Yu, S., Zhao, L. H., and Valadez, G. (2010). Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322.

[Romney et al., 1986] Romney, A. K., Weller, S. C., and Batchelder, W. H. (1986). Culture as consensus: A theory of culture and informant accuracy. *American Anthropologist*, 88(2):313 – 338.

[Shahaf and Horvitz, 2010] Shahaf, D. and Horvitz, E. (2010). Generlized markets for human and machine computation. In *AAAI*.

[Sheng et al., 2008] Sheng, V. S., Provost, F., and Ipeirotis, P. G. (2008). Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[Snow et al., 2008] Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y. (2008). Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP*, pages 254–263.

[Weld et al., 2011] Weld, D. S., Mausam, and Dai, P. (2011). Human intelligence needs artificial intelligence. In *HCOMP*.

[Welinder et al., 2010] Welinder, P., Branson, S., Belongie, S., and Perona, P. (2010). The multidimensional wisdom of crowds. In *NIPS*.

[Whitehill et al., 2009] Whitehill, J., Ruvolo, P., Bergsma, J., Wu, T., and Movellan, J. (2009). Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*.

Response Aware Model-Based Collaborative Filtering

Guang Ling[1,2], Haiqin Yang[1,2], Michael R. Lyu[1,2], and Irwin King[2,3]

[1]Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
[2]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong {gling,hqyang,lyu,king}@cse.cuhk.edu.hk
[3]AT&T Labs Research, San Francisco, CA, USA, irwin@research.att.com

Abstract

Previous work on recommender systems mainly focus on fitting the ratings provided by users. However, the response patterns, i.e., some items are rated while others not, are generally ignored. We argue that failing to observe such response patterns can lead to *biased* parameter estimation and sub-optimal model performance. Although several pieces of work have tried to model users' response patterns, they miss the effectiveness and interpretability of the successful matrix factorization collaborative filtering approaches. To bridge the gap, in this paper, we unify explicit response models and PMF to establish the Response Aware Probabilistic Matrix Factorization (RAPMF) framework. We show that RAPMF subsumes PMF as a special case. Empirically we demonstrate the merits of RAPMF from various aspects.

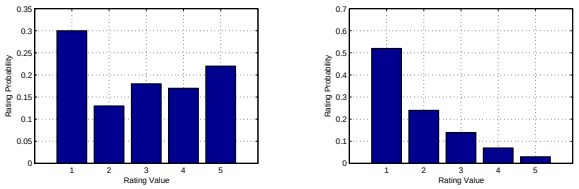
1 Introduction

Recently, online music and video streaming services have seen an explosive growth. As the user base and contents expanding tremendously, recommender systems become crucial for service providers. Cloud-based music streaming services such as iTunes Match, Google Music, Yahoo! Music, Pandora, Songify, etc. make it easier than ever to rate songs and buy new music. With the rocketing growth of the number of users, their explicit ratings become more and more accessible. Effective usage of these ratings can lead to high quality recommendation, which is vital for the cloud-based online streaming services. This is because most services charge very little subscription fee, if not none. The main income comes from the selling of music. Nowadays, online streaming services often have a large user base, so even if a small change in recommendation quality may have dramatic effect on sales.

Due to immense market value, various recommendation techniques have been proposed. Generally, these approaches can be classified into neighborhood-based methods and model-based methods. Typical neighborhood-based approaches includes user-based methods [1, 4] and item-based methods [3, 11, 25]. The state-of-the-art model-based methods include restricted Boltzmann machines [24], SVD++ [8, 9], Probabilistic Matrix Factorization (PMF) [22], and multi-domain collaborative filtering [32], graphical models [7], pair-wise tensor factorization [21], and matrix factorization with social regularization [14], etc.

However, in real-world rating systems, users' ratings carry twofold information. Firstly the rating value indicates a user's preference on a particular item as well as an item's inherent features. The scores that a user assigns to different items convey information on what the user likes and what the user dislikes. The rating values that an item received from different users also carry information on intrinsic properties of the item. Second, the ratings also reveal users' response patterns, i.e., some items are rated while others not. This information can be utilized to improve the model performance. However, previously proposed methods usually assume that all the users would rate *all* the inspected items, or more generally, *randomly* select inspected items to rate. These methods fit the users' ratings directly and ignore the key factor, users' response patterns. The ignorance will degrade the model performance. In this paper, we explore previously ignored response information to further boost recommender system's quality.

Practically, the assumption of *all inspection* or *randomly rate* is not true in real-world rating systems. Users are unlikely to rate all the inspected items or randomly select the inspected items to rate. Shown in Figure 1(a) is the rating value distribution of the items that users *choose to rate*, while Figure 1(b) shows the distribution of ratings for *randomly* selected songs



(a) User selected ratings (b) Randomly selected ratings

Figure 1: Distribution of ratings in a music website [16].

from the same group of users. Clearly these two distributions are very different. In the user selected ratings, there are far more items with high ratings than that in the randomly selected songs. This is compelling evidence showing that the assumption that all the users would rate all the inspected items or select random items to rate is unlikely to be true. The investigation of the Yahoo!LaunchCast data indicates that users are more likely to rate items they do love and hate, but not neutral [16, 26].

To further demonstrate the risk of *incorrect* parameter estimation and *biased* rating prediction when ignoring the response information, we show an intuitive example of five users' rating on five items in Table 1, where the ratings are skewed to either 4 or 5. Clearly, user-based approaches [1, 4] and item-based approaches [3, 11, 25] are more likely to predict rating values in the range of 4 to 5. Similarly, ignoring response information will cause the one-class issue for model-based approaches [10, 17, 18]. In a real-world recommender system, the case may not be as extreme as is in Table 1. Nevertheless, the effect is similar. By ignoring the response information, we will learn a model that has *bias*.

Currently, there are two main streams of work trying to solve the above response ignorance problem. One line of work try to model the above phenomena as a one-class collaborative filtering task [10, 17, 18]. A heuristic weight in the range of 0 to 1 is introduced to calibrate the loss on those unseen ratings, where the rating scores are set to zeros [17, 18]. Embedding user information is also adopted to optimize the weight on the unseen ratings via users' similarity [10]. However, these methods do not model the users' missing response information together with the ratings. The other line of work model the response ignorance through missing data theory [13]. The multinomial mixture model is adopted to model the non-random response [16]. The work is also extended for collaborative ranking [15]. These methods model users' response patterns and ratings via multinomial mixture model, but they discard the effectiveness and interpretability of the matrix factorization approaches [8, 22].

Table 1: Skewed ratings on 5 items from 5 users

	item1	item2	item3	item4	item5
user1	5	4			
user2		5		4	
user3	4			4	
user4	5		5		
user5		4			5

To bridge this gap, we are the first to integrate the users' response patterns into PMF to establish a unified framework, which we refer to as Response Aware PMF (RAPMF). The response models we propose include the rating dominating response model, and a generalized one, the context-aware response model. We demonstrate the advantages of our proposed RAPMF through detailed and fair experimental comparison.

The rest of the paper is organized as follows. In Section 2, we motivate the explicit modeling of user responses from a probabilistic point of view. In Section 3, we present how to incorporate response models into PMF and elaborate the proposed RAPMF model. Empirical study and comparison with previous work is conducted in Section 4. The paper is concluded in Section 5.

2 Response and Missing Theory

Modeling response patterns have a strong incentive from statistical missing data theory [13]. The response patterns can be hidden [2, 6] or explicit. In recommender system case, it is explicit. In the following, we show that without modeling the response patterns properly, we may learn a *bias* model.

2.1 Setup and Notation

Assume that we are given a partially observed $N \times M$ matrix X, where N is the number of users and M is the number of items, the (i, j) element of X denotes the rating assigned by user i to item j in the scale of 1 to D. Collaborative filtering approaches try to recover the original full matrix X_{full} to predict users' preferences.

In the matrix X, an unobserved entry is denoted by 0. Alternatively, we denote all the observations as a set of triplets $(i, j, x) \in \mathcal{Q}$. Moreover, we define a companion response indicator matrix R to denote whether the corresponding rating is observed in X. If $X_{ij} \neq 0$,

i.e., we have observed user i's rating on item j, then $R_{ij} = 1$. Otherwise $R_{ij} = 0$. Note that X is partially observed while R is fully observed.

2.2 Missing Data Theory

Following missing data theory in [13], we model the collaborative filtering data as a two-step procedure. First, a data model $P(X|\theta)$ generates the full data matrix X_{full}. Then, a response model $P(R|X, \mu)$ determines which elements in X_{full} are observed. Hence, we can take a parametric joint distribution on the observed data matrix X and the response matrix R, conditioned on the model parameters, θ and μ.

$$P(R, X|\mu, \theta) = P(R|X, \mu, \theta)P(X|\mu, \theta) \quad (1)$$
$$= P(R|X, \mu)P(X|\theta), \quad (2)$$

where $P(R|X, \mu)$ is also referred to as the missing data model. In the following, we use response model and missing data model interchangeably.

According to the missing data theory [13], there are three kinds of missing data assumptions: 1) Missing Completely At Random (MCAR); 2) Missing At Random (MAR), and 3) Not Missing At Random (NMAR). MCAR has the strongest independence assumption. Under the MCAR assumption, the missing mechanism cannot depend on the data in any way. Whether we will observe a response is fully determined by the parameter μ and is irrelevant to the users' rating, i.e.,

$$P(R|X, \mu) = P(R|\mu) \quad (3)$$

One typical example where MCAR holds is that given an inspected item, whether it will be observed is a Bernoulli trail with probability μ.

The MAR assumption is slightly different from the MCAR assumption. Let $X_{full} = (X_{obs}, X_{mis})$, i.e., the full data matrix X_{full} is separated into observed data matrix X_{obs} and missing data matrix X_{mis}. Under the MAR assumption, the response probability depends on the *observed* data and μ, i.e.,

$$P(R|X, \mu) = P(R|X_{obs}, \mu). \quad (4)$$

Marlin and Zemel [15] refer to this as the probability of observing a particular response only depending on the observed elements of the data vector. The assumption made by MAR may seem bizarre. However, it comes up naturally if we want to ignore response model and still learn *unbiased* data model parameters. We demonstrate this in the following.

Let $\mathcal{L}(\mu, \theta|X_{obs}, R)$ be the likelihood of μ and θ given the observation X_{obs} and R. Under the MAR assumption, we have

$$\begin{aligned}
& \mathcal{L}(\mu, \theta|X_{obs}, R) = P(R, X_{obs}|\mu, \theta) \\
= & \int_{X_{mis}} P(R, X|\mu, \theta)dX_{mis} \\
= & \int_{X_{mis}} P(R|X, \mu)P(X|\theta)dX_{mis} \\
= & \int_{X_{mis}} P(R|X_{obs}, \mu)P(X|\theta)dX_{mis} \quad (5) \\
= & P(R|X_{obs}, \mu) \int_{X_{mis}} P(X|\theta)dX_{mis} \\
= & P(R|X_{obs}, \mu)P(X_{obs}|\theta) \\
\propto & P(X_{obs}|\theta).
\end{aligned}$$

The key to marginalize the missing data is that the missing data model depends only on the observed data, i.e., MAR assumption in Eq. (4). Under the MCAR assumption, we can simplify Eq. (5) similarly, which only depends on μ. Note that the assumption made by MAR appears naturally in the derivation. This is the independence assumption we cannot release anymore without losing the ability to marginalize the complete data model independently of missing data model.

If both MCAR and MAR fail to hold, then NMAR assumption is made. Unlike MCAR and MAR, NMAR requires an explicit response model in order to learn unbiased model parameters. Otherwise, maximizing $P(X_{obs}|\theta)$ directly can yield a *biased* θ. With only a few exceptions [15, 16], nearly all the previous work on recommender systems try to maximize the data model directly [5, 14, 19, 22, 23]. In a typical recommender system, the data collected can easily violate the MAR assumption. The distinct distribution of rating values on user selected items and randomly selected items hints that the response pattern depends on not only the observed data. Also, a survey on the Yahoo! LanuchCast provides evidence that the response probability might depend on the fondness of particular items [16, 26].

3 Models and Analysis

In the following, we first review the Probabilistic Matrix Factorization (PMF). After that, we present the response aware PMF and show how it can incorporate PMF with the response models. More specifically, we introduce two response models, the rating dominant response model and the context-aware response model. The updating rules and complexity analysis are provided correspondingly.

3.1 Probabilistic Matrix Factorization

PMF [22] is one of the most famous matrix factorization models in collaborative filtering, which decomposes the partially observed data matrix X into the product of two low-rank latent feature matrices, U and V, where $U \in \mathbb{R}^{K\times N}$, $V \in \mathbb{R}^{K\times M}$, and $K \ll \min(N, M)$.

By assuming Gaussian distribution on the residual noise of observed data and placing Gaussian priors on the latent feature matrices, PMF tries to maximize the log-likelihood of the posterior distribution on the user and item features as follows:

$$\mathcal{L}_{PMF} = -\sum_{(i,j,x)\in\mathcal{Q}} \frac{(x - U_i^T V_j)^2}{2\sigma^2} - \frac{\|U\|_F^2}{2\sigma_U^2} - \frac{\|V\|_F^2}{2\sigma_V^2}. \tag{6}$$

This is equivalent to minimizing a squared loss with regularization defined as follows:

$$\mathcal{E} = \frac{1}{2}\sum_{(i,j,x)\in\mathcal{Q}} (x - U_i^T V_j)^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2, \tag{7}$$

where $\lambda_U = \sigma^2/\sigma_U^2$ and $\lambda_V = \sigma^2/\sigma_V^2$ are positive constants to control the trade-off between the loss and the regularization terms. $\|\cdot\|_F^2$ denotes the Frobenious norm.

After training the PMF model via gradient descent or stochastic gradient algorithms [22], the predicted rating that user i would assign to item j can be computed as the expected mean of the Gaussian distribution $\hat{x}_{ij} = U_i^T V_j$.

3.2 Response Aware PMF

In Sec. 2, we have demonstrated that by neglecting response patterns, not only do we lose the potential information that might boost the model performance, but also can it lead to incorrect or biased parameters estimation. Due to the effectiveness and interpretability of PMF, we will unify it with explicit response models, which we refer to as Response Aware PMF (RAPMF).

Replacing θ in Eq. (2) by the low-rank latent feature matrices in PMF, we have

$$P(R, X|U, V, \mu, \sigma^2) = P(R|X, U, V, \mu, \sigma^2)P(X|U, V, \sigma^2). \tag{8}$$

The probability of full model, $P(R, X|U, V, \mu, \sigma^2)$, is decomposed into data model $P(X|U, V, \sigma^2)$ and the missing data model $P(R|X, U, V, \mu, \sigma^2)$.

3.3 Response Model

Modeling the missing data successfully requires a correct and tractable distribution on the response patterns. Bernoulli distribution is an intuitive distribution to explain data missing phenomena [16]. Depending on whether users' and items' features are incorporated, we propose two response models, *rating dominant response model* and *context-aware response model*.

3.4 Rating Dominant Response Model

For the sake of simplification, we assume the probability that a user chooses to rate an item follows a Bernoulli distribution given the rating assigned is k. Hence, for a scale of 1 to D, the rating dominant response model has D parameters $\mu_1, \mu_2, \cdots, \mu_D$.

If X is fully observed, then the response mechanism can be modeled as [16]:

$$\begin{aligned} &P(R|X, U, V, \mu, \sigma^2) = P(R|X, \mu) \\ &= \prod_{i=1}^{N}\prod_{j=1}^{M}\prod_{k=1}^{D} (\mu_k^{[r_{ij}=1]}(1-\mu_k)^{[r_{ij}=0]})^{[x_{ij}=k]}, \end{aligned} \tag{9}$$

where $[r = 0]$ is an indicator variable that outputs 1 if the expression is valid and 0 otherwise. It is noted that Eq. (9) adopts the "winner-take-all" scheme, i.e., a hard assignment scheme, to model users' response on a particular rating.

However, in real-world recommender system, the data is not fully observed. The "winner-take-all" scheme brings the risk of deteriorating assignment probability when the data is recovered based on the learned model. Hence, we adopt a soft assignment using probability of the possible rating values in the response model as follows:

$$\begin{aligned} &P(R|X, U, V, \mu, \sigma^2) = P(R|U, V, \mu, \sigma^2) \\ &= \prod_{i=1}^{N}\prod_{j=1}^{M}\sum_{k=1}^{D} (\mu_k^{[r_{ij}=1]}(1-\mu_k)^{[r_{ij}=0]})P(x_{ij} = k|U, V, \sigma^2), \end{aligned} \tag{10}$$

where $P(X|U, V, \sigma^2)$, the probability of X being assigned to k, can be set to $\mathcal{N}(k|U^T V, \sigma^2)$ as is in [22].

To relieve the inaccuracy issue when recovering the original model, we further introduce a discount parameter β on the assignment probability

$$P(R|X, U, V, \mu, \sigma^2) = P(R|U, V, \mu, \sigma^2) \tag{11}$$

$$\propto \prod_{i=1}^{N}\prod_{j=1}^{M} (\sum_{k=1}^{D} (\mu_k^{[r_{ij}=1]}(1-\mu_k)^{[r_{ij}=0]})\mathcal{N}(k|U^T V, \sigma^2))^{\beta}, \tag{12}$$

where the parameter β, in the range of 0 to 1, can be interpreted as the faith we have on the response model relative to the data model. As β decreases, the effect of the response model decreases correspondingly. When $\beta = 0$, the RAPMF collapses to PMF.

More importantly, the expectations of Bernoulli distributions, μ_k's should be in the range of 0 to 1. With the performance consideration, the logistic function is usually adopted to constrain the range of μ_k's [15],

$$g(\mu_k) = \frac{1}{1+\exp(-\mu_k)}, \quad k = 1, \ldots, D. \tag{13}$$

Similarly, we place a zero mean Gaussian prior on μ_k to regularize it.

Note that in Eq. (10), we use only one parameter for each possible rating value, so all the users and items share the same probability as long as the rating values are the same. This is a simple approach to capture the intuition that the rating assigned to an item may influence the chance that it got rated. This motivates us to name this model as *rating dominant response model*. We refer to PMF with Rating dominate response model as RAPMF-r.

By incorporating the response model in Eq. (12) and the PFM model in Eq. (6) into RAPMF in Eq. (8), we obtain the log-likelihood of the RAPMF-r as follows:

$$\begin{aligned}
&\mathcal{L}(U, V, \sigma^2, \mu) \\
&= \beta \sum_{i=1}^{N}\sum_{j=1}^{M} \log(\sum_{k=1}^{D} \alpha_{kij} \mathcal{N}(k|U^T V, \sigma^2)) - \frac{1}{2\sigma_\mu^2}\|\mu\|^2 - \\
&\sum_{(i,j,x)\in Q} \frac{(x_{ij} - U_i^T V_j)^2}{2\sigma^2} - \frac{1}{2\sigma_U^2}\|U\|_F^2 - \frac{1}{2\sigma_V^2}\|V\|_F^2 + C,
\end{aligned} \tag{14}$$

where C denotes the constant terms and α_{kij} is defined as

$$\alpha_{kij} = (g(\mu_k)^{[r_{ij}=1]}(1 - g(\mu_k))^{[r_{ij}=0]}). \tag{15}$$

The gradient of $\mathcal{L}$ with respect to U_i is:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial U_i} =& -\beta \sum_{j=1}^{M} \frac{\sum_{k=1}^{D} \alpha_{kij} \mathcal{N}(k|U^T V, \sigma^2)(U_i^T V_j - k) V_j}{\sum_{k=1}^{D} \alpha_{kij} \mathcal{N}(k|U^T V, \sigma^2)} \\
&- \sum_{j=1}^{M} (U_i^T V_j - x_{ij})[r_{ij} = 1] V_j - \lambda_U U_i.
\end{aligned} \tag{16}$$

Similarly, the gradient of $\mathcal{L}$ with respect to V_j is:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial V_j} =& -\beta \sum_{i=1}^{N} \frac{\sum_{k=1}^{D} \alpha_{kij} \mathcal{N}(k|U^T V, \sigma^2)(U_i^T V_j - k) U_i}{\sum_{k=1}^{D} \alpha_{kij} \mathcal{N}(k|U^T V, \sigma^2)} \\
&- \sum_{j=1}^{M} (U_i^T V_j - x_{ij})[r_{ij} = 1] U_i - \lambda_V V_j.
\end{aligned} \tag{17}$$

Both Eq. (16) and Eq. (17) consist of three terms. The first term corresponds to the change due to the response model, the second term is the change due to the data model and third is a regularization to avoid overfitting. Note that by adjusting β, we effectively alter the weight of the response model when updating parameters.

Finally, the gradient of $\mathcal{L}$ with respect to μ_l is

$$\frac{\partial \mathcal{L}}{\partial \mu_l} = \sum_{i=1}^{N}\sum_{j=1}^{M} \frac{\mathcal{N}(l|U^T V, \sigma^2) g'(\mu_l)(-1)^{[r_{ij}=0]}}{\sum_{k=1}^{D} \alpha_{kij} \mathcal{N}(k|U^T V, \sigma^2)} - \lambda_\mu \mu_l, \tag{18}$$

where $g'(x)$ is the derivative of the sigmoid function $g(x)$. In Eq. (16), (17), (18), $\lambda_U = \sigma^2/\sigma_U^2, \lambda_V = \sigma^2/\sigma_V^2$ and $\lambda_\mu = \sigma^2/\sigma_\mu^2$ and a multiplicative constant $1/\sigma^2$ is dropped in all three equations.

To learn model parameters, we alternatively update U, V and μ using the gradient algorithm with a learning rate η by maximizing the log-likelihood. First we update U, V by

$$U_i \leftarrow U_i + \eta \frac{\partial \mathcal{L}}{\partial U_i}, \qquad V_j \leftarrow V_j + \eta \frac{\partial \mathcal{L}}{\partial V_j}. \tag{19}$$

Then using the updated U, V, we update μ_l by

$$\mu_l \leftarrow \mu_l + \eta \frac{\partial \mathcal{L}}{\partial \mu_l}. \tag{20}$$

Similar to PMF [22], we linearly map the rating values in $[1, D]$ to $[0, 1]$ and pass $U_i^T V_j$ through the sigmoid function as defined in Eq. (13). To avoid cluttered notations, we drop all the logistic function in our derivation process. After obtained the trained model, we convert the expected value, $g(U_i^T V_j)$, back to the scale of 1 to D and set it as the predicted score of user i's rating on item j.

3.5 Context aware response model

In real-world recommender systems, the probability of an item being rated may not only depend on users' rating score. Many factors affect the response probabilities. For example, in a movie rating system, some popular movies such as Titanic, Avatar, may have much higher probability of being rated than a mediocre movie. Moreover, the features of users and items may contain group structure [30]. One may argue this might be caused by the higher inspection rate, i.e., it is likely that a reputable movie is being watched more than an obscure one. Nevertheless, it still makes sense that some items may have higher chance of receiving a rating due to the high quality that a user will not hesitate to rate it. In addition, different user may have distinct rating habits. Some users might be more willing to provide ratings in order to get high quality

recommendation. This is supported by the fact that the number of ratings received from different users can differ wildly in real-world deployed recommender systems.

To capture such factors, we generalize the rating dominant response model by including both item features and user features. To keep the model tractable and efficient, we introduce a linear combination of the item features, user features and a constant related to the rating scores and pass it through the logistic function to model the response probability,

$$\mu_{ijk} = \frac{1}{1+\exp(-(\delta_k + U_i^T\boldsymbol{\theta}_U + V_j^T\boldsymbol{\theta}_V))}. \quad (21)$$

We refer to Eq. (21) as context-aware response model, in which the response probability is on a per-user-item-rating basis. More sophisticated relationship definition can be referred to [28, 29, 31]. The PMF integrated with the context-aware response model is named RAPMF-c. Note that by setting $\boldsymbol{\theta}_U$ and $\boldsymbol{\theta}_V$ to zero, we can recover the rating dominant response model in Eq. (13).

The log-likelihood of RAPMF-c is in the same structure as RAPMF-r. We only need to substitute μ_k in Eq. (15) by μ_{ijk} defined in Eq. (21). Similarly, the gradients of $\mathcal{L}$ with respect to U_i and V_j are

$$\frac{\partial \mathcal{L}}{\partial U_i} = \beta \sum_{j=1}^{M} \frac{\sum_{k=1}^{D} t_{Ukij}\mathcal{N}(k|U^TV,\sigma^2)}{\sum_{k=1}^{D}\alpha_{kij}\mathcal{N}(k|U^TV,\sigma^2)} - \sum_{j=1}^{M}(U_i^TV_j - x_{ij})[r_{ij}=1]V_j - \lambda_U U_i, \quad (22)$$

$$\frac{\partial \mathcal{L}}{\partial V_j} = -\beta \sum_{i=1}^{N} \frac{\sum_{k=1}^{D} t_{Vkij}\mathcal{N}(k|U^TV,\sigma^2)}{\sum_{k=1}^{D}\alpha_{kij}\mathcal{N}(k|U^TV,\sigma^2)} - \sum_{j=1}^{M}(U_i^TV_j - x_{ij})[r_{ij}=1]U_i - \lambda_V V_j, \quad (23)$$

where the t_{Ukij} and t_{Vkij} is defined as following

$$t_{Ukij} = g'(\mu_{kij})(-1)^{[r_{ij}=0]}\boldsymbol{\theta}_U - \alpha_{kij}(U_i^TV_j - k)V_j, \quad (24)$$

$$t_{Vkij} = g'(\mu_{kij})(-1)^{[r_{ij}=0]}\boldsymbol{\theta}_V - \alpha_{kij}(U_i^TV_j - k)U_i. \quad (25)$$

Correspondingly, the gradients of $\mathcal{L}$ with respect to δ_l, $\boldsymbol{\theta}_U$ and $\boldsymbol{\theta}_V$ are

$$\frac{\partial \mathcal{L}}{\partial \delta_l} = \sum_{i=1}^{N}\sum_{j=1}^{M} \frac{\mathcal{N}(l|U^TV,\sigma^2)g'(\mu_{kij})(-1)^{[r_{ij}=0]}}{\sum_{k=1}^{D}\alpha_{kij}\mathcal{N}(k|U^TV,\sigma^2)} - \lambda_\mu\delta_l, \quad (26)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_U} = \sum_{i=1}^{N}\sum_{j=1}^{M} \frac{\sum_{k=1}^{D}\mathcal{N}(k|U^TV,\sigma^2)g'(\mu_{kij})(-1)^{[r_{ij}=0]}U_i}{\sum_{k=1}^{D}\alpha_{kij}\mathcal{N}(k|U^TV,\sigma^2)} - \lambda_\mu\boldsymbol{\theta}_U, \quad (27)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_V} = \sum_{i=1}^{N}\sum_{j=1}^{M} \frac{\sum_{k=1}^{D}\mathcal{N}(k|U^TV,\sigma^2)g'(\mu_{kij})(-1)^{[r_{ij}=0]}V_j}{\sum_{k=1}^{D}\alpha_{kij}\mathcal{N}(k|U^TV,\sigma^2)} - \lambda_\mu\boldsymbol{\theta}_V. \quad (28)$$

To learn RAPMF-c, we adopt the alternatively updating scheme to maximize the log-likelihood, where the updating rules of U and V are the same as those in Eq. (19). After updating U and V, we update δ_l, $\boldsymbol{\theta}_U$ and $\boldsymbol{\theta}_V$ by

$$\boldsymbol{\vartheta} \leftarrow \boldsymbol{\vartheta} + \eta\frac{\partial \mathcal{L}}{\partial \boldsymbol{\vartheta}},$$

where $\boldsymbol{\vartheta}$ is replaced by δ_l, $\boldsymbol{\theta}_U$ and $\boldsymbol{\theta}_V$, respectively.

3.6 Complexity and Parallelization

The training complexity of RAPMF, $O(MN)$, can be quite time consuming compared with the PMF, which is linear in number of observations, $O(|\mathcal{Q}|)$. However, we argue that the time spent on training is worthy since it can boost the model performance. More importantly, the prediction complexity of RAPMF is the same as PMF, $O(K)$, which can be taken as a constant time given a moderate sized K. Since the training procedure can be performed offline, RAPMF can accommodate the hard response time constraint in real-world deployed recommender systems due to the succinct prediction cost.

In addition, RAPMF can be speedup by parallelization. The intensive computation cost, calculating the gradients, can be decoupled and distributed to a cluster of computers. It is also possible to use online learning to speed up the training process [12].

4 Experiments and Results

We conduct empirical evaluation to compare the performance of PMF [22], CPT-v [16], Logit-vd [15], and our RAPMF. We try to answer the following questions:

1. How to collect data with benchmark response patterns to evaluate the models fairly?

2. How to design experiment protocols to evaluate the performance the models with and without response models fairly?
3. How the compared models perform on the collected data?
4. How the parameters, β and λ, affect the performance of RAPMF?

Section 4.1-4.4 answer the above questions, respectively.

4.1 Datasets

We conduct our empirical analysis on two datasets: a synthetic dataset and a real-world dataset, the *Yahoo! Music ratings for User Selected and Randomly Selected songs, version 1.0* (Yahoo dataset)[1].

Synthetic dataset. The data generation process consists of two steps: generating full rating matrix and generating response matrix. To generate the full rating matrix, we first generate the latent user features and item features from zero-mean spherical Gaussian as follows:

$$U_i \sim \mathcal{N}(\mathbf{0}_K, \sigma_U^2 \mathbf{I}_K), \quad V_j \sim \mathcal{N}(\mathbf{0}_K, \sigma_V^2 \mathbf{I}_K),$$

where $i = 1, \ldots, N$, $j = 1, \ldots, M$, $\mathbf{0}_K$ is a K-dimensional vector with each element being 0 and $\mathbf{I}_K$ is the $K \times K$ identity matrix. The full rating matrix X is then obtained by re-scale the sigmoid value of $U^T V$ to 1 to D by $X_{ij} = \lceil g(U_i^T V_j) \times D \rceil$.

To generate the response matrix R, we first set the inspection probability of a user inspecting an item, $P_{inspect}$. Then, the partitioning of inspected ratings and un-inspected ratings are done by the Bernoulli trails with success probability $P_{inspect}$. For all the inspected ratings, we model their response probability by a Bernoulli distribution with the success probability P_k, where $k \in \{1, 2, \ldots, D\}$. Table 2 summarizes the parameters used for generating the synthetic dataset. The parameters are selected so that they can faithfully simulate real users' ratings and response behaviors. The rating probabilities P_k are chosen according to Fig. 1. To minimize the effect of randomness, we generate the dataset independently 10 times and report the average result in the following. On average, we provide about 3.3% of the full matrix as training set, around 3.4% as testing set for traditional protocol, around 17.3% as testing set for adversarial protocol and all the remaining 80% as testing set for realistic protocol.

Yahoo dataset. It provides a unique opportunity to investigate the response ignorance problem. The dataset contains 311,704 *training ratings* collected from 15,400 users on 1,000 songs during the normal interaction between the users and the Yahoo! Music system, with at least 10 ratings for each user. During a survey conducted by Yahoo! Research, exactly 10 songs randomly selected from these 1,000 songs are presented to the user to listen and rate. In total there are 5,400 users participated this survey and these 54,000 ratings are the *testing ratings*.

[1]http://webscope.sandbox.yahoo.com

Table 2: Parameters for generating the synthetic dataset.

N	M	D	K	$P_{inspect}$
1000	1000	5	5	0.2
P_1	P_2	P_3	P_4	P_5
0.073	0.068	0.163	0.308	0.931

4.2 Setup and Evaluation Metrics

In a real-world deployed recommender system, the status of an item given a user follows exactly one of the three types: *un-inspected*, *inspected-unrated*, and *inspected-rated*. Traditional collaborative filtering approaches separate the inspected-rated data into training set and test set and evaluate the model on the test set. Since both the training set and the test set belongs to the inspected-rated type, their rating distributions are the same. Thus, the traditional evaluation scheme may hide the significance of the ignoring response model. In the experiment, we first investigate two existing experimental protocols:

- **Traditional protocol:** Both the training set and the test set are randomly selected from inspected-rated items together with their rating users and assigned scores. This is exactly the traditional experiment protocol [22].
- **Realistic protocol:** The training set is randomly selected from inspected-rated items, but the test set is randomly selected from un-inspected items. This is an experimental protocol adopted in [15, 16]. This protocol captures the ultimate goal of a recommender system, i.e., recommending un-inspected items to potential users who are interested.

Moreover, we will investigate a new experimental protocol:

- **Adversarial protocol:** The training set is randomly selected from inspected-rated items, but the test set is randomly selected from inspected-unrated items. This setting tests the model's performance when the distribution of the training set and test are very divergent. It can reveal the property of the model in some real-world cases where

most of inspected-rated items receive very high scores, while those inspected-unrated items have low scores. This setting also demonstrates the model performance when we have an adversary that manipulates the responses.

For the synthetic dataset, we use the same training set for all three protocols. For various protocols, different test set can reveal different properties of the PMF with and without the response models. We report the average performance on the 10 independently generated datasets.

For Yahoo dataset, we use only traditional and realistic protocol and do not evaluate the adversarial protocol due to the missing of necessary inspection information. For the traditional protocol, we perform 10 fold cross validation on the training ratings. For the realistic protocol, we train the model using training ratings and test on the testing ratings. We perform the experiment 10 times and report the average results.

In the experiment, we use Root Mean Square Error (RMSE) to evaluate the performance of various approaches [16, 22, 27], i.e., RMSE $= \sqrt{1/|\mathcal{T}| \sum_{(i,j,x)\in\mathcal{T}} (\hat{x}_{ij} - x)^2}$, where $\mathcal{T}$ is the set of (i, j, x) triplets reserved for testing and $\hat{x}_{ij}$ is model prediction for user i's rating on item j.

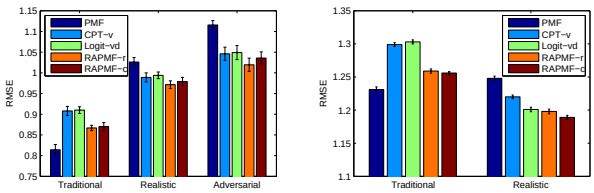
4.3 Model Comparison

For both the synthetic dataset and Yahoo dataset, we randomly select 10% of the testing ratings from realistic protocol as validation set to tune the parameters (more advanced techniques can be referred to [20]):

- λ_U and λ_V: They are tuned by the grid search scheme, i.e., first selecting from $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10, 10^2\}$, respectively. We then fine-tune the range to achieve the best performance of PMF; see an example in Fig. 3(a).
- β: We first fix the optimal λ_U and λ_V obtained from PMF, we then tune it in $\{0.0, 10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$ and fine-tune it further for RAPMF-r; see an example in Fig. 3(b).
- λ_μ: As shown later in Fig. 3(c), this parameter is insensitive on a large range.

These parameters are then used across different protocols. The hyper-parameters used for CPT-v and Logit-vd follow the settings used in [15]. We choose $K = 5$ as the latent dimension size for all the experiments. All the models are trained using 500 iterations. According to our experience, the change in performance after 200 iterations is negligible.

Figure 2 shows the results of various models' performance under different protocols on both the synthetic and Yahoo datasets. Figure 2(a) shows the results on the synthetic datasets. We see that PMF performs best under traditional protocol. This is expected because under the traditional setting, the testing set and the training set have exactly the same distribution. The response model does not help. However, under realistic and adversarial protocol, the proposed RAPMF-r outperforms PMF by 5.5% and 9.2%, with 95% confidence level on the paired t-test, respectively. The RAPMF-c performs slightly worse than RAPMF-r. This is probably due to the reason that we does not take the user and item features into account when generating the dataset.



(a) Synthetic dataset (b) Yahoo dataset

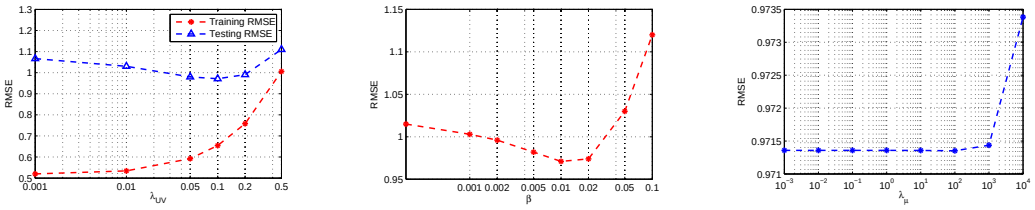
Figure 2: Relative performance of various models. Smaller value indicates a better model performance.

More importantly, the learned rating probability for a typical run of RAPMF-r is [0.0125, 0.0124, 0.0155, 0.0267, 0.105]. Comparing this with the parameter used in Table 2 when generating the data, we see that although RAPMF cannot recover the rating probabilities exactly, the overall trend is captured quite precisely. This explains the significant performance boost in realistic and adversarial protocol.

Figure 2(b) shows the results on the Yahoo dataset. Again, PMF attains the best performance under traditional protocol. Under realistic protocol, the RAPMF-r and RAPMF-c outperforms PMF by 4.1% and 4.9%, with 95% confidence level on the paired t-test, respectively. The performance gain is slightly less than that in the synthetic dataset, probably due to the reason that the rating probability in real-world dataset is not as dynamic as the value we choose in Table 2. The performance boost gained from context-awareness is not as significant as we have expected. This result hints that the rating value might impact a user's decision on whether to rate the item more than the user and item features.

4.4 Sensitivity Analysis

In the following, we investigate how the model parameters affect the performance of RAPMF. All the sensitivity analysis is done under the realistic setting in one-trial of the generated synthetic dataset.



(a) λ_{UV} (b) β (c) λ_μ

Figure 3: Sensitivity analysis of hyper-parameters on RAPMF on one-trial test.

4.4.1 Impact of β

The faith parameter β is arguably the most important parameter in our RAPMF model. As has been discussed in Section 3, we adopt a soft-margin approximation of the ideal hard-margin response model. When making this approximation, we unavoidably introduce some bias in the response model because we cannot fully recover the true U and V. Hence, we introduce β to control the weight of the response model.

Figure 3(b) plots the performance of RAPMF versus β on the logarithmic scale. When $\beta = 0$, RAPMF fall back to PMF, whose performance is 1.015, corresponding to the most left point in Fig. 3(b). Clearly, by incorporating an explicit response model, RAPMF is able to beat PMF by a large margin (nearly 5%) by using a proper β value. However, if we use too large a β, we quickly lose the boost provided by the response model. An observation of the experiment is that when β is too large, the model does not converge. This is because the model training starts from randomly initialized U and V, a large weight on the response model will pull the model away from the true model and cause divergence.

4.4.2 Impact of λ's

The regularization parameters λ are placed on U, V and μ. Since in the dataset, users and items are symmetric, we use the same regularization parameter λ_{UV} for U and V and use another parameter λ_μ to control μ.

Figure 3(a) shows the impact of λ_{UV} on the performance of RAPMF. When λ_{UV} is very small, although the RAPMF is able to fit the training data very well, it does not generalize well to the test set. This is a sign of over-fitting. As λ_{UV} becomes larger, which limits the norms of U and V, the training RMSE increases but the test RMSE decreases gradually. However, after a turning point, both the training RMSE and test RMSE start to increase. This is when the regularization is too stringent that it hinders the proper fitting of the model.

Figure 3(c) shows the impact of λ_μ on the model performance. As we can see, it is basically a straight line in a large range from 10^{-3} to 10^4, while the RMSE is changed only from 0.9714 to 0.9734, a very small scale. The effect of λ_μ is inappreciable. This is probably due to the fact that μ is a parameter in D-dimension (D=5) and no significant over-fitting can occur.

5 Conclusion and Future Work

In this paper, we propose two response models, rating dominant and context-aware response models, to capture users' response patterns. Further, we unify the response models with one of famous collaborative filtering model-based methods, the Probabilistic Matrix Factorization, to establish the Response Aware Probabilistic Matrix Factorization framework (RAPMF). The RAPMF also generalizes PMF as its special case. Empirically, we verify the performance of RAPMF under carefully designed experimental protocols and show that RAPMF performs best when it tries to fulfill the ultimate goal of real-world recommender systems, i.e., recommending items to those who may be interested in. The empirical evaluation demonstrates the potential of our RAPMF model in real-world recommender system deployment.

There are several interesting directions worthy of considering for future study. One direction is to study how to model the response when the response patterns are hidden. The second fascinating avenue is to study how to speed up our RAPMF through parallelization, online learning, or sampling techniques. The third direction is to design a smart way to efficiently tune the hyper-parameters or to design the learning scheme to automatically learn the model parameters.

6 Acknowledgments

The work described in this paper was fully supported by the Shenzhen Major Basic Research Program (Project No. JC201104220300A) and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. CUHK413210, CUHK415311 and N_CUHK405/11).

References

[1] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[2] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*, Toronto, Canada, 2012.

[3] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22:143–177, January 2004.

[4] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237, 1999.

[5] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, January 2004.

[6] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[7] Rong Jin, Luo Si, and ChengXiang Zhai. Preference-based graphic models for collaborative filtering. In *UAI*, pages 329–336, 2003.

[8] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, New York, NY, USA, 2008. ACM.

[9] Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, 2010.

[10] Yanen Li, Jia Hu, ChengXiang Zhai, and Ye Chen. Improving one-class collaborative filtering by incorporating rich user information. In *CIKM*, pages 959–968, 2010.

[11] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7:76–80, January 2003.

[12] Guang Ling, Haiqin Yang, Irwin King, and Michael R. Lyu. Online learning for collaborative filtering. In *WCCI*, Brisbane, Australia, 2012.

[13] Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data, Second Edition*. Wiley-Interscience, 2 edition, September 2002.

[14] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *WSDM*, pages 287–296, 2011.

[15] Benjamin M. Marlin and Richard S. Zemel. Collaborative prediction and ranking with non-random missing data. In *RecSys*, pages 5–12, 2009.

[16] Benjamin M. Marlin, Richard S. Zemel, Sam T. Roweis, and Malcolm Slaney. Collaborative filtering and the missing at random assumption. In *UAI*, pages 267–275, 2007.

[17] Rong Pan and Martin Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *KDD*, pages 667–676, 2009.

[18] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *ICDM*, pages 502–511, 2008.

[19] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *UAI*, pages 473–480, 2000.

[20] Steffen Rendle. Learning recommender systems with adaptive regularization. In *WSDM*, pages 133–142, 2012.

[21] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90, 2010.

[22] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[23] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887, 2008.

[24] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, pages 791–798, 2007.

[25] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[26] Harald Steck. Training and testing of recommender systems on data missing not at random. In *KDD*, pages 713–722, 2010.

[27] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456, 2011.

[28] Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, pages 1175–1182, Haifa, Israel, 2010.

[29] Haiqin Yang, Irwin King, and Michael R. Lyu. *Sparse Learning Under Regularization Framework*. LAP Lambert Academic Publishing, first edition, April 2011.

[30] Haiqin Yang, Zenglin Xu, Irwin King, and Michael R. Lyu. Online learning for group lasso. In *ICML*, pages 1191–1198, 2010.

[31] Haiqin Yang, Zenglin Xu, Jieping Ye, Irwin King, and Michael R. Lyu. Efficient sparse generalized multiple kernel learning. *IEEE Transactions on Neural Networks*, 22(3):433–446, 2011.

[32] Yu Zhang, Bin Cao, and Dit-Yan Yeung. Multi-domain collaborative filtering. In *UAI*, pages 725–732, 2010.

Graphical-model Based Multiple Testing under Dependence, with Applications to Genome-wide Association Studies

Jie Liu
Computer Sciences, UW-Madison

Chunming Zhang
Statistics, UW-Madison

Catherine McCarty
Essentia Institute of Rural Health

Peggy Peissig
Marshfield Clinic Research Foundation

Elizabeth Burnside
Radiology, UW-Madison

David Page
BMI & CS, UW-Madison

Abstract

Large-scale multiple testing tasks often exhibit dependence, and leveraging the dependence between individual tests is still one challenging and important problem in statistics. With recent advances in graphical models, it is feasible to use them to perform multiple testing under dependence. We propose a multiple testing procedure which is based on a Markov-random-field-coupled mixture model. The ground truth of hypotheses is represented by a latent binary Markov random field, and the observed test statistics appear as the coupled mixture variables. The parameters in our model can be automatically learned by a novel EM algorithm. We use an MCMC algorithm to infer the posterior probability that each hypothesis is null (termed *local index of significance*), and the false discovery rate can be controlled accordingly. Simulations show that the numerical performance of multiple testing can be improved substantially by using our procedure. We apply the procedure to a real-world genome-wide association study on breast cancer, and we identify several SNPs with strong association evidence.

1 Introduction

Observations from large-scale multiple testing problems often exhibit dependence. For instance, in genome-wide association studies, researchers collect hundreds of thousands of highly correlated genetic markers (single-nucleotide polymorphisms, or SNPs) with the purpose of identifying the subset of markers associated with a heritable disease or trait. In functional magnetic resonance imaging studies of the brain, thousands of spatially correlated voxels are collected while subjects are performing certain tasks, with the purpose of detecting the relevant voxels. The most popular family of large-scale multiple testing procedures is the false discovery rate analysis, such as the p-value thresholding procedures (Benjamini & Hochberg, 1995, 2000; Genovese & Wasserman, 2004), the local false discovery rate procedure (Efron et al., 2001), and the positive false discovery rate procedure (Storey, 2002, 2003). However, all these classical multiple testing procedures ignore the correlation structure among the individual factors, and the question is *whether we can reduce the false non-discovery rate by leveraging the dependence, while still controlling the false discovery rate in multiple testing.*

Graphical models provide an elegant way of representing dependence. With recent advances in graphical models, especially more efficient algorithms for inference and parameter learning, it is feasible to use these models to leverage the dependence between individual tests in multiple testing problems. One influential paper (Sun & Cai, 2009) in the statistics community uses a hidden Markov model to represent the dependence structure, and has shown its optimality under certain conditions and its strong empirical performance. It is the first graphical model (and the only one so far) used in multiple testing problems. However, their procedure can only deal with a sequential dependence structure, and the dependence parameters are homogenous. *In this paper, we propose a multiple testing procedure based on a Markov-random-field-coupled mixture model which allows arbitrary dependence structures and heterogeneous dependence parameters.* This extension requires more sophisticated algorithms for parameter learning and inference. For parameter learning, we design an EM algorithm with MCMC in the E-step and persistent contrastive divergence algorithm (Tieleman, 2008) in the M-step. We use the MCMC algorithm to infer the posterior probability that each hypothesis is null (termed *local index of significance* or LIS). Finally, the false discovery rate can be controlled by thresholding the LIS.

Section 2 introduces related work and our procedure. Sections 3 and 4 evaluate our procedure on a variety of simulations, and the empirical results show that the numerical performance can be improved substantially by using our procedure. In Section 5, we apply the procedure to a real-world genome-wide association study (GWAS) on breast cancer, and we identify several SNPs with strong association evidence. We finally conclude in Section 6.

2 Method

2.1 Terminology and Previous Work

Table 1: Classification of tested hypotheses

	Not rejected	Rejected	Total
Null	N_{00}	N_{10}	m_0
Non-null	N_{01}	N_{11}	m_1
Total	S	R	m

Suppose that we carry out m tests whose results can be categorized as in Table 1. *False discovery rate* (FDR), defined as $E(N_{10}/R|R > 0)P(R > 0)$, depicts the expected proportion of incorrectly rejected null hypotheses (Benjamini & Hochberg, 1995). *False non-discovery rate* (FNR), defined as $E(N_{01}/S|S > 0)P(S > 0)$, depicts the expected proportion of false non-rejections in those tests whose null hypotheses are not rejected (Genovese & Wasserman, 2002). An FDR procedure is *valid* if it controls FDR at a nominal level, and *optimal* if it has the smallest FNR among all the valid FDR procedures (Sun & Cai, 2009). The effects of correlation on multiple testing have been discussed, under different assumptions, with a focus on the validity issue (Benjamini & Yekutieli, 2001; Finner & Roters, 2002; Owen, 2005; Sarkar, 2006; Efron, 2007; Farcomeni, 2007; Romano et al., 2008; Wu, 2008; Blanchard & Roquain, 2009). The efficiency issue has also been investigated (Yekutieli & Benjamini, 1999; Genovese et al., 2006; Benjamini & Heller, 2007; Zhang et al., 2011), indicating FNR could be decreased by considering dependence in multiple testing. Several approaches have been proposed, such as dependence kernels (Leek & Storey, 2008), factor models (Friguet et al., 2009) and principal factor approximation (Fan et al., 2012). Sun & Cai (2009) explicitly use a hidden Markov model (HMM) to represent the dependence structure and analyze the optimality under the compound decision framework (Sun & Cai, 2007). However, their procedure can only deal with sequential dependence, and it uses only a single dependence parameter throughout. In this paper, we replace HMM with a Markov-random-field-coupled mixture model, which allows richer and more flexible dependence structures. The Markov-random-field-coupled mixture models are related to the hidden Markov random field models used in many image segmentation problems (Zhang et al., 2001; Celeux et al., 2003; Chatzis & Varvarigou, 2008).

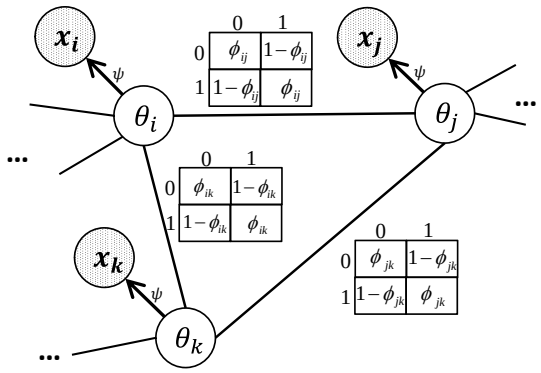


Figure 1: The MRF-coupled mixture model for three dependent hypotheses $\mathcal{H}_i$, $\mathcal{H}_j$ and $\mathcal{H}_k$ with *observed* test statistics (x_i, x_j and x_k) and *latent* ground truth (θ_i,θ_j and θ_k). The dependence is captured by potential functions parameterized by ϕ_{ij},ϕ_{jk} and ϕ_{ik}, and coupled mixtures are parameterized by ψ.

2.2 Our Multiple Testing Procedure

Let $\mathbf{x} = (x_1, ..., x_m)$ be a vector of test statistics from a set of hypotheses $(\mathcal{H}_1, ..., \mathcal{H}_m)$. The ground truth of these hypotheses is denoted by a latent Bernoulli vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_m) \in \{0, 1\}^m$, with $\theta_i = 0$ denoting that the hypothesis $\mathcal{H}_i$ is null and $\theta_i = 1$ denoting that the hypothesis $\mathcal{H}_i$ is non-null. The dependence among these hypotheses is represented as a binary Markov random field (MRF) on $\boldsymbol{\theta}$. The structure of the MRF can be described by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with the node set $\mathcal{V}$ and the edge set $\mathcal{E}$. The dependence between $\mathcal{H}_i$ and $\mathcal{H}_j$ is denoted by an edge connecting node$_i$ and node$_j$ in $\mathcal{E}$, and the strength of dependence is parameterized by the potential function on the edge. Suppose that the probability density function of the test statistic x_i given $\theta_i = 0$ is f_0, and the density of x_i given $\theta_i = 1$ is f_1. Then, $\mathbf{x}$ is an MRF-coupled mixture. The mixture model is parameterized by a parameter set $\vartheta = (\boldsymbol{\phi}, \boldsymbol{\psi})$, where $\boldsymbol{\phi}$ parameterizes the binary MRF and $\boldsymbol{\psi}$ parameterizes f_0 and f_1. For example, if f_0 is standard normal $\mathcal{N}(0, 1)$ and f_1 is noncentered normal $\mathcal{N}(\mu, 1)$, then $\boldsymbol{\psi}$ only contains parameter μ. Figure 1 shows the MRF-coupled mixture model for three dependent hypotheses $\mathcal{H}_i$, $\mathcal{H}_j$ and $\mathcal{H}_k$.

In our MRF-coupled mixture model, $\mathbf{x}$ is observable, and $\boldsymbol{\theta}$ is hidden. With the parameter set $\vartheta = (\boldsymbol{\phi}, \boldsymbol{\psi})$, the joint probability density over $\mathbf{x}$ and $\boldsymbol{\theta}$ is

$$P(\mathbf{x}, \boldsymbol{\theta}|\boldsymbol{\phi}, \boldsymbol{\psi}) = P(\boldsymbol{\theta}; \boldsymbol{\phi}) \prod_{i=1}^{m} P(x_i|\theta_i; \boldsymbol{\psi}). \quad (1)$$

Define the marginal probability that $\mathcal{H}_i$ is null given all the observed statistics $\mathbf{x}$ under the parameters in ϑ, $P_\vartheta(\theta_i = 0|\mathbf{x})$, to be the *local index of significance* (LIS) for $\mathcal{H}_i$ (Sun & Cai, 2009). If we can accurately calculate the posterior marginal probabilities of $\boldsymbol{\theta}$ (or LIS), then we can use a step-up procedure to control FDR at the nominal level α as follows (Sun & Cai, 2009). We first sort LIS from the smallest value to the largest value. Suppose $\text{LIS}_{(1)}$, $\text{LIS}_{(2)}$, ..., and $\text{LIS}_{(m)}$ are the ordered LIS, and the corresponding hypotheses are $\mathcal{H}_{(1)}$, $\mathcal{H}_{(2)}$,..., and $\mathcal{H}_{(m)}$. Let

$$k = \max\left\{i : \frac{1}{i}\sum\nolimits_{j=1}^{i} \text{LIS}_{(j)} \leq \alpha\right\}. \qquad (2)$$

Then we reject $\mathcal{H}_{(i)}$ for $i = 1, ..., k$.

Therefore, the key inferential problem that we need to solve is that of computing the posterior marginal distribution of the hidden variables θ_i given the test statistics $\mathbf{x}$, namely $P_\vartheta(\theta_i = 0|\mathbf{x})$, for $i = 1, ..., m$. It is a typical inference problem if the parameters in ϑ are known. Section 2.3 provides possible inference algorithms for calculating $P_\vartheta(\theta_i = 0|\mathbf{x})$ for given ϑ. However, ϑ is usually unknown in real-world applications, and we need to estimate it. Section 2.4 provides a novel EM algorithm for parameter learning in our MRF-coupled mixture model.

2.3 Posterior Inference

Now we are interested in calculating $P_\vartheta(\theta_i = 0|\mathbf{x})$ for a given parameter set ϑ. One popular family of inference algorithms is the sum-product family (Kschischang et al., 2001), also known as belief propagation (Yedidia et al., 2000). For loop-free graphs, belief propagation algorithms provide exact inference results with a computational cost linear in the number of variables. In our MRF-coupled mixture model, the structure of the latent MRF is described by a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. When $\mathcal{G}$ is chain structured, the instantiation of belief propagation is the forward-backward algorithm (Baum et al., 1970). When $\mathcal{G}$ is tree structured, the instantiation of belief propagation is the upward-downward algorithm (Crouse et al., 1998). For graphical models with cycles, loopy belief propagation (Murphy et al., 1999; Weiss, 2000) and the tree-reweighted algorithm (Wainwright et al., 2003a) can be used for approximate inference. Other inference algorithms for graphical models include junction trees (Lauritzen & Spiegelhalter, 1988), sampling methods (Gelfand & Smith, 1990), and variational methods (Jordan et al., 1999). Recent papers (Schraudolph & Kamenetsky, 2009; Schraudolph, 2010) discuss exact inference algorithms on binary Markov random fields which allow loops. In our simulations, we use belief propagation when the graph $\mathcal{G}$ has no loops. When $\mathcal{G}$ has loops (e.g. in the simulations on genetic data and the real-world application), we use a Markov chain Monte Carlo (MCMC) algorithm to perform inference for $P_\vartheta(\theta_i = 0|\mathbf{x})$.

2.4 Parameters and Parameter Learning

In our procedure, the dependence among these hypotheses is represented by a graphical model on the latent vector $\boldsymbol{\theta}$ parameterized by $\boldsymbol{\phi}$, and observed test statistics $\mathbf{x}$ are represented by the coupled mixture parameterized by $\boldsymbol{\psi}$. In Sun and Cai's work on HMMs, $\boldsymbol{\phi}$ is the transition parameter and $\boldsymbol{\psi}$ is the emission parameter. One implicit assumption in their work is that the transition parameter and the emission parameter stay the same for $i(i = 1, ..., m)$. Our extension to MRFs also allows us to untie these parameters. In the second set of basic simulations in Section 3, we make $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ heterogeneous and investigate how this affects the numerical performance. In the simulations on genetic data in Section 4 and the real-world GWAS application in Section 5, we have different parameters for SNP pairs with different levels of correlation.

In our model, learning $(\boldsymbol{\phi}, \boldsymbol{\psi})$ is difficult for two reasons. First, learning parameters is difficult by nature in undirected graphical models due to the global normalization constant (Wainwright et al., 2003b; Welling & Sutton, 2005). State-of-the-art MRF parameter learning methods include MCMC-MLE (Geyer, 1991), contrastive divergence (Hinton, 2002) and variational methods (Ganapathi et al., 2008). Several new sampling methods with higher efficiency have been recently proposed, such as persistent contrastive divergence (Tieleman, 2008), fast-weight contrastive divergence (Tieleman & Hinton, 2009), tempered transitions (Salakhutdinov, 2009), and particle-filtered MCMC-MLE (Asuncion et al., 2010). In our procedure, we use the persistent contrastive divergence algorithm to estimate parameters $\boldsymbol{\phi}$. Another difficulty is that $\boldsymbol{\theta}$ is latent and we only have one observed training sample $\mathbf{x}$. We use an EM algorithm to solve this problem. In the E-step, we run our MCMC algorithm in Section 2.3 to infer the latent $\boldsymbol{\theta}$ based on the currently estimated parameters $\vartheta = (\boldsymbol{\phi}, \boldsymbol{\psi})$. In the M-step, we run the persistent contrastive divergence (PCD) algorithm (Tieleman, 2008) to estimate $\boldsymbol{\phi}$ from the currently inferred $\boldsymbol{\theta}$. Note that PCD is also an iterative algorithm, and we run it until it converges in each M-step. In the M-step, we also do a maximum likelihood estimation of $\boldsymbol{\psi}$ from the currently inferred $\boldsymbol{\theta}$ and observed $\mathbf{x}$. We run the EM algorithm until both $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ converge. Although this EM algorithm involves intensive computation in both E-step and M-step, it converges very quickly in our experiments.

3 Basic Simulations

In the basic simulations, we investigate the numerical performance of our multiple testing approach on different fabricated dependence structures where we can control the ground truth parameters. We first simulate $\boldsymbol{\theta}$ from $P(\boldsymbol{\theta};\boldsymbol{\phi})$ and then simulate $\mathbf{x}$ from $P(\mathbf{x}|\boldsymbol{\theta};\boldsymbol{\psi})$ under a variety of settings of $\vartheta = (\boldsymbol{\phi},\boldsymbol{\psi})$. Because we have the ground truth parameters, we have two versions of our multiple testing approach, namely the oracle procedure (OR) and the data-driven procedure (LIS). The oracle procedure knows the true parameters ϑ in the graphical models, whereas the data-driven procedure does not and has to estimate ϑ. The baseline procedures include the BH procedure (Benjamini & Hochberg, 1995) and the adaptive p-value procedure (AP) (Benjamini & Hochberg, 2000; Genovese & Wasserman, 2004) which are compared by Sun & Cai (2009). We include another baseline procedure, the local false discovery rate procedure (localFDR) (Efron et al., 2001). The adaptive p-value procedure requires a consistent estimate of the proportion of the true null hypotheses. The localFDR procedure requires a consistent estimate of the proportion of the true null hypotheses and the knowledge of the distribution of the test statistics under the null and under the alternative. In our simulations, we endow AP and localFDR with the ground truth values of these in order to let these baseline procedures achieve their best performance.

In the simulations, we assume that the observed x_i under the null hypothesis (namely $\theta_i = 0$) is standard-normally distributed and that x_i under the alternative hypothesis (namely $\theta_i = 1$) is normally distributed with mean μ and standard deviation 1.0. We choose the setup and parameters to be consistent with the work of Sun & Cai (2009) when possible. In total, we consider three MRF models, namely a chain-structured MRF, tree-structured MRF and grid-structured MRF. For chain-MRF, we choose the number of hypotheses $m = 3,000$. For tree-MRF, we choose perfect binary trees of height 12 which yields a total number of $8,191$ hypotheses. For grid-MRF, we choose the number of rows and the number of columns to be 100 which yields a total number of $10,000$ hypotheses. In all the experiments, we choose the number of replications $N = 500$ which is also the same as the work of Sun & Cai (2009). In total, we have three sets of simulations with different goals as follows.

Basic simulation 1: We stay consistent with Sun & Cai (2009) in the simulations except that we use the three MRF models. In all three structures, $(\theta_i)_1^m$ is generated from the MRFs whose potentials on the edges are $\begin{pmatrix} \phi & 1-\phi \\ 1-\phi & \phi \end{pmatrix}$. Therefore, $\boldsymbol{\phi}$ only contains parameter ϕ, and $\boldsymbol{\psi}$ only includes parameter μ.

Basic simulation 2: One assumption in basic simulation 1 is that the parameters ϕ and μ are homogeneous in the sense that they stay the same for $i(i = 1,...,m)$. This assumption is carried down from the work of Sun & Cai (2009). However in many real-world applications, the transition parameters can be different across the multiple hypotheses. Similarly, the test statistics for the non-null hypotheses, although normally distributed and standardized, could have different μ values. Therefore, we investigate the situation where the parameters can vary in different hypotheses. The simulations are carried out for all three different dependence structures aforementioned. In the first set of simulations, instead of fixing ϕ, we choose ϕ's uniformly distributed on the interval $(0.8-\Delta(\phi)/2, 0.8+\Delta(\phi)/2)$. In the second set of simulations, instead of fixing μ, we choose μ's uniformly distributed on the interval $(2.0-\Delta(\mu)/2, 2.0+\Delta(\mu)/2)$. The oracle procedure knows the true parameters. The data-driven procedure does not know the parameters, and assumes the parameters are homogeneous.

Basic simulation 3: Another implicit assumption in basic simulation 1 is that each individual test in the multiple testing problem is exact. Many widely used hypothesis tests, such as Pearson's χ^2 test and the likelihood ratio test, are asymptotic in the sense that we only know the limiting distribution of the test statistics for large samples. As an example, we simulate the two-proportion z-test in this section and show how the sample size affects the performance of the procedures when the individual test is asymptotic. Suppose that we have n samples (half of them are positive samples and half of them are negative samples). For each sample, we have m Bernoulli distributed attributes. A fraction of the attributes are relevant. If the attribute A is relevant, then the probability of "heads" in the positive samples (p_A^+) is different from that in the negative samples (p_A^-). p_A^+ and p_A^- are the same if A is non-relevant. For each individual test, the null hypothesis is that the attribute is not relevant, and the alternative hypothesis is otherwise. The two-proportion z-test can be used to test whether $p_A^+ - p_A^-$ is zero, which yields an asymptotic $\mathcal{N}(0,1)$ under the null and $\mathcal{N}(\mu,1)$ under the alternative (μ is nonzero). In the simulations, we fix μ, but vary the sample size n, and apply the aforementioned tree-MRF structure ($m = 8,191$). The oracle procedure and localFDR only know the limiting distribution of the test statistics and assume the test statistics exactly follow the limiting distributions even when the sample size is small.

Figure 2 shows the numerical results in basic simulation 1. Figures (1a)-(1f) are for the chain structure. Figures (2a)-(2f) are for tree structure. Figures (3a)-(3f) are for the grid structure. In Figures (1a)-(1c),

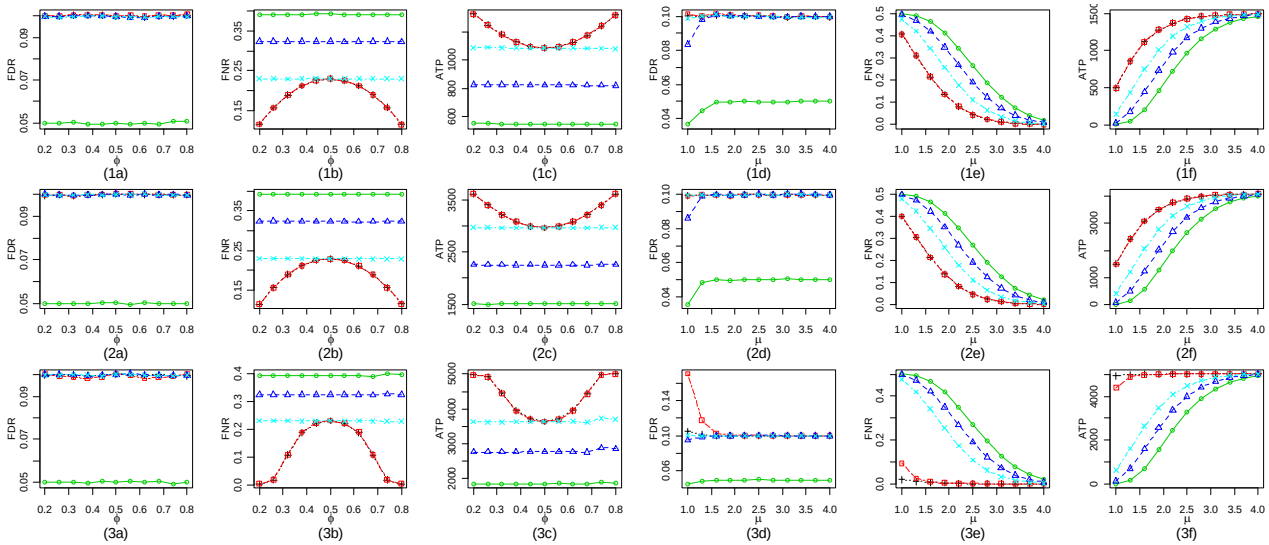


Figure 2: Comparison of BH(◯), AP(△), localFDR(×), OR (+), and LIS (□) in basic simulation 1: (1) chain-MRF, (2) tree-MRF, (3) grid-MRF; (a) FDR vs ϕ, (b) FNR vs ϕ, (c) ATP vs ϕ, (d) FDR vs μ, (e) FNR vs μ, (f) ATP vs μ.

(2a)-(2c) and (3a)-(3c), we set $\mu = 2$ and plot FDR, FNR and the average number of true positives (ATP) when we vary ϕ between 0.2 and 0.8. In Figures (1d)-(1f), (2d)-(2f) and (3d)-(3f), we set $\phi = 0.8$ and plot FDR, FNR and ATP when we vary μ between 1.0 and 4.0. The nominal FDR level is set to be 0.10. From Figure 2, we can observe comparable numerical results between the chain structure and tree structure. The FDR levels of all five procedures are controlled at 0.10 and BH is conservative. From the plots for FNR and ATP, we can observe that the data-driven procedure performs almost the same as the oracle procedure, and they dominate the p-value thresholding procedures BH and AP. The oracle procedure and the data-driven procedure also dominate localFDR except when $\phi = 0.5$, when they perform comparably. This is to be expected because the dependence structure is no longer informative when ϕ is 0.5. In this situation when the hypotheses are independent, our procedure reduces to the localFDR procedure. As ϕ departs from 0.5 and approaches either 0 or 1.0, the difference between OR/LIS and the baselines gets larger. When the individual hypotheses are easy to test (large μ values), the differences between them are not substantial. When we turn to the grid structure, the numerical performance is similar to that in the chain structure and the tree structure except for two observations. First, the data-driven procedure does not appear to control the FDR at 0.1 when μ is small (e.g. $\mu = 1.0$), although the oracle procedure does, which indicates the parameter estimation in the EM algorithm is difficult when μ is small. In other words, with a limited number of hypotheses, it is difficult to estimate the pairwise potential parameters if the test statistics of the non-nulls do not look much different from the test statistics of the nulls. The second observation is that the slopes of the FNR curve and ATP curve for the grid structure are different from those in the chain and tree structures. The reason is that the connectivity in the grid structure is higher than that in the chain and tree. Therefore we can observe that even when the individual hypotheses are difficult to test (small μ values), the FNR is still low because each individual hypothesis has more neighbors in the grid than in the chain or tree, and the neighbors are informative.

Figure 3 shows the numerical performance in basic simulation 2. Figures (1a)-(1f), (2a)-(2f), and (3a)-(3f) correspond to the chain structure, the tree structure and the grid structure respectively. In Figures (1a)-(1c), (2a)-(2c), and (3a-3c), we set $\mu = 2$ and vary $\Delta(\phi)$ between 0 and 0.4. In Figures (1d)-(1f), (2d)-(2f), and (3d)-(3f), we set $\phi = 0.8$ and vary $\Delta(\mu)$ between 0 and 4.0. Again, the nominal FDR level is set to be 0.10. From Figure 3, we observe that all five procedures control FDR at the nominal level and BH is conservative when the transition parameter ϕ is heterogeneous. However, the data-driven procedure becomes more and more conservative as we increase the variance of ϕ in the grid-structure. Nevertheless, the data-driven procedure does not lose much efficiency compared with the oracle procedure based on FNR and ATP. Both the data-driven procedure and the oracle procedure dominate the three baselines. When the μ parameter is heterogeneous, all five procedures

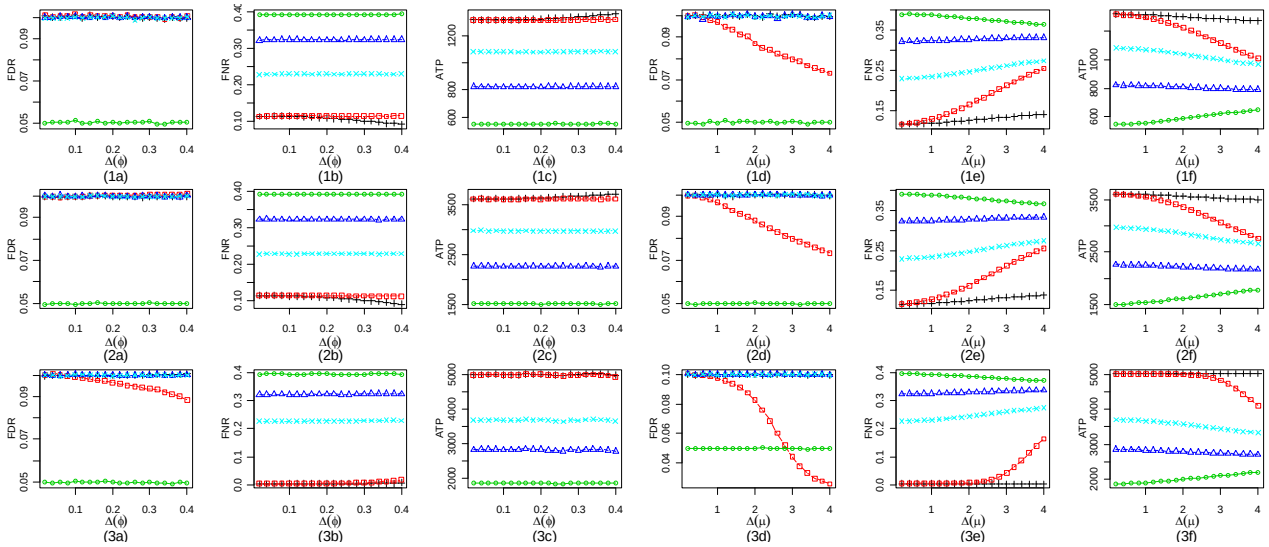


Figure 3: Comparison of BH(◯), AP(△), localFDR(×), OR (+), and LIS (□) in basic simulation 2: (1) chain-MRF, (2) tree-MRF, (3) grid-MRF; (a) FDR vs $\Delta(\phi)$, (b) FNR vs $\Delta(\phi)$, (c) ATP vs $\Delta(\phi)$, (d) FDR vs $\Delta(\mu)$, (e) FNR vs $\Delta(\mu)$, (f) ATP vs $\Delta(\mu)$.

are still valid, but the data-driven procedure becomes more and more conservative as we increase the variance of μ. The data-driven procedure can be more conservative than the BH procedure when $\Delta(\mu)$ is large enough. The conservativeness appears most severe in the grid-structure. However when we look at the FNR and ATP, the data-driven procedure still dominates BH, AP and localFDR substantially in all the situations, although the data-driven procedure loses a certain amount of efficiency compared with the oracle procedure when the variance of μ gets large.

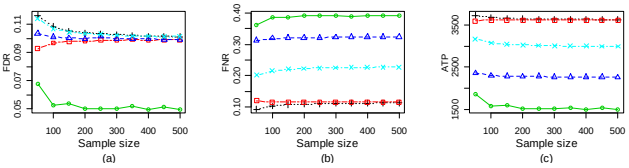


Figure 4: Comparing BH(◯), AP(△), localFDR(×), OR(+), and LIS(□) in basic simulation 3: (a)FDR vs n, (b)FNR vs n, (c)ATP vs n.

Figure 4 shows the results from basic simulation 3. The oracle procedure and localFDR are liberal when the sample size is small. This is because when the sample size is small, there exists a discrepancy between the true distribution of the test statistic and the limiting distribution. Quite surprisingly, the data-driven procedure stays valid. The reason is that the data-driven procedure can estimate the parameters from data. The data-driven procedure and the oracle procedure still have comparable performance and enjoy a much lower level of FNR compared with the baselines. For all the basic simulations, we set the nominal FDR level to be 0.10. We have also replicated the basic simulations by setting the nominal level to be 0.05, and similar conclusions can be made.

4 Simulations on Genetic Data

Unlike the fabricated dependence structures in the basic simulations in Section 3, the dependence structure in the simulations on genetic data in this section is real. We simulate the linkage disequilibrium structure of a segment on human chromosome 22, and treat a test of whether a SNP is associated as one individual test. We follow the simulation settings in the work of Wu et al. (2010). We use HAPGEN2 (Su et al., 2011) and the CEU sample of HapMap (The International HapMap Consortium, 2003) (Release 22) to generate SNP genotype data at each of the 2,420 loci between bp 14431347 and bp 17999745 on Chromosome 22. A total of 685 out of 2,420 SNPs can be genotyped with the Affymetrix 6.0 array. These are the typed SNPs that we use for our simulations. Within the overall 2,420 SNPs, we randomly select 10 SNPs to be the *causal* SNPs. All the SNPs on the Affymetrix 6.0 array whose r^2 values, according to HapMap, with any of the causal SNPs are above t are set to be the *associated* SNPs. In the simulations, we report results for three different t values, namely 0.8, 0.5 and 0.25. We also simulate three different genetic models (additive model, dominant model, and recessive model) with different levels of relative risk (1.2 and 1.3). In

total, we simulate 250 cases and 250 controls. The experiment is replicated for 100 times and the average result is provided. With the simulated data, we apply our multiple testing procedure (LIS) and three baseline procedures: the BH procedure, the adaptive p-value procedure (AP), and the local false discovery rate procedure (localFDR). Because the dependence structure is real and the ground truth parameters are unknown to us, we do not have the oracle procedure in the simulations on genetic data.

With the simulated genetic data, we use two commonly used tests in genetic association studies, namely two-proportion z-test and Cochran-Armitage's trend test (CATT) (Cochran, 1954; Armitage, 1955; Slager & Schaid, 2001; Freidlin et al., 2002) as the individual tests for the association of each SNP. CATT also yields an asymptotic $\mathcal{N}(0,1)$ under the null and $\mathcal{N}(\mu,1)$ under the alternative (μ is nonzero). Therefore, we parameterize $\boldsymbol{\psi} = (\mu_1, \sigma_1^2)$ where μ_1 and σ_1^2 are the mean and variance of the test statistics under alternative. The graph structure is built as follows. Each SNP becomes a node in the graph. For each SNP, we connect it with the SNP with the highest r^2 value with it. There are in total 490 edges in the graph. We further categorize the edges into a high correlation edge set $\mathcal{E}_h$ (r^2 above 0.8), medium correlation edge set $\mathcal{E}_m$ (r^2 between 0.5 and 0.8) and low correlation edge set $\mathcal{E}_l$ (r^2 between 0.25 and 0.5). We have three different parameters (ϕ_h, ϕ_m, and ϕ_l) for the three sets of edges. Then the density of $\boldsymbol{\theta}$ in formula (1) takes the form

$$P(\boldsymbol{\theta};\boldsymbol{\phi}) \propto \exp\{\sum_{(i,j)\in\mathcal{E}_h} \phi_h I(\theta_i=\theta_j) + \sum_{(i,j)\in\mathcal{E}_m} \phi_m I(\theta_i=\theta_j) + \sum_{(i,j)\in\mathcal{E}_l} \phi_l I(\theta_i=\theta_j)\}, \tag{3}$$

where $I(\theta_i=\theta_j)$ is an indicator variable that indicates whether θ_i and θ_j take the same value. In the MCMC algorithm, we run the Markov chain for $20,000$ iterations with a burn-in of 100 iterations. In the PCD algorithm, we generate 100 particles. In each iteration of PCD learning, the particles move forward for 5 iterations (the n parameter in PCD-n). The learning rate in PCD gradually decreases as suggested by Tieleman (2008). The EM algorithm converges after about 10 to 20 iterations, which usually take less than 10 minutes on a 3.00GHz CPU.

Figure 5 shows the performance of the procedures in the additive models with the homozygous relative risk set to 1.2 and 1.3. The test statistics are from a two-proportion z-test. We have also replicated the simulations on Cochran-Armitage's trend test, and the results are almost the same. In Figure 5, table (1) summarizes the empirical FDR and the total number of true positives (#TP) of our LIS procedure, BH, AP and localFDR (lfdr), in the additive models with different (homozygous) relative risk levels, when we vary t and when we vary the nominal FDR level α. We regard a SNP having r^2 above t with any causal SNP as an associated SNP, and we regard a rejection of the null hypothesis for an associated SNP as a true positive. Our LIS procedure and localFDR are valid while being conservative. BH and AP appear liberal in some of the configurations. In any of the circumstances, our LIS procedure can identify more associated SNPs than the baselines. We can find a clue to why our procedure LIS is being conservative from the results in Figure 3. In basic simulation 2, we observe that when the parameters μ and ϕ are heterogeneous and we carry out the data-driven procedure under the homogeneous parameter assumption, the data-driven procedure is conservative. The discrepancy between the nominal FDR level and the empirical FDR level increases as the parameters move further away from homogeneity. Although we assign three different parameters ϕ_h, ϕ_m, and ϕ_l to $\mathcal{E}_h$, $\mathcal{E}_m$ and $\mathcal{E}_l$ respectively, the edges within the same set (e.g. $\mathcal{E}_l$) may still be heterogeneous. The fact that the LIS procedure recaptures more true positives than the baselines while remaining more conservative in many configurations indicates that the local indices of significance provide a ranking more efficient than the ranking provided by the p-values from the individual tests. Therefore, we further plot the ROC curves and precision-recall (PR) curves when we rank SNPs by LIS and by the p-values from the two-proportion z-test. The ROC curve and PR curve are vertically averaged from 100 replications. Subfigures (2a)-(2f) are for the additive model with homozygous relative risk level set to be 1.2. Subfigures (3a)-(3f) are for the additive model with homozygous relative risk level set to be 1.3. It is observed that the curves from LIS dominate those from the p-values from individual tests in most places, which further suggests that LIS provides a more efficient ranking of the SNPs than the individual tests.

Figure 6 shows the performance of the procedures in the dominant model and the recessive model with the homozygous relative risk set to be 1.2. The test statistics are from a two-proportion z-test. In Figure 6, table (1) summarizes the empirical FDR and the total number of true positives (#TP) of our LIS procedure, BH, AP and localFDR (lfdr) in the dominant model and the recessive model when we vary t and when we vary the nominal FDR level α. Our LIS procedure and localFDR are valid while being conservative in all configurations, and they appear more conservative in the recessive model than in the dominant model. On the other hand, BH and AP appear liberal in the recessive model. Our LIS procedure still confers an advantage

			t = 0.8				t = 0.5				t = 0.25			
			LIS	BH	AP	lfdr	LIS	BH	AP	lfdr	LIS	BH	AP	lfdr
rr = 1.2	α = 0.05	FDR:	0.018	0.059	0.059	0.010	0.018	0.059	0.059	0.010	0.018	0.058	0.058	0.009
		#TP:	12	11	11	1	12	11	11	1	20	18	19	7
	α = 0.10	FDR:	0.077	0.089	0.089	0.010	0.077	0.089	0.089	0.010	0.076	0.079	0.079	0.009
		#TP:	13	11	11	1	13	11	11	1	21	20	20	8
rr = 1.3	α = 0.05	FDR:	0.047	0.044	0.054	0.015	0.047	0.044	0.064	0.005	0.046	0.044	0.064	0.014
		#TP:	16	4	4	1	16	4	4	1	22	10	10	6
	α = 0.10	FDR:	0.067	0.104	0.104	0.015	0.067	0.104	0.104	0.005	0.066	0.103	0.103	0.014
		#TP:	18	15	15	1	18	15	15	1	27	21	21	6

(1)

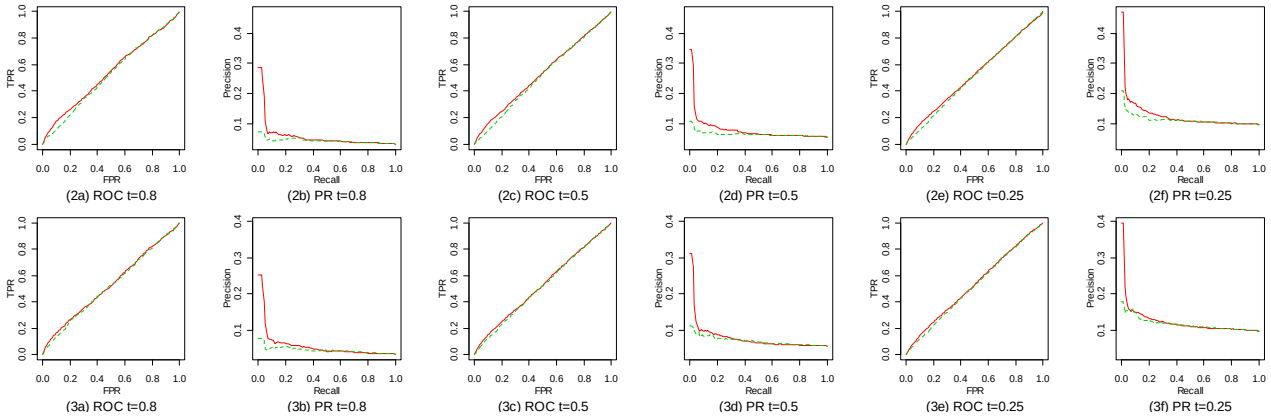


Figure 5: Comparison of BH, AP, localFDR and LIS in the additive models when we vary relative risk rr, t and the nominal FDR level α. Table (1) summarizes results. Subfigures (2a)-(2f) shows ROC and PR curves of LIS (solid red lines) and individual p-values (dashed green lines) with $rr = 1.2$. Subfigures (3a)-(3f) shows ROC and PR curves of LIS (solid red lines) and individual p-values (dashed green lines) with $rr = 1.3$.

over the baselines in the dominant model. The LIS procedure also recaptures almost the same number of true positives as BH and AP while maintaining a much lower FDR in the recessive model. Again, we further plot the ROC curves and precision-recall curves when we rank SNPs by LIS and by the p-values from individual tests. Subfigures (2a)-(2f) are for the dominant model. Subfigures (3a)-(3f) are for the recessive model. It is also observed that the curves from LIS dominate those from the p-values from individual tests in most places, which also suggests that LIS provides a more efficient ranking.

5 Real-world Application

Our primary GWAS dataset on breast cancer is from NCI's Cancer Genetics Markers of Susceptibility (CGEMS) (Hunter et al., 2007). 528,173 SNPs for 1,145 cases and 1,142 controls are genotyped on the Illumina HumanHap500 array. Our secondary GWAS dataset comes from Marshfield Clinic. The Personalized Medicine Research Project (McCarty et al., 2005), sponsored by Marshfield Clinic, was used as the sampling frame to identify 162 breast cancer cases and 162 controls. The project was reviewed and approved by the Marshfield Clinic IRB. Subjects were selected using clinical data from the Marshfield Clinic Cancer Registry and Data Warehouse. Cases were defined as women having a confirmed diagnosis of breast cancer. Both the cases and controls had to have at least one mammogram within 12 months prior to having a biopsy. The subjects also had DNA samples that were genotyped using the Illumina HumanHap660 array, as part of the eMERGE (electronic MEdical Records and Genomics) network by McCarty et al. (2011).

We apply our multiple testing procedure on the CGEMS data. The settings of the procedure are the same as in the simulations on genetic data in Section 4. The individual test is two-proportion z-test. Our procedure reports 32 SNPs with LIS value of 0.0 (an estimated probability 1.0 of being associated). We further calculate the per-allele odds-ratio of these SNPs on the Marshfield data, and 14 of them have an odds-ratio around 1.2 or above. The details about the 14 SNPs are given in supplementary material. There are two clusters among them. First, rs3870371, rs7830137 and rs920455 (on chromosome 8) locate near each other and near the gene hyaluronan synthase 2 (HAS2) which has been shown to be associated with invasive breast cancer by many studies (Udabage et al., 2005;

			t = 0.8				t = 0.5				t = 0.25			
			LIS	BH	AP	lfdr	LIS	BH	AP	lfdr	LIS	BH	AP	lfdr
Dominant	α = 0.05	FDR:	0.026	0.040	0.040	0.010	0.026	0.040	0.040	0.010	0.025	0.039	0.039	0.009
		#TP:	14	4	4	2	14	4	4	2	21	10	10	7
	α = 0.10	FDR:	0.051	0.079	0.089	0.010	0.048	0.079	0.109	0.010	0.044	0.079	0.109	0.009
		#TP:	20	12	12	3	22	12	12	3	33	19	29	18
Recessive	α = 0.05	FDR:	0.009	0.079	0.079	0.009	0.009	0.079	0.079	0.009	0.009	0.079	0.079	0.009
		#TP:	11	11	11	11	11	11	11	11	18	17	18	17
	α = 0.10	FDR:	0.018	0.104	0.104	0.009	0.018	0.104	0.114	0.009	0.017	0.104	0.114	0.009
		#TP:	11	12	12	11	11	12	12	11	22	21	21	17

(1)

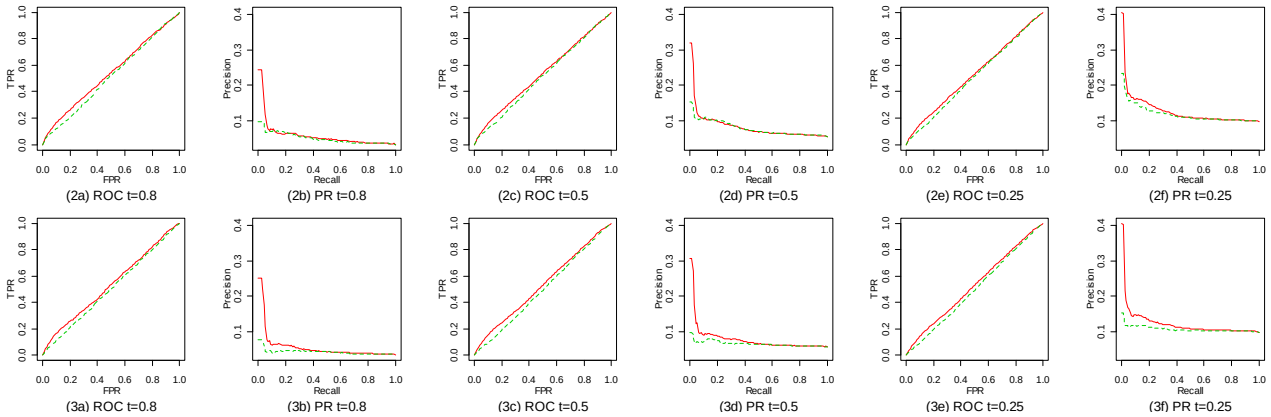


Figure 6: Comparison of BH, AP, localFDR and LIS in the dominant model and the recessive model with different t values and different nominal FDR α values. Table (1) summarizes results. Subfigures (2a)-(2f) shows ROC and PR curves of LIS (solid red lines) and individual p-values (dashed green lines) in the dominant model. Subfigures (3a)-(3f) shows ROC and PR curves of LIS and individual p-values in the recessive model.

Li et al., 2007; Bernert et al., 2011). The other cluster includes rs11200014, rs2981579, rs1219648, and rs2420946 on chromosome 10. They are exactly the 4 SNPs reported by Hunter et al. (2007). Their associated gene FGFR2 is also well known to be associated with breast cancer. SNP rs4866929 on chromosome 5 is also very likely to be associated because it is highly correlated (r^2=0.957) with SNP rs981782 (not included in our data) which was identified from a much larger dataset (4,398 cases and 4,316 controls and a follow-up confirmation stage on 21,860 cases and 22,578 controls) by Easton et al. (2007).

6 Conclusion

In this paper, we use an MRF-coupled mixture model to leverage the dependence in multiple testing problems, and show the improved numerical performance on a variety of simulations and its applicability in a real-world GWAS problem. A theoretical question of interest is whether this graphical model based procedure is optimal in the sense that it has the smallest FNR among all the valid procedures. The optimality of the oracle procedure can be proved under the compound decision framework (Sun & Cai, 2007, 2009), as long as an exact inference algorithm exists or an approximate inference algorithm can be guaranteed to converge to the correct marginal probabilities. The asymptotic optimality of the data-driven procedure (the FNR yielded by the data-driven procedure approaches the FNR yielded by the oracle procedure as the number of tests $m \to \infty$) requires consistent estimates of the unknown parameters in the graphical models. Parameter learning in undirected models is more complicated than in directed models due to the normalization constant. To the best of our knowledge, asymptotic properties of parameter learning for hidden MRFs and MRF-coupled mixture models have not been investigated. Therefore, we cannot prove the asymptotic optimality of the data-driven procedure so far, although we can observe its close-to-oracle performance in the basic simulations.

Acknowledgements

The authors acknowledge the support of the Wisconsin Genomics Initiative, NCI grant R01CA127379-01 and its ARRA supplement 3R01CA127379-03S1, NIGMS grant R01GM097618-01, NLM grant R01LM011028-01, NIEHS grant 5R01ES017400-03, eMERGE grant 1U01HG004608-01, NSF grant DMS-1106586 and the UW Carbone Cancer Center.

References

Armitage, P. (1955). Tests for linear trends in proportions and frequencies. *BIOMETRICS*, **11**, 375C386.

Asuncion, A. U., Liu, Q., Ihler, A. T., & Smyth, P. (2010). Particle filtered MCMC-MLE with connections to contrastive divergence. In *ICML*.

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *ANN MATH STAT*, **41**(1), 164–171.

Benjamini, Y. & Heller, R. (2007). False discovery rates for spatial signals. *J AM STAT ASSOC*, **102**, 1272–1281.

Benjamini, Y. & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J ROY STAT SOC B*, **57**(1), 289–300.

Benjamini, Y. & Hochberg, Y. (2000). On the adaptive control of the false discovery rate in multiple testing with independent statistics. *J EDUC BEHAV STAT*, **25**(1), 60–83.

Benjamini, Y. & Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *ANN STAT*, **29**, 1165–1188.

Bernert, B., Porsch, H., & Heldin, P. (2011). Hyaluronan synthase 2 (HAS2) promotes breast cancer cell invasion by suppression of tissue metalloproteinase inhibitor 1 (TIMP-1). *J BIOL CHEM*, **286**(49), 42349–42359.

Blanchard, G. & Roquain, E. (2009). Adaptive false discovery rate control under independence and dependence. *J MACH LEARN RES*, **10**, 2837–2871.

Celeux, G., Forbes, F., & Peyrard, N. (2003). EM procedures using mean field-like approximations for Markov model-based image segmentation. *Pattern Recognition*, **36**, 131–144.

Chatzis, S. P. & Varvarigou, T. A. (2008). A fuzzy clustering approach toward hidden Markov random field models for enhanced spatially constrained image segmentation. *IEEE Transactions on Fuzzy Systems*, **16**, 1351 – 1361.

Cochran, W. G. (1954). Some methods for strengthening the common chi-square tests. *BIOMETRICS*, **10**, 417–451.

Crouse, M. S., Nowak, R. D., & Baraniuk, R. G. (1998). Wavelet-based statistical signal processing using hidden Markov models. *IEEE T SIGNAL PROCES*, **46**(4), 886–902.

Easton, D. F., Pooley, K. A., Dunning, A. M., Pharoah, P. D. P., Thompson, D., Ballinger, D. G., Struewing, J. P., Morrison, J., Field, H., Luben, R., Wareham, N., Ahmed, S., Healey, C. S., Bowman, R., Meyer, K. B., Haiman, C. A., Kolonel, L. K., Henderson, B. E., Le Marchand, L., Brennan, P., Sangrajrang, S., Gaborieau, V., Odefrey, F., Shen, C.-Y., Wu, P.-E., Wang, H.-C., Eccles, D., Evans, G. D., Peto, J., Fletcher, O., Johnson, N., Seal, S., Stratton, M. R., Rahman, N., Chenevix-Trench, G., Bojesen, S. E., Nordestgaard, B. G., Axelsson, C. K., Garcia-Closas, M., Brinton, L., Chanock, S., Lissowska, J., Peplonska, B., Nevanlinna, H., Fagerholm, R., Eerola, H., Kang, D., Yoo, K.-Y., Noh, D.-Y., Ahn, S.-H., Hunter, D. J., Hankinson, S. E., Cox, D. G., Hall, P., Wedren, S., Liu, J., Low, Y.-L., Bogdanova, N., Schürmann, P., Dörk, T., Tollenaar, R. A. E. M., Jacobi, C. E., Devilee, P., Klijn, J. G. M., Sigurdson, A. J., Doody, M. M., Alexander, B. H., Zhang, J., Cox, A., Brock, I. W., Macpherson, G., Reed, M. W. R., Couch, F. J., Goode, E. L., Olson, J. E., Meijers-Heijboer, H., van den Ouweland, A., Uitterlinden, A., Rivadeneira, F., Milne, R. L., Ribas, G., Gonzalez-Neira, A., Benitez, J., Hopper, J. L., Mccredie, M., Southey, M., Giles, G. G., Schroen, C., Justenhoven, C., Brauch, H., Hamann, U., Ko, Y.-D., Spurdle, A. B., Beesley, J., Chen, X., Mannermaa, A., Kosma, V.-M., Kataja, V., Hartikainen, J., Day, N. E., Cox, D. R., & Ponder, B. A. J. (2007). Genome-wide association study identifies novel breast cancer susceptibility loci. *Nature*, **447**, 1087–1093.

Efron, B. (2007). Correlation and large-scale simultaneous significance testing. *J AM STAT ASSOC*, **102**(477), 93–103.

Efron, B., Tibshirani, R., Storey, J. D., & Tusher, V. (2001). Empirical Bayes analysis of a microarray experiment. *J AM STAT ASSOC*, **96**, 1151–1160.

Fan, J., Han, X., & Gu, W. (2012). Control of the false discovery rate under arbitrary covariance dependence. (to appear) J AM STAT ASSOC.

Farcomeni, A. (2007). Some results on the control of the false discovery rate under dependence. *SCAND J STAT*, **34**(2), 275–297.

Finner, H. & Roters, M. (2002). Multiple hypotheses testing and expected number of type I errors. *ANN STAT*, **30**, 220–238.

Freidlin, B., Zheng, G., Li, Z., & Gastwirth, J. L. (2002). Trend tests for case-control studies of genetic markers: power, sample size and robustness. *HUM HERED*, **53**(3), 146–152.

Friguet, C., Kloareg, M., & Causeur, D. (2009). A factor model approach to multiple testing under dependence. *J AM STAT ASSOC*, **104**(488), 1406–1415.

Ganapathi, V., Vickrey, D., Duchi, J., & Koller, D.

(2008). Constrained approximate maximum entropy learning of Markov random fields. In *UAI*.

Gelfand, A. E. & Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *J AM STAT ASSOC*, **85**(410), 398–409.

Genovese, C. & Wasserman, L. (2002). Operating characteristics and extensions of the false discovery rate procedure. *J ROY STAT SOC B*, **64**, 499–517.

Genovese, C. & Wasserman, L. (2004). A stochastic process approach to false discovery control. *ANN STAT*, **32**, 1035–1061.

Genovese, C., Roeder, K., & Wasserman, L. (2006). False discovery control with p-value weighting. *BIOMETRIKA*, **93**, 509–524.

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. *COMP SCI STAT*, pages 156–163.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *NEURAL COMPUT*, **14**, 1771–1800.

Hunter, D. J., Kraft, P., Jacobs, K. B., Cox, D. G., Yeager, M., Hankinson, S. E., Wacholder, S., Wang, Z., Welch, R., Hutchinson, A., Wang, J., Yu, K., Chatterjee, N., Orr, N., Willett, W. C., Colditz, G. A., Ziegler, R. G., Berg, C. D., Buys, S. S., Mccarty, C. A., Feigelson, H. S., Calle, E. E., Thun, M. J., Hayes, R. B., Tucker, M., Gerhard, D. S., Fraumeni, J. F., Hoover, R. N., Thomas, G., & Chanock, S. J. (2007). A genome-wide association study identifies alleles in FGFR2 associated with risk of sporadic postmenopausal breast cancer. *NAT GENET*, **39**(7), 870–874.

Jordan, M. I., Ghahramani, Z., Jaakkola, T., & Saul, L. K. (1999). An introduction to variational methods for graphical models. *MACH LEARN*, **37**, 183–233.

Kschischang, F., Frey, B., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE T INFORM THEORY*, **47**(2), 498 –519.

Lauritzen, S. L. & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J ROY STAT SOC B*, **50**(2), 157–224.

Leek, J. T. & Storey, J. D. (2008). A general framework for multiple testing dependence. *P NATL ACAD SCI USA*, **105**(48), 18718–18723.

Li, Y., Li, L., Brown, T. J., & Heldin, P. (2007). Silencing of hyaluronan synthase 2 suppresses the malignant phenotype of invasive breast cancer cells. *INT J CANCER*, **120**(12), 2557–2567.

McCarty, C., Wilke, R., Giampietro, P., Wesbrook, S., & Caldwell, M. (2005). Marshfield Clinic Personalized Medicine Research Project (PMRP): design, methods and recruitment for a large population-based biobank. *PERS MED*, **2**, 49–79.

McCarty, C. A., Chisholm, R. L., Chute, C. G., Kullo, I. J., Jarvik, G. P., Larson, E. B., Li, R., Masys, D. R., Ritchie, M. D., Roden, D. M., Struewing, J. P., Wolf, W. A., & eMERGE Team (2011). The eMERGE Network: a consortium of biorepositories linked to electronic medical records data for conducting genomic studies. *BMC MED GENET*, **4**(1), 13.

Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *UAI*, pages 467–475.

Owen, A. B. (2005). Variance of the number of false discoveries. *J ROY STAT SOC B*, **67**, 411–426.

Romano, J., Shaikh, A., & Wolf, M. (2008). Control of the false discovery rate under dependence using the bootstrap and subsampling. *TEST*, **17**, 417–442.

Salakhutdinov, R. (2009). Learning in Markov random fields using tempered transitions. In *NIPS*, pages 1598–1606.

Sarkar, S. K. (2006). False discovery and false nondiscovery rates in single-step multiple testing procedures. *ANN STAT*, **34**(1), 394–415.

Schraudolph, N. N. (2010). Polynomial-time exact inference in NP-hard binary MRFs via reweighted perfect matching. In *AISTATS*.

Schraudolph, N. N. & Kamenetsky, D. (2009). Efficient exact inference in planar Ising models. In *NIPS*.

Slager, S. L. & Schaid, D. J. (2001). Case-control studies of genetic markers: power and sample size approximations for Armitage's test for trend. *HUM HERED*, **52**(3), 149–153.

Storey, J. D. (2002). A direct approach to false discovery rates. *J ROY STAT SOC B*, **64**, 479–498.

Storey, J. D. (2003). The positive false discovery rate: A Bayesian interpretation and the q-value. *ANN STAT*, **31**(6), 2013–2035.

Su, Z., Marchini, J., & Donnelly, P. (2011). HAPGEN2: simulation of multiple disease SNPs. *BIOINFORMATICS*.

Sun, W. & Cai, T. T. (2007). Oracle and adaptive compound decision rules for false discovery rate control. *J AM STAT ASSOC*, **102**(479), 901–912.

Sun, W. & Cai, T. T. (2009). Large-scale multiple testing under dependence. *J ROY STAT SOC B*, **71**, 393–424.

The International HapMap Consortium (2003). The international HapMap project. *NATURE*, **426**, 789–796.

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In *ICML*, pages 1064–1071.

Tieleman, T. & Hinton, G. (2009). Using fast weights to improve persistent contrastive divergence. In *ICML*, pages 1033–1040.

Udabage, L., Brownlee, G. R., Nilsson, S. K., & Brown, T. J. (2005). The over-expression of HAS2, Hyal-2 and CD44 is implicated in the invasiveness of breast cancer. *EXP CELL RES*, **310**(1), 205 – 217.

Wainwright, M. J., Jaakkola, T. S., & Willsky, A. S. (2003a). Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE T INFORM THEORY*, **49**, 2003.

Wainwright, M. J., Jaakkola, T. S., & Willsky, A. S. (2003b). Tree-reweighted belief propagation algorithms and approximate ML estimation via pseudo-moment matching. In *AISTATS*.

Weiss, Y. (2000). Correctness of local probability propagation in graphical models with loops. *NEURAL COMPUT*, **12**(1), 1–41.

Welling, M. & Sutton, C. (2005). Learning in Markov random fields with contrastive free energies. In *AISTATS*.

Wu, M. C., Kraft, P., Epstein, M. P., Taylor, D. M., Chanock, S. J., Hunter, D. J., & Lin, X. (2010). Powerful SNP-set analysis for case-control genome-wide association studies. *AM J HUM GENET*, **86**(6), 929–942.

Wu, W. B. (2008). On false discovery control under dependence. *ANN STAT*, **36**(1), 364–380.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2000). Generalized belief propagation. In *NIPS*, pages 689–695. MIT Press.

Yekutieli, D. & Benjamini, Y. (1999). Resampling-based false discovery rate controlling multiple test procedures for correlated test statistics. *J STAT PLAN INFER*, **82**, 171–196.

Zhang, C., Fan, J., & Yu, T. (2011). Multiple testing via FDR_L for large-scale imaging data. *ANN STAT*, **39**(1), 613–642.

Zhang, Y., Brady, M., & Smith, S. (2001). Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. *IEEE Transactions on Medical Imaging*.

Belief Propagation for Structured Decision Making

Qiang Liu
Department of Computer Science
University of California, Irvine
Irvine, CA, 92697
qliu1@ics.uci.edu

Alexander Ihler
Department of Computer Science
University of California, Irvine
Irvine, CA, 92697
ihler@ics.uci.edu

Abstract

Variational inference algorithms such as belief propagation have had tremendous impact on our ability to learn and use graphical models, and give many insights for developing or understanding exact and approximate inference. However, variational approaches have not been widely adoped for *decision making* in graphical models, often formulated through influence diagrams and including both centralized and decentralized (or multi-agent) decisions. In this work, we present a general variational framework for solving structured cooperative decision-making problems, use it to propose several belief propagation-like algorithms, and analyze them both theoretically and empirically.

1 Introduction

Graphical modeling approaches, including Bayesian networks and Markov random fields, have been widely adopted for problems with complicated dependency structures and uncertainties. The problems of *learning*, i.e., estimating a model from data, and *inference*, e.g., calculating marginal probabilities or maximum *a posteriori* (MAP) estimates, have attracted wide attention and are well explored. Variational inference approaches have been widely adopted as a principled way to develop and understand many exact and approximate algorithms. On the other hand, the problem of *decision making* in graphical models, sometimes formulated via influence diagrams or decision networks and including both sequential centralized decisions and decentralized or multi-agent decisions, is surprisingly less explored in the approximate inference community.

Influence diagrams (ID), or *decision networks*, [Howard and Matheson, 1985, 2005] are a graphical model representation of structured decision problems under uncertainty; they can be treated as an extension of Bayesian networks, augmented with decision nodes and utility functions. Traditionally, IDs are used to model centralized, sequential decision processes under "perfect recall", which assumes that the decision steps are ordered in time and that all information is remembered across time; limited memory influence diagrams (LIMIDs) [Zhang et al., 1994, Lauritzen and Nilsson, 2001] relax the perfect recall assumption, creating a natural framework for representing decentralized and information-limited decision problems, such as team decision making and multi-agent systems. Despite the close connection and similarity to Bayes nets, IDs have less visibility in the graphical model and automated reseasoning community, both in terms of modeling and algorithm development; see Pearl [2005] for an interesting historical perspective.

Solving an ID refers to finding decision rules that maximize the expected utility function (MEU); this task is significantly more difficult than standard inference on a Bayes net. For IDs with perfect recall, MEU can be restated as a dynamic program, and solved with cost exponential in a constrained tree-width of the graph that is subject to the temporal ordering of the decision nodes. The constrained tree-width can be much higher than the tree-width associated with typical inference, making MEU significantly more complex. For LIMIDs, non-convexity issues also arise, since the limited shared information and simultaneous decisions may create locally optimal policies. The most popular algorithm for LIMIDs is based on policy-by-policy improvement [Lauritzen and Nilsson, 2001], and provides only a "person-by-person" notion of optimality. Surprisingly, the variational ideas that revolutionized inference in Bayes nets have not been adopted for influence diagrams. Although there exists work on transforming MEU problems into sequences of standard marginalization problems [e.g., Zhang, 1998], on which variational methods apply, these methods do not yield general frameworks, and usually only work for IDs with perfect recall. A full variational frame-

work would provide general procedures for developing efficient approximations such as loopy belief propagation (BP), that are crucial for large scale problems, or providing new theoretical analysis.

In this work, we propose a general variational framework for solving influence diagrams, both with and without perfect recall. Our results on centralized decision making include traditional inference in graphical models as special cases. We propose a spectrum of exact and approximate algorithms for MEU problems based on the variational framework. We give several optimality guarantees, showing that under certain conditions, our BP algorithm can find the globally optimal solution for ID with perfect recall and solve LIMIDs in a stronger locally optimal sense than coordinate-wise optimality. We show that a temperature parameter can also be introduced to smooth between MEU tasks and standard (easier) marginalization problems, and can provide good solutions by annealing the temperature or using iterative proximal updates.

This paper is organized as follows. Section 2 sets up background on graphical models, variational methods and influence diagrams. We present our variational framework of MEU in Section 3, and use it to develop several BP algorithms in Section 4. We present numerical experiments in Section 5. Finally, we discuss additional related work in Section 6 and concluding remarks in Section 7. Proofs and additional information can be found in the appendix.

2 Background

2.1 Graphical Models

Let $\boldsymbol{x} = \{x_1, x_2, \cdots, x_n\}$ be a random vector in $\mathbb{X} = \mathbb{X}_1 \times \cdots \times \mathbb{X}_n$. Consider a factorized probability on $\boldsymbol{x}$,

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(x_\alpha) \;=\; \frac{1}{Z} \exp\Big[\sum_{\alpha \in \mathcal{I}} \theta_\alpha(x_\alpha) \Big],$$

where $\mathcal{I}$ is a set of variable subsets, and $\psi_\alpha : \mathbb{X}_\alpha \to \mathbb{R}^+$ are positive factors; the $\theta_\alpha(x_\alpha) = \log \psi_\alpha(x_\alpha)$ are the natural parameters of the exponential family representation; and $Z = \sum_{\boldsymbol{x}} \prod_{\alpha \in \mathcal{I}} \psi_\alpha$ is the normalization constant or partition function with $\Phi(\boldsymbol{\theta}) = \log Z$ the log-partition function. Let $\boldsymbol{\theta} = \{\theta_\alpha | \alpha \in \mathcal{I}\}$ and $\theta(\boldsymbol{x}) = \sum_\alpha \theta_\alpha(x_\alpha)$. There are several ways to represent a factorized distribution using graphs (i.e., *graphical models*), including Markov random fields, Bayesian networks, factors graphs and others.

Given a graphical model, *inference* refers to the procedure of answering probabilistic queries. Important inference tasks include marginalization, maximum *a posteriori* (MAP, sometimes called maximum probability of evidence or MPE), and marginal MAP (sometimes simply MAP). All these are NP-hard in general. Marginalization calculates the marginal probabilities of one or a few variables, or equivalently the normalization constant Z, while MAP/MPE finds the mode of the distribution. More generally, marginal MAP seeks the mode of a marginal probability,

$$\text{Marginal MAP:} \quad \boldsymbol{x}^* = \arg\max_{x_A} \sum_{x_B} \prod_{\alpha} \psi_\alpha(x_\alpha),$$

where A, B are disjoint sets with $A \cup B = V$; it reduces to marginalization if $A = \emptyset$ and to MAP if $B = \emptyset$.

Marginal Polytope. A marginal polytope $\mathbb{M}$ is a set of local marginals $\boldsymbol{\tau} = \{\tau_\alpha(x_\alpha) \colon \alpha \in \mathcal{I}\}$ that are extensible to a global distribution over $\boldsymbol{x}$, that is, $\mathbb{M} = \{\boldsymbol{\tau} | \; \exists \text{ a distribution } p(\boldsymbol{x}), \text{ s.t. } \sum_{x_{V \setminus \alpha}} p(\boldsymbol{x}) = \tau_\alpha(x_\alpha) \; \}$. Call $\mathcal{P}[\boldsymbol{\tau}]$ the set of global distributions consistent with $\boldsymbol{\tau} \in \mathbb{M}$; there exists a unique distribution in $\mathcal{P}[\boldsymbol{\tau}]$ that has maximum entropy and follows the exponential family form for some $\boldsymbol{\theta}$. We abuse notation to denote this unique global distribution $\tau(\boldsymbol{x})$.

A basic result for variational methods is that $\Phi(\boldsymbol{\theta})$ is convex and can be rewritten into a dual form,

$$\Phi(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) \}, \tag{1}$$

where $\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle = \sum_{\boldsymbol{x}} \sum_\alpha \theta_\alpha(x_\alpha) \tau_\alpha(x_\alpha)$ is the point-wise inner product, and $H(\boldsymbol{x}; \boldsymbol{\tau}) = -\sum_{\boldsymbol{x}} \tau(\boldsymbol{x}) \log \tau(\boldsymbol{x})$ is the entropy of distribution $\tau(\boldsymbol{x})$; the maximum of (1) is obtained when $\boldsymbol{\tau}$ equals the marginals of the original distribution with parameter $\boldsymbol{\theta}$. See Wainwright and Jordan [2008].

Similar dual forms hold for MAP and marginal MAP. Letting $\Phi_{A,B}(\boldsymbol{\theta}) = \log \max_{x_A} \sum_{x_B} \exp(\theta(\boldsymbol{x}))$, we have [Liu and Ihler, 2011]

$$\Phi_{A,B}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(x_B | x_A \,; \boldsymbol{\tau}) \}, \tag{2}$$

where $H(x_B | x_A \,; \boldsymbol{\tau}) = -\sum_{\boldsymbol{x}} \tau(\boldsymbol{x}) \log \tau(x_B | x_A)$ is the conditional entropy; its appearance corresponds to the sum operators.

The dual forms in (1) and (2) are no easier to compute than the original inference. However, one can approximate the marginal polytope $\mathbb{M}$ and the entropy in various ways, yielding a body of approximate inference algorithms, such as loopy belief propagation (BP) and its generalizations [Yedidia et al., 2005, Wainwright et al., 2005], linear programming solvers [e.g., Wainwright et al., 2003b], and recently hybrid message passing algorithms [Liu and Ihler, 2011, Jiang et al., 2011].

Junction Graph BP. Junction graphs provide a procedural framework to approximate the dual (1). A cluster graph is a triple $(\mathcal{G}, \mathcal{C}, \mathcal{S})$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph, with each node $k \in \mathcal{V}$ associated

with a subset of variables $c_k \in \mathcal{C}$ (clusters), and each edge $(kl) \in \mathcal{E}$ a subset $s_{kl} \in \mathcal{S}$ (separator) satisfying $s_{kl} \subseteq c_k \cap c_l$. We assume that $\mathcal{C}$ subsumes the index set $\mathcal{I}$, that is, for any $\alpha \in \mathcal{I}$, there exists a $c_k \in \mathcal{C}$, denoted $c[\alpha]$, such that $\alpha \subseteq c_k$. In this case, we can reparameterize $\boldsymbol{\theta} = \{\theta_\alpha | \alpha \in \mathcal{I}\}$ into $\boldsymbol{\theta} = \{\theta_{c_k} | k \in \mathcal{V}\}$ by taking $\theta_{c_k} = \sum_{\alpha:\, c[\alpha]=c_k} \theta_\alpha$, without changing the distribution. A cluster graph is called a *junction graph* if it satisfies the *running intersection property* – for each $i \in V$, the induced sub-graph consisting of the clusters and separators that include i is a connected tree. A junction graph is a junction tree if $\mathcal{G}$ is tree.

To approximate the dual (1), we can replace $\mathbb{M}$ with a locally consistent polytope $\mathbb{L}$: the set of local marginals $\boldsymbol{\tau} = \{\tau_{c_k}, \tau_{s_{kl}} : k \in \mathcal{V}, (kl) \in \mathcal{E}\}$ satisfying $\sum_{x_{c_k \setminus s_{kl}}} \tau_{c_k}(x_{c_k}) = \tau(x_{s_{kl}})$. Clearly, $\mathbb{M} \subseteq \mathbb{L}$. We then approximate (1) by

$$\max_{\boldsymbol{\tau} \in \mathbb{L}} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{k \in \mathcal{V}} H(x_{c_k}; \boldsymbol{\tau}_{c_k}) - \sum_{(kl) \in \mathcal{E}} H(x_{s_{kl}}; \boldsymbol{\tau}_{s_{kl}})\},$$

where the joint entropy is approximated by a linear combination of the entropies of local marginals. The approximate objective can be solved using Lagrange multipliers [Yedidia et al., 2005], leading to a sum-product message passing algorithm that iteratively sends messages between neighboring clusters via

$$m_{k \to l}(x_{c_l}) \propto \sum_{x_{c_k \setminus s_{kl}}} \psi_{c_k}(x_{c_k}) m_{\sim k \setminus l}(x_{c_{\mathcal{N}(k)}}), \qquad (3)$$

where $\psi_{c_k} = \exp(\theta_{c_k})$, and $m_{\sim k \setminus l}$ is the product of messages into k from its neighbors $\mathcal{N}(k)$ except l. At convergence, the (locally) optimal marginals are

$$\tau_{c_k} \propto \psi_{c_k} m_{\sim k} \qquad \text{and} \qquad \tau_{s_{kl}} \propto m_{k \to l} m_{l \to k},$$

where $m_{\sim k}$ is the product of messages into k. Max-product and hybrid methods can be derived analogously for MAP and marginal MAP problems.

2.2 Influence Diagrams

Influence diagrams (IDs) or *decision networks* are extensions of Bayesian networks to represent structured decision problems under uncertainty. Formally, an influence diagram is defined on a directed acyclic graph $G = (V, E)$, where the nodes V are divided into two subsets, $V = R \cup D$, where R and D represent respectively the set of chance nodes and decision nodes. Each chance node $i \in R$ represents a random variable x_i with a conditional probability table $p_i(x_i | x_{\mathrm{pa}(i)})$. Each decision node $i \in D$ represents a controllable decision variable x_i, whose value is determined by a decision maker via a decision rule (or policy) $\delta_i : \mathbb{X}_{\mathrm{pa}(i)} \to \mathbb{X}_i$, which determines the values of x_i based on the observation on the values of $x_{\mathrm{pa}(i)}$; we call the collection

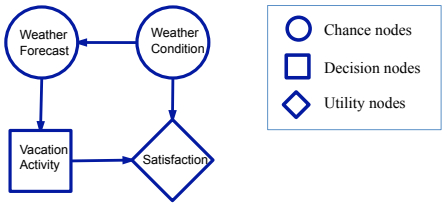


Figure 1: A simple influence diagram for deciding vacation activity [Shachter, 2007].

of policies $\boldsymbol{\delta} = \{\delta_i | i \in D\}$ a *strategy*. Finally, a utility function $u : \mathbb{X} \to \mathbb{R}^+$ measures the reward given an instantiation of $\boldsymbol{x} = [x_R, x_D]$, which the decision maker wants to maximize. It is reasonable to assume some decomposition structure on the utility $u(\boldsymbol{x})$, either additive, $u(\boldsymbol{x}) = \sum_{j \in U} u_j(x_{\beta_j})$, or multiplicative, $u(\boldsymbol{x}) = \prod_{j \in U} u_j(x_{\beta_j})$. A decomposable utility function can be visualized by augmenting the DAG with a set of leaf nodes U, called *utility nodes*, each with parent set β_j. See Fig. 1 for a simple example.

A decision rule δ_i is alternatively represented as a deterministic conditional "probability" $p_i^\delta(x_i | x_{\mathrm{pa}(i)})$, where $p_i^\delta(x_i | x_{\mathrm{pa}(i)}) = 1$ for $x_i = \delta_i(x_{\mathrm{pa}(i)})$ and zero otherwise. It is helpful to allow *soft decision rules* where $p_i^\delta(x_i | x_{\mathrm{pa}(i)})$ takes fractional values; these define a *randomized strategy* in which x_i is determined by randomly drawing from $p_i^\delta(x_i | x_{\mathrm{pa}(i)})$. We denote by Δ^o the set of deterministic strategies and Δ the set of randomized strategies. Note that Δ^o is a discrete set, while Δ is its convex hull.

Given an influence diagram, the optimal strategy should maximize the *expected utility function* (MEU):

$$\begin{aligned} \mathrm{MEU} &= \max_{\boldsymbol{\delta} \in \Delta} \mathrm{EU}(\boldsymbol{\delta}) = \max_{\boldsymbol{\delta} \in \Delta} \mathbb{E}(u(\boldsymbol{x}) | \boldsymbol{\delta}) \\ &= \max_{\boldsymbol{\delta} \in \Delta} \sum_{\boldsymbol{x}} u(\boldsymbol{x}) \prod_{i \in C} p_i(x_i | x_{\mathrm{pa}(i)}) \prod_{i \in D} p_i^\delta(x_i | x_{\mathrm{pa}(i)}) \\ &\overset{def}{=} \max_{\boldsymbol{\delta} \in \Delta} \sum_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x})) \prod_{i \in D} p_i^\delta(x_i | x_{\mathrm{pa}(i)}) \qquad (4) \end{aligned}$$

where $\theta(\boldsymbol{x}) = \log[u(\boldsymbol{x}) \prod_{i \in C} p_i(x_i | x_{\mathrm{pa}(i)})]$; we call the distribution $q(\boldsymbol{x}) \propto \exp(\theta(\boldsymbol{x}))$ the *augmented distribution* [Bielza et al., 1999]. The concept of the augmented distribution is critical since it completely specifies a MEU problem without the semantics of the influence diagram; hence one can specify $q(\boldsymbol{x})$ arbitrarily, e.g., via an undirected MRF, extending the definition of IDs. We can treat MEU as a special sort of "inference" on the augmented distribution, which as we will show, generalizes more common inference tasks.

In (4) we maximize the expected utility over Δ; this is equivalent to maximizing over Δ^o, since

Lemma 2.1. *For any ID,* $\max_{\boldsymbol{\delta} \in \Delta} EU(\boldsymbol{\delta}) = \max_{\boldsymbol{\delta} \in \Delta^o} EU(\boldsymbol{\delta})$.

Perfect Recall Assumption. The MEU problem can be solved in closed form if the influence diagram satisfies a *perfect recall assumption* (PRA) — there exists a "temporal" ordering over all the decision nodes, say $\{d_1, d_2, \cdots, d_m\}$, consistent with the partial order defined by the DAG G, such that every decision node observes all the earlier decision nodes and their parents, that is, $\{d_j\} \cup \mathrm{pa}(d_j) \subseteq \mathrm{pa}(d_i)$ for any $j < i$. Intuitively, PRA implies a centralized decision scenario, where a global decision maker sets all the decision nodes in a predefined order, with perfect memory of all the past observations and decisions.

With PRA, the chance nodes can be grouped by when they are observed. Let r_{i-1} $(i = 1, \ldots, m)$ be the set of chance nodes that are parents of d_i but not of any d_j for $j < i$; then both decision and chance nodes are ordered by $o = \{r_0, d_1, r_1, \cdots, d_m, r_m\}$. The MEU and its optimal strategy for IDs with PRA can be calculated by a sequential sum-max-sum rule,

$$\mathrm{MEU} = \sum_{x_{r_0}} \max_{x_{d_1}} \sum_{x_{r_1}} \cdots \max_{x_{d_m}} \sum_{x_{r_m}} \exp(\theta(\boldsymbol{x})), \qquad (5)$$

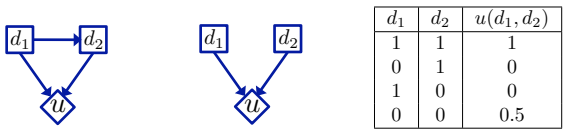
$$\delta^*_{d_i}(x_{\mathrm{pa}(d_i)}) = \arg\max_{x_{d_i}} \Big\{ \sum_{x_{r_i}} \cdots \max_{x_{d_m}} \sum_{x_{r_m}} \exp(\theta(\boldsymbol{x})) \Big\},$$

where the calculation is performed in reverse temporal ordering, interleaving marginalizing chance nodes and maximizing decision nodes. Eq. (5) generalizes the inference tasks in Section 2.1, arbitrarily interleaving the sum and max operators. For example, marginal MAP can be treated as a *blind decision problem*, where no chance nodes are observed by any decision nodes.

As in other inference tasks, the calculation of the sum-max-sum rule can be organized into local computations if the augmented distribution $q(\boldsymbol{x})$ is factorized. However, since the max and sum operators are not exchangeable, the calculation of (5) is restricted to elimination orders consistent with the "temporal ordering". Notoriously, this "constrained" tree-width can be very high even for trees. See Koller and Friedman [2009].

However, PRA is often unrealistic. First, most systems lack enough memory to express arbitrary policies over an entire history of observations. Second, many practical scenarios, like team decision analysis [Detwarasiti and Shachter, 2005] and decentralized sensor networks [Kreidl and Willsky, 2006], are distributed by nature: a team of agents makes decisions independently based on sharing limited information with their neighbors. In these cases, relaxing PRA is very important.

Imperfect Recall. General IDs with the perfect recall assumption relaxed are discussed in Zhang et al. [1994], Lauritzen and Nilsson [2001], Koller and Milch [2003], and are commonly referred as limited memory influence diagrams (LIMIDs). Unfortunately, the relaxation causes many difficulties. First, it is no longer possible to eliminate the decision nodes in a sequential "sum-max-sum" fashion. Instead, the dependencies of the decision nodes have cycles, formally discussed in Koller and Milch [2003] by defining a relevance graph over the decision nodes; the relevance graph is a tree with PRA, but is usually loopy with imperfect recall. Thus iterative algorithms are usually required for LIMIDs. Second and more importantly, the incomplete information may cause the agents to behave myopically, selfishly choosing locally optimal strategies due to ignorance of the global statistics. This breaks the strategy space into many local modes, making the MEU problem non-convex; see Fig. 2 for an illustration.



(a) Perfect Recall (b) Imperfect Recall

d_1	d_2	$u(d_1, d_2)$
1	1	1
0	1	0
1	0	0
0	0	0.5

(c) Utility Function

Figure 2: Illustrating imperfect recall. In (a) d_2 observes d_1; its optimal decision rule is to equal d_1's state (whatever it is); knowing d_2 will follow, d_1 can choose $d_1 = 1$ to achieve the global optimum. In (b) d_1 and d_2 do not know the other's states; both $d_1 = d_2 = 1$ and $d_1 = d_2 = 0$ (suboptimal) become locally optimal strategies and the problem is multi-modal.

The most popular algorithms for LIMIDs are based on policy-by-policy improvement, e.g., the *single policy update* (SPU) algorithm [Lauritzen and Nilsson, 2001] sequentially optimizes δ_i with $\boldsymbol{\delta}_{\neg i} = \{\delta_j : j \neq i\}$ fixed:

$$\delta_i(x_{\mathrm{pa}(i)}) \leftarrow \arg\max_{x_i} \mathbb{E}(u(\boldsymbol{x})|x_{\mathrm{fam}(i)}\,; \boldsymbol{\delta}_{\neg i}), \qquad (6)$$

$$\mathbb{E}(u(\boldsymbol{x})|x_{\mathrm{fam}(i)}; \boldsymbol{\delta}_{\neg i}) = \sum_{x_{\neg \mathrm{fam}(i)}} \exp(\theta(\boldsymbol{x})) \prod_{j \in D \setminus \{i\}} p^{\delta}_j(x_j|x_{\mathrm{pa}(j)}),$$

where $\mathrm{fam}(i) = \{i\} \cup \mathrm{pa}(i)$. The update circles through all $i \in D$ in some order, and ties are broken arbitrarily in case of multiple maxima. The expected utility in SPU is non-decreasing at each iteration, and it gives a locally optimal strategy at convergence in the sense that the expected utility can not be improved by changing any single node's policy. Unfortunately, SPU's solution is heavily influenced by initialization and can be very suboptimal.

This issue is helped by generalizing SPU to the strategy improvement (SI) algorithm [Detwarasiti and Shachter, 2005], which simultaneously updates subgroups of decisions nodes. However, the time and space complexity of SI grows exponentially with the sizes of subgroups. In the sequel, we present a novel variational framework for MEU, and propose BP-like algorithms that go beyond the naïve greedy paradigm.

3 Duality Form of MEU

In this section, we derive a duality form for MEU, generalizing the duality results of the standard inference in Section 2.1. Our main result is summarized in the following theorem.

Theorem 3.1. *(a). For an influence diagram with augmented distribution $q(\boldsymbol{x}) \propto \exp(\theta(\boldsymbol{x}))$, its log maximum expected utility* $\log \mathrm{MEU}(\boldsymbol{\theta})$ *equals*

$$\max_{\boldsymbol{\tau}\in\mathbb{M}}\{\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle + H(\boldsymbol{x};\boldsymbol{\tau}) - \sum_{i\in D} H(x_i|x_{\mathrm{pa}(i)};\boldsymbol{\tau})\}. \quad (7)$$

Suppose $\boldsymbol{\tau}^*$ is a maximum of (7), then $\boldsymbol{\delta}^* = \{\boldsymbol{\tau}^*(x_i|x_{\mathrm{pa}(i)})|i\in D\}$ is an optimal strategy.

(b). For IDs with perfect recall, (7) reduces to

$$\max_{\boldsymbol{\tau}\in\mathbb{M}}\{\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle + \sum_{o_i\in C} H(x_{o_i}|x_{o_{1:i-1}};\boldsymbol{\tau})\}, \quad (8)$$

where o is the temporal ordering of the perfect recall.

Proof. (a) See appendix; (b) note PRA implies $\mathrm{pa}(i) = o_{1:i-1}$ $(i\in D)$, and apply the entropy chain rule. □

The distinction between (8) and (7) is subtle but important: although (8) (with perfect recall) is always (if not strictly) a convex optimization, (7) (without perfect recall) may be non-convex if the subtracted entropy terms overwhelm; this matches the intuition that incomplete information sharing gives rise to multiple locally optimal strategies.

The MEU duality (8) for ID with PRA generalizes earlier duality results of inference: with no decision nodes, $D = \emptyset$ and (8) reduces to (1) for the log-partition function; when $C = \emptyset$, no entropy terms appear and (8) reduces to the linear program relaxation of MAP. Also, (8) reduces to marginal MAP when no chance nodes are observed before any decision. As we show in Section 4, this unification suggests a line of unified algorithms for all these different inference tasks.

Several corollaries provide additional insights.

Corollary 3.2. *For an ID with parameter $\boldsymbol{\theta}$, we have*

$$\log \mathrm{MEU} = \max_{\boldsymbol{\tau}\in\mathbb{I}}\{\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle + \sum_{o_i\in C} H(x_{o_i}|x_{o_{1:i-1}};\boldsymbol{\tau})\} \quad (9)$$

where $\mathbb{I} = \{\boldsymbol{\tau}\in\mathbb{M}\colon x_{o_i} \perp x_{o_{1:i-1}\setminus\mathrm{pa}(o_i)}|x_{\mathrm{pa}(o_i)}, \forall o_i \in D\}$, corresponding to those distributions that respect the imperfect recall constraints; "$x \perp y\,|\,z$" denotes conditional independence of x and y given z.

Corollary 3.2 gives another intuitive interpretation of imperfect recall vs. perfect recall: MEU with imperfect recall optimizes same objective function, but over a subset of the marginal polytope that restricts the observation domains of the decision rules; this non-convex inner subset is similar to the mean field approximation for partition functions. See Wolpert [2006] for a similar connection to mean field for bounded rational game theory. Interestingly, this shows that extending a LIMID to have perfect recall (by extending the observation domains of the decision nodes) can be considered a "convex" relaxation of the LIMID.

Corollary 3.3. *For any ϵ, if $\boldsymbol{\tau}^*$ is global optimum of*

$$\max_{\boldsymbol{\tau}\in\mathbb{M}}\{\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle + H(\boldsymbol{x}) - (1-\epsilon)\sum_{i\in D} H(x_i|x_{\mathrm{pa}(i)})\}. \quad (10)$$

and $\boldsymbol{\delta}^* = \{\tau^*(x_i|x_{\mathrm{pa}(i)})|i\in D\}$ is a deterministic strategy, then it is an optimal strategy for MEU.

The parameter ϵ is a temperature to "anneal" the MEU problem, and trades off convexity and optimality. For large ϵ, e.g., $\epsilon \geq 1$, the objective in (10) is a strictly convex function, while $\boldsymbol{\delta}^*$ is unlikely to be deterministic nor optimal (if $\epsilon = 1$, (10) reduces to standard marginazation); as ϵ decreases towards zero, $\boldsymbol{\delta}^*$ becomes more deterministic, but (10) becomes more non-convex and is harder to solve. In Section 4 we derive several possible optimization approaches.

4 Algorithms

The duality results in Section 3 offer new perspectives for MEU, allowing us to bring the tools of variational inference to develop new efficient algorithms. In this section, we present a junction graph framework for BP-like MEU algorithms, and provide theoretical analysis. In addition, we propose two double-loop algorithms that alleviate the issue of non-convexity in LIMIDs or provide convergence guarantees: a deterministic annealing approach suggested by Corollary 3.3 and a method based on the proximal point algorithm.

4.1 A Belief Propagation Algorithm

We start by formulating the problem (7) into the junction graph framework. Let $(\mathcal{G},\mathcal{C},\mathcal{S})$ be a junction graph for the augmented distribution $q(\boldsymbol{x}) \propto \exp(\theta(\boldsymbol{x}))$. For each decision node $i \in D$, we associate it with exactly one cluster $c_k \in \mathcal{C}$ satisfying $\{i, \mathrm{pa}(i)\} \subseteq c_k$; we call such a cluster a *decision cluster*. The clusters $\mathcal{C}$ are thus partitioned into decision clusters $\mathcal{D}$ and the other (normal) clusters $\mathcal{R}$. For simplicity, we assume each decision cluster $c_k \in \mathcal{D}$ is associated with exactly one decision node, denoted d_k.

Following the junction graph framework in Section 2.1, the MEU dual (10) (with temperature parameter ϵ) is

approximated by

$$\max_{\boldsymbol{\tau}\in\mathbb{L}}\{\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle+\sum_{k\in\mathcal{R}}H_{c_k}+\sum_{k\in\mathcal{D}}H^{\epsilon}_{c_k}-\sum_{(kl)\in\mathcal{E}}H_{s_{kl}}\},\quad(11)$$

where $H_{c_k}=H(x_{c_k})$, $H_{s_{kl}}=H(x_{s_{kl}})$ and $H^{\epsilon}(x_{c_k})=H(x_{c_k})-(1-\epsilon)H(x_{d_k}|x_{\mathrm{pa}(d_k)})$. The dependence of entropies on $\boldsymbol{\tau}$ is suppressed for compactness. Eq. (11) is similar to the objective of regular sum-product junction graph BP, except the entropy terms of the decision clusters are replaced by $H^{\epsilon}_{c_k}$.

Using a Lagrange multiplier method similar to Yedidia et al. [2005], a hybrid message passing algorithm can be derived for solving (11):

Sum messages: (normal clusters)
$$m_{k\to l}\propto\sum_{x_{c_k\setminus s_{kl}}}\psi_{c_k}m_{\sim k\setminus l},\quad(12)$$

MEU messages: (decision clusters)
$$m_{k\to l}\propto\sum_{x_{c_k\setminus s_{kl}}}\frac{\sigma_k[\psi_{c_k}m_{\sim k};\epsilon]}{m_{l\to k}},\quad(13)$$

where $\sigma_k[\cdot]$ is an operator that solves an annealed local MEU problem associated with $b(x_{c_k})\propto\psi_{c_k}m_{\sim k}$:

$$\sigma_k[b(x_{c_k});\epsilon]\overset{def}{=}b(x_{c_k})b_\epsilon(x_{d_k}|x_{\mathrm{pa}(d_k)})^{1-\epsilon}$$

where $b_\epsilon(x_{d_k}|x_{\mathrm{pa}(d_k)})$ is the "annealed" optimal policy

$$b_\epsilon(x_{d_k}|x_{\mathrm{pa}(d_k)})=\frac{b(x_{d_k},x_{\mathrm{pa}(d_k)})^{1/\epsilon}}{\sum_{x_{d_k}}b(x_{d_k},x_{\mathrm{pa}(d_k)})^{1/\epsilon}},$$

$$b(x_{d_k},x_{\mathrm{pa}(d_k)})=\sum_{x_{z_k}}b(x_{c_k}),\qquad z_k=c_k\setminus\{d_k,\mathrm{pa}(d_k)\}.$$

As $\epsilon\to0^+$, one can show that $b_\epsilon(x_{d_k}|x_{\mathrm{pa}(d_k)})$ is exactly an optimal strategy of the local MEU problem with augmented distribution $b(x_{c_k})$.

At convergence, the stationary point of (11) is:

$$\tau_{c_k}\propto\psi_{c_k}m_{\sim k}\quad\textit{for normal clusters}\quad(14)$$
$$\tau_{c_k}\propto\sigma_k[\psi_{c_k}m_{\sim k};\epsilon]\quad\textit{for decision clusters}\quad(15)$$
$$\tau_{s_{kl}}\propto m_{k\to l}m_{l\to k}\quad\textit{for separators}\quad(16)$$

This message passing algorithm reduces to sum-product BP when there are no decision clusters. The outgoing messages from decision clusters are the crucial ingredient, and correspond to solving local (annealed) MEU problems.

Taking $\epsilon\to0^+$ in the MEU message update (13) gives a fixed point algorithm for solving the original objective directly. Alternatively, one can adopt a deterministic annealing approach [Rose, 1998] by gradually decreasing ϵ, e.g., taking $\epsilon^t=1/t$ at iteration t.

Reparameterization Properties. BP algorithms, including sum-product, max-product, and hybrid message passing, can often be interpreted as reparameterization operators, with fixed points satisfying some sum (resp. max or hybrid) consistency property yet leaving the joint distribution unchanged [e.g., Wainwright et al., 2003a, Weiss et al., 2007, Liu and Ihler, 2011]. We define a set of "MEU-beliefs" $\boldsymbol{b}=\{b(x_{c_k}),b(x_{s_{kl}})\}$ by $b(x_{c_k})\propto\psi_{c_k}m_k$ for all $c_k\in\mathcal{C}$, and $b(x_{s_{kl}})\propto m_{k\to l}m_{l\to k}$; note that the "beliefs" $\boldsymbol{b}$ are distinguished from the "marginals" $\boldsymbol{\tau}$. We can show that at each iteration of MEU-BP in (12)-(13), the $\boldsymbol{b}$ satisfy

$$\textbf{Reparameterization:}\quad q(\boldsymbol{x})\propto\frac{\prod_{k\in\mathcal{V}}b(x_{c_k})}{\prod_{(kl)\in\mathcal{E}}b(x_{s_{kl}})},\quad(17)$$

and further, at a fixed point of MEU-BP we have

$$\textbf{Sum-consistency:}\ \text{(normal clusters)}\quad\sum_{c_k\setminus s_{ij}}b(x_{c_k})=b(x_{s_{kl}}),\quad(18)$$

$$\textbf{MEU-consistency:}\ \text{(decision clusters)}\quad\sum_{c_k\setminus s_{ij}}\sigma_k[b(x_{c_k});\epsilon]=b(x_{s_{kl}}).\quad(19)$$

Optimality Guarantees. Optimality guarantees of MEU-BP (with $\epsilon\to0^+$) can be derived via reparameterization. Our result is analogous to those of Weiss and Freeman [2001] for max-product BP and Liu and Ihler [2011] for marginal-MAP.

For a junction tree, a tree-order is a partial ordering on the nodes with $k\preceq l$ iff the unique path from a special cluster (called root) to l passes through k; the parent $\pi(k)$ is the unique neighbor of k on the path to the root. Given a subset of decision nodes D', a junction tree is said to be *consistent* for D' if there exists a tree-order with $s_{k,\pi(k)}\subseteq\mathrm{pa}(d_k)$ for any $d_k\in D'$.

Theorem 4.1. *Let $(\mathcal{G},\mathcal{C},\mathcal{S})$ be a consistent junction tree for a subset of decision nodes D', and $\boldsymbol{b}$ be a set of MEU-beliefs satisfying the reparameterization and consistency conditions (17)-(19) with $\epsilon\to0^+$. Let $\boldsymbol{\delta}^*=\{b_{c_k}(x_{d_k}|x_{\mathrm{pa}(d_k)})\colon d_k\in D\}$; then $\boldsymbol{\delta}^*$ is a locally optimal strategy in the sense that $\mathrm{EU}(\{\boldsymbol{\delta}^*_{D'},\boldsymbol{\delta}_{D\setminus D'}\})\le\mathrm{EU}(\boldsymbol{\delta}^*)$ for any $\boldsymbol{\delta}_{D\setminus D'}$.*

A junction tree is said to be *globally* consistent if it is consistent for all the decision nodes, which as implied by Theorem 4.1, ensures a globally optimal strategy; this notation of *global consistency* is similar to the *strong junction trees* in Jensen et al. [1994]. For IDs with perfect recall, a globally consistent junction tree can be constructed by a standard procedure which triangulates the DAG of the ID along reverse temporal order. For IDs without perfect recall, it is usually not possible to construct a globally consistent junction tree; this is the case for the toy example in Fig. 2b. However, coordinate-wise optimality follows as a consequence of Theorem 4.1 for general IDs with arbitrary junction trees, indicating that MEU-BP is at least as "optimal" as SPU.

Theorem 4.2. *Let $(\mathcal{G}, \mathcal{C}, \mathcal{S})$ be an arbitrary junction tree, and $\boldsymbol{b}$ and $\boldsymbol{\delta}^*$ defined in Theorem 4.1. Then $\boldsymbol{\delta}^*$ is a locally person-by-person optimal strategy: $\mathrm{EU}(\{\boldsymbol{\delta}_i^*, \boldsymbol{\delta}_{D\setminus i}\}) \leq \mathrm{EU}(\boldsymbol{\delta}^*)$ for any $i \in D$ and $\boldsymbol{\delta}_{D\setminus i}$.*

Additively Decomposable Utilities. Our algorithms rely on the factorization structure of the augmented distribution $q(\boldsymbol{x})$. For this reason, multiplicative utilities fit naturally, but additive utilities are more difficult (as they also are in exact inference) [Koller and Friedman, 2009]. To create factorization structure in additive utility problems, we augment the model with a latent "selector" variable, similar to that in mixture models. For details, see the appendix.

4.2 Proximal Algorithms

In this section, we present a proximal point approach [e.g., Martinet, 1970, Rockafellar, 1976] for the MEU problems. Similar methods have been applied to standard inference problems, e.g., Ravikumar et al. [2010].

We start with a brief introduction to the proximal point algorithm. Consider an optimization problem $\min_{\boldsymbol{\tau}\in\mathbb{M}} f(\boldsymbol{\tau})$. A proximal method instead iteratively solves a sequence of "proximal" problems

$$\boldsymbol{\tau}^{t+1} = \arg\min_{\boldsymbol{\tau}\in\mathbb{M}} \{f(\boldsymbol{\tau}) + w^t D(\boldsymbol{\tau}||\boldsymbol{\tau}^t)\}, \qquad (20)$$

where $\boldsymbol{\tau}^t$ is the solution at iteration t and w^t is a positive coefficient. $D(\cdot||\cdot)$ is a distance, called the proximal function; typical choices are Euclidean or Bregman distances or ψ-divergences [e.g., Teboulle, 1992, Iusem and Teboulle, 1993]. Convergence of proximal algorithms has been well studied: the objective series $\{f(\boldsymbol{\tau}^t)\}$ is guaranteed to be non-increasing at each iteration, and $\{\boldsymbol{\tau}^t\}$ converges to an optimal solution (sometimes superlinearly) for convex programs, under some regularity conditions on the coefficients $\{w^t\}$. See, e.g., Rockafellar [1976], Tseng and Bertsekas [1993], Iusem and Teboulle [1993].

Here, we use an entropic proximal function that naturally fits the MEU problem:

$$D(\boldsymbol{\tau}||\boldsymbol{\tau}') = \sum_{i\in D}\sum_{\boldsymbol{x}} \tau(\boldsymbol{x}) \log[\tau_i(x_i|x_{\mathrm{pa}(i)})/\tau_i'(x_i|x_{\mathrm{pa}(i)})],$$

a sum of conditional KL-divergences. The proximal update for the MEU dual (7) then reduces to

$$\boldsymbol{\tau}^{t+1} = \arg\max_{\boldsymbol{\tau}\in\mathbb{M}} \{\langle \boldsymbol{\theta}^t, \boldsymbol{\tau}\rangle + H(\boldsymbol{x}) - (1-w^t) H(x_i|x_{\mathrm{pa}(i)})\}$$

where $\theta^t(\boldsymbol{x}) = \theta(\boldsymbol{x}) + w^t \sum_{i\in D} \log \tau_i^t(x_i|x_{\mathrm{pa}(i)})$. This has the same form as the annealed problem (10) and can be solved by the message passing scheme (12)-(13). Unlike annealing, the proximal algorithm updates $\boldsymbol{\theta}^t$ each iteration and does not need w^t to approach zero.

We use two choices of coefficients $\{w^t\}$: (1) $w^t = 1$ (*constant*), and (2) $w^t = 1/t$ (*harmonic*). The choice $w^t = 1$ is especially interesting because the proximal update reduces to a standard *marginalization* problem, solvable by standard tools without the MEU's temporal elimination order restrictions. Concretely, the proximal update in this case reduces to

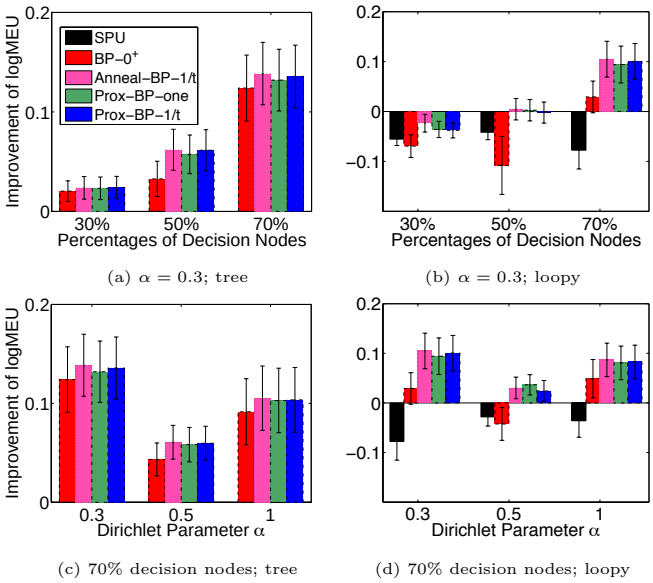
$$\tau_i^{t+1}(x_i|x_{\mathrm{pa}(i)}) \propto \tau_i^t(x_i|x_{\mathrm{pa}(i)}) \mathbb{E}(u(\boldsymbol{x})|x_{\mathrm{fam}(i)}\,;\, \boldsymbol{\delta}_{\neg i}^n)$$

with $\mathbb{E}(u(\boldsymbol{x})|x_{\mathrm{fam}(i)}\,;\, \boldsymbol{\delta}_{\neg i}^n)$ as defined in (6). This proximal update can be seen as a "soft" and "parallel" version of the greedy update (6), which makes a hard update at a single decision node, instead of a soft modification simutaneously for all decision nodes. The soft update makes it possible to correct earlier suboptimal choices and allows decision nodes to make cooperative movements. However, convergence with $w^t = 1$ may be slow; using $w^t = 1/t$ takes larger steps but is no longer a standard marginalization.

5 Experiments

We demonstrate our algorithms on several influence diagrams, including randomly generated IDs, large scale IDs constructed from problems in the UAI08 inference challenge, and finally practically motivated IDs for decentralized detection in wireless sensor networks. We find that our algorithms typically find better solutions than SPU with comparable time complexity; for large scale problems with many decision nodes, our algorithms are more computationally efficient than SPU because one step of SPU requires updating (6) (a global expectation) for all the decision nodes.

In all experiments, we test single policy updating (`SPU`), our MEU-BP running directly at zero temperature (`BP-0`$^+$), annealed BP with temperature $\epsilon^t = 1/t$ (`Anneal-BP-1/t`), and the proximal versions with $w^t = 1$ (`Prox-BP-one`) and $w^t = 1/t$ (`Prox-BP-1/t`). For the BP-based algorithms, we use two constructions of junction graphs: a standard junction tree by triangulating the DAG in backwards topological order, and a loopy junction graph following [Mateescu et al., 2010] that corresponds to Pearl's loopy BP; for SPU, we use the same junction graphs to calculate the inner update (6). The junction trees ensure the inner updates of `SPU` and `Prox-BP-one` are performed exactly, and has optimality guarantees in Theorem 4.1, but may be computationally more expensive than the loopy junction graphs. For the proximal versions, we set a maximum of 5 iterations in the inner loop; changing this value did not seem to lead to significantly different results. The BP-based algorithms may return non-deterministic strategies; we round to deterministic strategies by taking the largest values.



(a) $\alpha = 0.3$; tree (b) $\alpha = 0.3$; loopy

(c) 70% decision nodes; tree (d) 70% decision nodes; loopy

Figure 3: Results on random IDs of size 20. The y-axes show the log MEU of each algorithm compared to SPU on a junction tree. The left panels correspond to running the algorithms on junction trees, and right panels on loopy junction graphs. (a) & (b) shows MEUs as the percentage of decision nodes changes. (c) & (d) show MEUs v.s. the Dirichlet parameter α. The results are averaged on 20 random models.

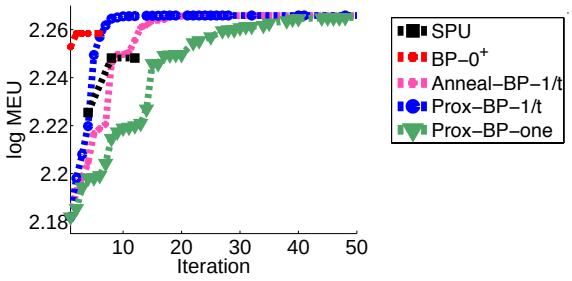
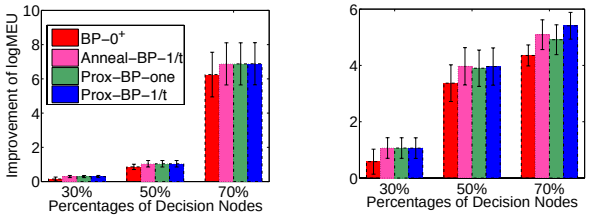


Figure 4: A typical trajectory of MEU (of the rounded deterministic strategies) v.s. iterations for the random IDs in Fig. 3. One iteration of the BP-like methods denotes a forward-backward reduction on the junction graph; One step of SPU requires $|D|$ (number of decision nodes) reductions. SPU and BP−0+ are stuck at a local model in the 2nd iteration.



(a). Diagnostic BN 1 (b). Diagnostic BN 2

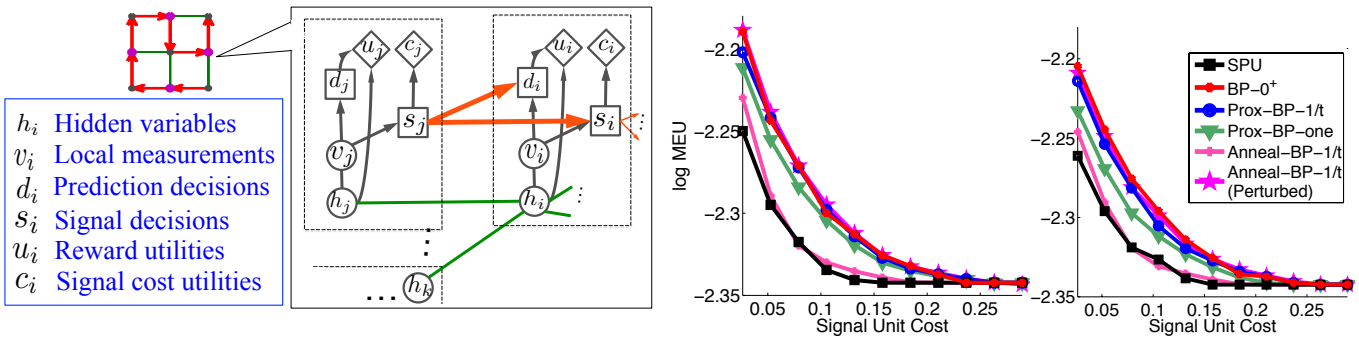
Figure 5: Results on IDs constructed from two diagnostic BNs from the UAI08 challenge. Here all algorithms used the loopy junction graph and are initialized uniformly. (a)-(b) the logMEU of algorithms normalized to that of SPU. Averaged on 10 trails.

Random Bayesian Networks. We test our algorithms on randomly constructed IDs with additive utilities. We first generate a set of random DAGs of size 20 with maximum parent size of 3. To create IDs, we take the leaf nodes to be utility nodes, and among non-leaf nodes we randomly select a fixed percentage to be decision nodes, with the others being chance nodes. We assume the chance and decision variables are discrete with 4 states. The conditional probability tables of the chance nodes are randomly drawn from a symmetric Dirichlet distribution $\mathrm{Dir}(\alpha)$, and the entries of the utility function from Gamma distribution $\Gamma(\alpha, 1)$.

The relative improvement of log MEU compared to the SPU with junction tree are reported in Fig. 3. We find that when using junction trees, all our BP-based methods dominate SPU; for loopy junction graphs, BP−0+ occasionally performs worse than SPU, but all the annealed and proximal algorithms outperform SPU with the same loopy junction graph, and often even SPU with junction tree. As the percentage of decision nodes increases, the improvement of the BP-based methods on SPU generally increases. Fig. 4 shows a typical trajectory of the algorithms across iterations. The algorithms were initialized uniformly; random initializations behaved similarly, but are omitted for space.

Diagnostic Bayesian networks. We construct larger scale IDs based on two diagnostic Bayes nets with 200-300 nodes and 300-600 edges, taken from the UAI08 inference challenge. To create influence diagrams, we made the leaf nodes utilities, each defined by its conditional probability when clamped to a randomly chosen state, and total utility as the product of the local utilities (multiplicatively decomposable). The set of decision nodes is again randomly selected among the non-leaf nodes with a fixed percentage. Since the network sizes are large, we only run the algorithms on the loopy junction graphs. Again, our algorithms significantly improve on SPU; see Fig. 5.

Decentralized Sensor Network. In this section, we test an influence diagram constructed for decentralized detection in wireless sensor networks [e.g., Viswanathan and Varshney, 1997, Kreidl and Willsky, 2006]. The task is to detect the states of a hidden process $p(h)$ (as a pairwise MRF) using a set of distributed sensors; each sensor provides a noisy measurement v_i of the local state h_i, and overall performance is boosted by allowing the sensor to transmit small (1-bit) signals s_i along an directional path, to



(a) The ID for sensor network detection (b) Junction tree (c) Loopy junction graph

Figure 6: (a) A sensor network on 3×3 grid; green lines denote the MRF edges of the hidden process $p(h)$, on some of which (red arrows) signals are allowed to pass; each sensor may be accurate (purple) or noisy (black). Optimal strategies should pass signals from accurate sensors to noisy ones but not the reverse. (b)-(c) The log MEU of algorithms running on (b) a junction tree and (c) a loopy junction graph. As the signal cost increases, all algorithms converge to the communication-free strategy. Results averaged on 10 random trials.

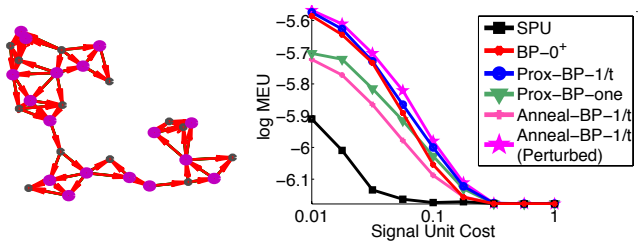
help the predictions of their downstream sensors. The utility function includes rewards for correct prediction and a cost for sending signals. We construct an ID as sketched in Fig. 6(a) for addressing the *offline* policy design task, finding optimal policies of how to predict the states based on the local measurement and received signals, and policies of whether and how to pass signals to downstream nodes; see appendix for more details.

To escape the "all-zero" fixed point, we initialize the proximal algorithms and SPU with 5 random policies, and `BP-0+` and `Anneal-BP-1/t` with 5 random messages. We first test on a sensor network on a 3×3 grid, where the algorithms are run on both a junction tree constructed by standard triangulation and a loopy junction graph (see the Appendix for construction details). As shown in Fig. 6(b)-(c), `SPU` performs worst in all cases. Interestingly, `Anneal-BP-1/t` performs relatively poorly here, because the annealing steps make it insensitive to and unable to exploit the random initializations; this can be fixed by a "perturbed" annealed method that injects a random perturbation into the model, and gradually decreases the perturbation level across iterations (`Anneal-BP-1/t (Perturbed)`).

A similar experiment (with only the loopy junction graph) is performed on the larger random graph in Fig. 7; the algorithm performances follow similar trends. SPU performs even worse in this case since it appears to over-send signals when two "good" sensors connect to one "bad" sensor.

6 Related Works

Many exact algorithms for ID have been developed, usually in a variable-elimination or message-passing form; see Koller and Friedman [2009] for a recent review. Approximation algorithms are relatively unexplored, and usually based on separately approximating individual components of exact algorithms [e.g., Sabbadin et al., 2011, Sallans, 2003]; our method instead builds an integrated framework. Other approaches, including MCMC [e.g., Charnes and Shenoy, 2004] and search methods [e.g., Marinescu, 2010], also exist but are usually more expensive than SPU or our BP-like methods. See the appendix for more discussion.



(a) Sensor network (b) Loopy junction graph

Figure 7: The results on a sensor network on a random graph with 30 nodes (the MRF edges overlap with the signal paths). Averaged on 5 random models.

7 Conclusion

In this work we derive a general variational framework for influence diagrams, for both the "convex" centralized decisions with perfect recall and "non-convex" decentralized decisions. We derive several algorithms, but equally importantly open the door for many others that can be applied within our framework. Since these algorithms rely on *decomposing* the global problems into local ones, they also open the possibility of efficiently distributable algorithms.

Acknowledgements. Work supported in part by NSF IIS-1065618 and a Microsoft Research Fellowship.

References

C. Bielza, P. Muller, and D. R. Insua. Decision analysis by augmented probability simulation. *Management Science*, 45(7):995–1007, 1999.

J. Charnes and P. Shenoy. Multistage Monte Carlo method for solving influence diagrams using local computation. *Management Science*, pages 405–418, 2004.

A. Detwarasiti and R. D. Shachter. Influence diagrams for team decision analysis. *Decision Anal.*, 2, Dec 2005.

R. Howard and J. Matheson. Influence diagrams. In *Readings on Principles & Appl. Decision Analysis*, 1985.

R. Howard and J. Matheson. Influence diagrams. *Decision Anal.*, 2(3):127–143, 2005.

A. Iusem and M. Teboulle. On the convergence rate of entropic proximal optimization methods. *Computational and Applied Mathematics*, 12:153–168, 1993.

F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In *UAI*, pages 367–373. Morgan Kaufmann, 1994.

J. Jiang, P. Rai, and H. Daumé III. Message-passing for approximate MAP inference with latent variables. In *NIPS*, 2011.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221, 2003.

O. P. Kreidl and A. S. Willsky. An efficient message-passing algorithm for optimizing decentralized detection networks. In *IEEE Conf. Decision Control*, Dec 2006.

S. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Managment Sci.*, pages 1235–1251, 2001.

Q. Liu and A. Ihler. Variational algorithms for marginal MAP. In *UAI*, Barcelona, Spain, July 2011.

R. Marinescu. A new approach to influence diagrams evaluation. In *Research and Development in Intelligent Systems XXVI*, pages 107–120. Springer London, 2010.

B. Martinet. Régularisation d'inéquations variationnelles par approximations successives. *Revue Française dInformatique et de Recherche Opérationelle*, 4:154–158, 1970.

R. Mateescu, K. Kask, V. Gogate, and R. Dechter. Join-graph propagation algorithms. *Journal of Artificial Intelligence Research*, 37:279–328, 2010.

J. Pearl. Influence diagrams - historical and personal perspectives. *Decision Anal.*, 2(4):232–234, dec 2005.

P. Ravikumar, A. Agarwal, and M. J. Wainwright. Message-passing for graph-structured linear programs: Proximal projections, convergence, and rounding schemes. *Journal of Machine Learning Research*, 11: 1043–1080, Mar 2010.

R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877, 1976.

K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. IEEE*, 86(11):2210 –2239, Nov 1998.

R. Sabbadin, N. Peyrard, and N. Forsell. A framework and a mean-field algorithm for the local control of spatial processes. *International Journal of Approximate Reasoning*, 2011.

B. Sallans. Variational action selection for influence diagrams. Technical Report OEFAI-TR-2003-29, Austrian Research Institute for Artificial Intelligence, 2003.

R. Shachter. Model building with belief networks and influence diagrams. *Advances in decision analysis: from foundations to applications*, page 177, 2007.

M. Teboulle. Entropic proximal mappings with applications to nonlinear programming. *Mathematics of Operations Research*, 17(3):pp. 670–690, 1992.

P. Tseng and D. Bertsekas. On the convergence of the exponential multiplier method for convex programming. *Mathematical Programming*, 60(1):1–19, 1993.

R. Viswanathan and P. Varshney. Distributed detection with multiple sensors: part I – fundamentals. *Proc. IEEE*, 85(1):54 –63, Jan 1997.

M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, 2008.

M. Wainwright, T. Jaakkola, and A. Willsky. A new class of upper bounds on the log partition function. *IEEE Trans. Info. Theory*, 51(7):2313–2335, July 2005.

M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Trans. Info. Theory*, 45:1120–1146, 2003a.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. MAP estimation via agreement on (hyper) trees: Message-passing and linear programming approaches. *IEEE Trans. Info. Theory*, 51(11):3697 – 3717, Nov 2003b.

Y. Weiss and W. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Trans. Info. Theory*, 47(2):736 –744, Feb 2001.

Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *UAI*, 2007.

D. Wolpert. Information theory – the bridge connecting bounded rational game theory and statistical physics. *Complex Engineered Systems*, pages 262–290, 2006.

J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized BP algorithms. *IEEE Trans. Info. Theory*, 51, July 2005.

N. L. Zhang. Probabilistic inference in influence diagrams. In *Computational Intelligence*, pages 514–522, 1998.

N. L. Zhang, R. Qi, and D. Poole. A computational theory of decision networks. *Int. J. Approx. Reason.*, 11:83–158, 1994.

Closed-Form Learning of Markov Networks from Dependency Networks

Daniel Lowd
Dept. of Computer and Information Science
University of Oregon
Eugene, OR 97403
lowd@cs.uoregon.edu

Abstract

Markov networks (MNs) are a powerful way to compactly represent a joint probability distribution, but most MN structure learning methods are very slow, due to the high cost of evaluating candidates structures. Dependency networks (DNs) represent a probability distribution as a set of conditional probability distributions. DNs are very fast to learn, but the conditional distributions may be inconsistent with each other and few inference algorithms support DNs. In this paper, we present a closed-form method for converting a DN into an MN, allowing us to enjoy both the efficiency of DN learning and the convenience of the MN representation. When the DN is consistent, this conversion is exact. For inconsistent DNs, we present averaging methods that significantly improve the approximation. In experiments on 12 standard datasets, our methods are orders of magnitude faster than and often more accurate than combining conditional distributions using weight learning.

1 INTRODUCTION

Joint probability distributions are useful for representing and reasoning about uncertainty in many domains, from robotics to medicine to molecular biology. One of the most powerful and popular ways to represent a joint probability distribution compactly is with a Markov network (MN), an undirected probabilistic graphical model. Inference in an MN can compute marginal or conditional probabilities, or find the most probable configuration of a subset of variables given evidence. Although exact inference is usually intractable, numerous approximate inference algorithms exist to exploit the structure present in the MN representation.

One of the largest disadvantages of Markov networks is the difficulty of learning them from data. In the fully observed case, weight learning is a convex optimization problem, but cannot be done in closed form except in very special circumstances, e.g. [18]. Structure learning is even harder, since it typically involves a search over a large number of candidate structures, and weights must be learned for each candidate before it can be scored [4, 14, 3]. An increasingly popular alternative is to use local methods to select the structure, by finding the dependencies for each variable separately and combining them using weight learning [15, 12].

Dependency networks (DNs) [8] represent the joint distribution itself as a set of conditional probability distributions, one for each variable. In terms of representational power, DNs are comparable to MNs, since every MN can be represented as a consistent DN and vice versa. The advantage of DNs is that they are much easier to learn than MNs, since each conditional probability distribution can be learned separately. However, the local distributions may not be consistent with any joint distribution, making the model difficult to interpret. Furthermore, very few inference algorithms support DNs. As a result, DNs remain much less popular than MNs.

In this paper, we present a new method for converting dependency networks into Markov networks, allowing us to enjoy both the efficiency of DN learning and the convenience of the MN representation. Surprisingly, this translation can be done in closed-form without any kind of search or optimization. If the DN is consistent, then this conversion is exact. For the general case of an inconsistent DN, we present averaging methods that significantly improve the approximation. As long as the maximum feature length is bounded by a constant, our method runs in linear time and space with respect to the size of the original DN.

Our method is similar in spirit to previous work on learning MNs by combining local, conditional distributions with weight learning [12, 15]. However, instead of ignoring the parameters of those local distributions, we use them to directly compute the parameters of the joint distribution. Hulten et al. [9] proposed a method for converting DNs to Bayesian networks. However, this conversion process re-

quired searching through possible structures and led to less accurate models than learning a BN directly from data.

In experiments on 12 standard datasets, we find that our DN2MN conversion method is orders of magnitude faster than weight learning and often more accurate. For DNs with decision tree conditional distributions, the converted MN was often more accurate than the original DN. With logistic regression conditional distributions, the converted MNs were significantly more accurate than performing weight learning.

Our method has potential applications outside of MN structure learning as well. For domains where data is limited or unavailable, an expert could specify the conditional distributions of a DN, which could then be translated into the joint distribution of an MN. This could be much easier and less error-prone than attempting to specify the entire joint distribution directly.

Our paper is organized as follows. We begin with background on Markov networks and dependency networks in Sections 2 and 3. In Section 4, we describe how to convert consistent dependency networks into Markov networks that represent the exact same probability distribution. In Section 5, we discuss inconsistent conditional probability distributions and methods for improving the approximation quality through averaging. We evaluate our methods empirically in Section 6 and conclude in Section 7.

2 MARKOV NETWORKS

A *Markov network* (MN) represents a probability distribution over a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ as a normalized product of factors:

$$P(\mathbf{X}) = \frac{1}{Z} \prod_i \phi_i(\mathbf{D}_i) \tag{1}$$

Z is the partition function, a normalization constant to make the distribution sum to one; ϕ_i is the ith factor, sometimes referred to as a potential function; and $\mathbf{D}_i \subset \mathbf{X}$ is the set of variables in the domain of ϕ_i.

A probability distribution where $P(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathbf{X}$ is said to be *positive*. An MN that represents a positive distribution can also be written as a *log-linear model*. In a log-linear model, the probability distribution is expressed as an exponentiated weighted sum of feature functions $f_i(\mathbf{D}_i)$ rather than as a product of factors:

$$P(\mathbf{X}) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(\mathbf{D}_i)\right) \tag{2}$$

The correspondence between (1) and (2) is easily shown by letting $f_i = \log \phi_i$ and $w_i = 1$.

For discrete domains, a common choice is to use conjunctions of variable tests as the features. Each test is of the form $(X_i = v_i)$, where X_i is a variable in $\mathbf{X}$ and v_i is a value of that variable. We sometimes abbreviate tests of Boolean variables, so that $(X_i = T)$ is written as X_i and $(X_i = F)$ is written as $\neg X_i$. A conjunctive feature equals 1 if its arguments satisfy the conjunction and 0 otherwise. Any factor represented as a table can be converted into a set of conjunctive features with one feature for each entry in the table. When the factors have repeated values or other types of structure, a feature-based representation is often more compact than a tabular one.

The *Markov blanket* of a variable X_i, denoted $\mathrm{MB}(X_i)$, is the set of variables that render X_i independent from all other variables in the domain. In an MN, this set consists of all variables that appear in a factor or feature with X_i. These independencies, and others, are entailed by the factorization in (1).

2.1 INFERENCE

Given an MN, we often wish to answer queries such as the probability of one or more variables given evidence. In general, computing exact marginal and conditional probabilities is #P-complete [16], so approximate inference algorithms are commonly used instead. One of the most popular is Gibbs sampling [5].

Gibbs sampling is a Markov chain Monte Carlo method that generates a set of random samples and uses them to answer queries. The sampler is initialized to a random state consistent with the evidence. Samples are generated by resampling each non-evidence variable in turn, according to its conditional probability given the current states of the other variables. In early samples, the sampler may be in an unlikely and non-representative state, so these "burn-in" samples are typically excluded from consideration. The probability of a query is estimated as the fraction of samples consistent with the query. For positive distributions, Gibbs sampling will eventually converge to the correct probabilities, but this can take a very long time and convergence can be difficult to detect.

2.2 WEIGHT LEARNING

In maximum likelihood parameter estimation, the goal is to select a set of parameters that maximizes the log-likelihood of the model on the training data. Here we assume that the MN is expressed as a log-linear model with a fixed set of features and that all variables are observed in the training data. Since the log-likelihood of an MN is a concave function of the weights, parameter learning can be framed as a convex optimization problem and solved with standard gradient-based approaches. Since log-likelihood and its gradient are usually intractable to compute exactly and slow to approximate, a commonly-used alternative is pseudo-likelihood. Pseudo-log-likelihood (PLL) [2] is the sum of the conditional log-likelihood of each variable given

the other variables:

$$\log P_w^{\bullet}(\mathbf{X}=\mathbf{x}) = \sum_{i=1}^{n} \log P_w(X_i=x_i|\mathbf{X}_{-i}=\mathbf{x}_{-i})$$

where $\mathbf{X}_{-i} = \mathbf{X} - X_i$, the set of all variables except for X_i. Unlike log-likelihood, PLL and its gradient can both be evaluated efficiently. Like log-likelihood, PLL is a concave function, so any local optimum is also a global optimum. The main disadvantage of PLL is that it tends to handle long-range dependencies poorly. Optimizing PLL optimizes the model's ability to predict individual variables given large amounts of evidence, and this may lead to worse performance when less evidence is available. To control overfitting, a zero-mean Gaussian prior is typically placed on each weight.

2.3 STRUCTURE LEARNING

The goal of MN structure learning is to find a set of factors and parameters or features and weights with a high score on the training data. As with weight learning, pseudo-likelihood is a common choice due to the intractability of the partition function. Of the many MN structure learning variations that have been proposed, we touch on just a few algorithms and their main themes.

One common approach to structure learning is to perform a greedy search through the space of possible structures. Della Pietra et al. [4] conduct this search in a top-down manner, starting with atomic features (individual variable tests) and extending and combining these simple features until convergence. Davis and Domingos [3] propose a bottom-up search instead, creating many initial features based on the training data and repeatedly merging features as learning progresses. Search-based algorithms tend to be slow, since scoring candidate structures can be expensive and there are many candidate structures to consider.

Another approach is to use local models to find the Markov blanket of each variable separately and then combine the features of these local models into a global structure. Ravikumar et al. [15] do this by learning a logistic regression model with L1 regularization for each variable. The use of L1 regularization encourages sparsity, so that most of the interaction weights are zero. All non-zero interactions are turned into conjunctive features. In other words, if the logistic regression model for predicting X_i has a non-zero weight for X_j then the feature $X_i \wedge X_j$ is added to the model. Lowd and Davis [12] adopt a similar approach, but use probabilistic decision trees as the local model. Each decision tree is converted into a set of conjunctive features by considering each path from the tree root to a leaf as a conjunction of variable tests. In both works, the authors select feature weights by performing weight learning. In addition to being much faster than the search-based methods, the local approaches often lead to more accurate models.

3 DEPENDENCY NETWORKS

A *dependency network* (DN) [8] consists of a set of conditional probability distributions (CPDs) $P_i(X_i|\mathrm{MB}(X_i))$, each defining the probability of a single variable given its Markov blanket. A DN is said to be *consistent* if there exists a probability distribution P that is consistent with the DN's conditional distributions. Inconsistent DNs are sometimes called *general* dependency networks.

Since Gibbs sampling only uses the conditional probability of each variable given its Markov blanket, it is easily applied to DNs. The probability distribution represented by a DN is defined as the stationary distribution of the Gibbs sampler, given a fixed variable order. If the DN is consistent, then its conditional distributions must be consistent with some MN, and Gibbs sampling converges to the same distribution as the MN. If the DN is inconsistent, then the stationary distribution may depend on the order in which the variables are resampled. Furthermore, the joint distribution determined by the Gibbs sampler may be inconsistent with the conditional probabilities in the CPDs.

Few other inference algorithms have been defined for DNs. Heckerman et al. [8] describe a method for using Gibbs sampling and CPD values together to estimate rare probabilities with lower variance; Toutanova et al. [17] do maximum a posteriori (MAP) inference in a chain-structured DN with an adaptation of the Viterbi algorithm; and Lowd and Shamaei [13] propose mean field inference for DNs. We know of no other methods.

A DN can be constructed from any MN by constructing a conditional probability distribution for each variable given its Markov blanket. A DN can also be learned from data by learning a probabilistic classifier for each variable. This makes DN learning trivial to parallelize with a separate process for each variable. However, when learned from data, the resulting CPDs may be inconsistent with each other.

Any standard method for learning a probabilistic classifier can be used. Heckerman et al. [8] learn a probabilistic decision tree for each variable. In a probabilistic decision tree, the interior nodes of the tree are variable tests (such as $(X_4 = T)$), the branches are labeled with test outcomes (such as *true* and *false*), and the leaves specify the marginal distribution of the target variable given the tests on the path from the root to leaf. Probabilistic decision trees can be learned greedily to maximize the conditional log-likelihood of the target variable.

Hulten et al. [9] investigated the problem of converting a DN to a Bayesian network (BN). However, rather than trying to match the distribution as closely as possible, they tried to find the highest-scoring acyclic subset of the DN structure. After proving the problem to be NP-hard, they proposed a greedy algorithm for removing the least essential edges. The resulting BN was consistently less accurate

than training one from scratch. Furthermore, it only applies to DNs with tree CPDs.

4 CONVERTING CONSISTENT DEPENDENCY NETWORKS

We now discuss how to construct a joint distribution from a set of positive conditional distributions. To begin with, we assume that the conditional distributions, $P(X_i|\mathbf{X}_{-i})$, are the conditionals of some unknown joint probability distribution P. Later, we will relax this assumption and discuss how to find the most effective approximation.

Consider two instances, $\mathbf{x}$ and $\mathbf{x}'$, that only differ in the state of one variable, X_j. In other words, $x_i = x'_i$ for all $i \neq j$, so $\mathbf{x}_{-j} = \mathbf{x}'_{-j}$. We can express the ratio of their probabilities as follows:

$$\frac{P(\mathbf{x})}{P(\mathbf{x}')} = \frac{P(x_j, \mathbf{x}_{-j})}{P(x'_j, \mathbf{x}_{-j})} = \frac{P(x_j|\mathbf{x}_{-j})P(\mathbf{x}_{-j})}{P(x'_j|\mathbf{x}_{-j})P(\mathbf{x}_{-j})} = \frac{P(x_j|\mathbf{x}_{-j})}{P(x'_j|\mathbf{x}_{-j})} \tag{3}$$

Note that the final expression only involves a conditional distribution, $P(X_j|X_{-j})$, not the full joint. The conditional distribution must be positive in order to avoid division by zero.

If $\mathbf{x}$ and $\mathbf{x}'$ differ in multiple variables, then we can express their probability ratio as the product of multiple single-variable transitions. We construct a sequence of instances $\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$, each instance differing in at most one variable from the previous instance in the sequence. Let order $o \in S_n$ be a permutation of the numbers 1 to n and let $o[i]$ refer to the ith number in the order. We define $\mathbf{x}^{(i)}$ inductively:

$$\mathbf{x}^{(0)} = \mathbf{x}$$
$$\mathbf{x}^{(i)} = (x'_{o[i]}, \mathbf{x}^{(i-1)}_{-o[i]})$$

In other words, the ith instance $\mathbf{x}^{(i)}$ simply changes the $o[i]$th variable from $x_{o[i]}$ to $x'_{o[i]}$ and is otherwise identical to the previous element, $\mathbf{x}^{(i-1)}$. Thus, in $\mathbf{x}^{(i)}$ the first i variables in the order are set to their values in $\mathbf{x}'$ and the latter $n-i$ are set to their values in $\mathbf{x}$. Note that $\mathbf{x}^{(n)} = \mathbf{x}'$, since all n variables have been changed to their values in $\mathbf{x}'$.

We can use these intermediate instances to express the ratio $P(\mathbf{x})/P(\mathbf{x}')$ as a product:

$$\begin{aligned}
\frac{P(\mathbf{x})}{P(\mathbf{x}')} &= \frac{P(\mathbf{x}^{(0)})}{P(\mathbf{x}^{(n)})} \\
&= \frac{P(\mathbf{x}^{(0)})}{P(\mathbf{x}^{(n)})} \times \frac{P(\mathbf{x}^{(1)})}{P(\mathbf{x}^{(1)})} \times \cdots \times \frac{P(\mathbf{x}^{(n-1)})}{P(\mathbf{x}^{(n-1)})} \\
&= \frac{P(\mathbf{x}^{(0)})}{P(\mathbf{x}^{(1)})} \times \frac{P(\mathbf{x}^{(1)})}{P(\mathbf{x}^{(2)})} \times \cdots \times \frac{P(\mathbf{x}^{(n-1)})}{P(\mathbf{x}^{(n)})} \\
&= \prod_{i=1}^{n} \frac{P(\mathbf{x}^{(i-1)})}{P(\mathbf{x}^{(i)})} = \prod_{i=1}^{n} \frac{P(x_{o[i]}|\mathbf{x}^{(i)}_{-o[i]})}{P(x'_{o[i]}|\mathbf{x}^{(i)}_{-o[i]})}
\end{aligned} \tag{4}$$

The last equality follows from substituting (3), since $\mathbf{x}^{(i-1)}$ and $\mathbf{x}^{(i)}$ only differ in the $o[i]$th variable, $X_{o[i]}$.

Letting $\phi_i(\mathbf{X} = \mathbf{x}) = P(x_{o[i]}|\mathbf{x}^{(i)}_{-o[i]})/P(x'_{o[i]}|\mathbf{x}^{(i)}_{-o[i]})$ and $Z = 1/P(\mathbf{x}')$:

$$\frac{1}{Z}\prod_i \phi_i(\mathbf{x}) = P(\mathbf{x}')\frac{P(\mathbf{x})}{P(\mathbf{x}')} = P(\mathbf{x})$$

Therefore, a Markov network with the factors $\{\phi_i\}$ exactly represents the probability distribution $P(\mathbf{X})$. Since the factors are defined only in terms of conditional distributions of one variable given evidence, any consistent dependency network can be converted to a Markov network representing the exact same distribution. This holds for any ordering o and base instance $\mathbf{x}'$.

Note that, unlike $\mathbf{x}'$, $\mathbf{x}$ is not a fixed vector of values but can be set to be any instance in the state space. This is necessary for ϕ_i to be a function $\mathbf{x}$. The following subsection will make this clearer with an example.

4.1 EXAMPLE

We now show how this idea can be applied to a simple, consistent DN. Consider the following conditional distributions over binary variables X_1 and X_2:

$$\begin{aligned}
P(X_1 = T|X_2 = T) &= 4/5 \\
P(X_1 = T|X_2 = F) &= 2/5 \\
P(X_2 = T|X_1 = T) &= 2/3 \\
P(X_2 = T|X_1 = F) &= 1/4
\end{aligned}$$

Let $\mathbf{x}' = [T, T]$ and $o = [1, 2]$. Following the earlier construction:

$$\phi_1(x_1, x_2) = \frac{P(x_1|\mathbf{x}^{(1)}_{-1})}{P(x'_1|\mathbf{x}^{(1)}_{-1})} = \frac{P(x_1|x_2)}{P(X_1 = T|x_2)}$$

$$\phi_2(x_2) = \frac{P(x_2|\mathbf{x}^{(2)}_{-2})}{P(x'_2|\mathbf{x}^{(2)}_{-2})} = \frac{P(x_2|X_1 = T)}{P(X_2 = T|X_1 = T)}$$

Note that ϕ_2 is not a function of x_1. This is because $x^{(2)}_1 = x'_1$, so the value of X_1 in the evidence is defined by the base

instance, $\mathbf{x}'$. In general, the ith converted factor ϕ_i is not a function of the first $i-1$ variables, since they are fixed to values in $\mathbf{x}'$.

By simplifying the factors, we obtain the following:

X_1	X_2	$\phi_1(X_1, X_2)$
T	T	1
T	F	1
F	T	1/4
F	F	3/2

X_2	$\phi_2(X_2)$
T	1
F	1/2

Multiplying the factors together and renormalizing yields:

X_1	X_2	$P(X_1, X_2)$
T	T	0.4
T	F	0.2
F	T	0.1
F	F	0.3

which matches both conditional distributions. Therefore, an MN with the factors ϕ_1 and ϕ_2 represents the exact same distribution as a DN with the conditional distributions $P(X_1|X_2)$ and $P(X_2|X_1)$. Since the original DN is consistent, any choice of $\mathbf{x}'$ or o will lead to the same result.

4.2 LOG-LINEAR MODELS WITH CONJUNCTIVE FEATURES

In the previous example, we showed how to convert a DN to an MN using tables as factors. However, this can be very inefficient when the conditional distributions have structure, such as decision trees or logistic regression models. Rather than treating each type of CPD separately, we discuss how to convert any distribution that can be represented as a log-linear model with conjunctive features.

Suppose the CPD for X_i is a log-linear model:

$$P(X_i|\mathbf{X}_{-i}) = \frac{1}{Z(\mathbf{X}_{-i})} \exp\left(\sum_j w_j f_j(\mathbf{D}_j)\right)$$

Note that the normalization Z is now a function of the evidence variables, $\mathbf{X}_{-i}$. To represent $P(X_i|\mathbf{x}^{(i)}_{-o[i]})$, we can condition each f_j on $\mathbf{x}^{(i)}_{-o[i]}$ separately:

$$P(X_i|\mathbf{x}^{(i)}_{-o[i]}) = \frac{1}{Z(\mathbf{x}^{(i)}_{-o[i]})} \exp\left(\sum_j w_j f_j(X_i, \mathbf{x}^{(i)}_{-o[i]})\right)$$

$\mathbf{x}^{(i)}_{-o[i]}$ uses values from $\mathbf{x}'$ for the first i variables in o and $\mathbf{x}$ for the rest. The values from $\mathbf{x}'$ are constant but those in $\mathbf{x}$ are free variables, so that the resulting distribution is a function of $\mathbf{x}$. When simplifying $f_j(\mathbf{x}^{(i)}_{-o[i]})$, if the constant values in $\mathbf{x}^{(i)}_{-o[i]}$ violate one of the variable tests in f_j, then f_j is always zero and it can be removed entirely. Otherwise, any conditions satisfied by constant values in $\mathbf{x}^{(i)}_{-o[i]}$ are always satisfied and can be removed.

Table 1: The Basic DN2MN Algorithm

function DN2MN($\{P_i(X_i|\mathbf{X}_{-i})\}, \mathbf{x}', o^{-1}$)
$M \leftarrow \emptyset$
for $i = 1$ to n **do**
  Convert P_i to a set of weighted features, F_i.
  for each weighted feature $(w, f) \in F_i$ **do**
    $f_n \leftarrow$ SIMPLIFYFEATURE($i, f, \mathbf{x}', o^{-1}$, true)
    $f_d \leftarrow$ SIMPLIFYFEATURE($i, f, \mathbf{x}', o^{-1}$, false)
    $M = M \cup (w, f_n) \cup (-w, f_d)$
  end for
end for
return M

For example, suppose $P(X_2|X_1, X_4)$ uses the following three conjunctive features:

$$f_1(X_1, X_2, X_4) = X_1 \wedge \neg X_2 \wedge X_4$$
$$f_2(X_1, X_2) = X_1 \wedge X_2$$
$$f_3(X_2, X_4) = X_2 \wedge \neg X_4$$

If $\mathbf{x}' = [T, T, T, T]$ and $o = [1, 2, 3, 4]$, then $\mathbf{x}^{(2)} = [X_1, X_2, T, T]$. After conditioning f_1, f_2, and f_3 on $\mathbf{x}^{(2)}_{-o[2]}$, they simplify to:

$$f_1(X_1, X_2) = X_1 \wedge \neg X_2$$
$$f_2(X_1, X_2) = X_1 \wedge X_2$$

f_3 is removed entirely, since it is inconsistent with $\mathbf{x}^{(2)}_{-o[2]}$.

To compute ϕ_2, we must additionally condition the function in the denominator on $X_2 = \mathbf{x}'_2 = T$. If the weights of f_1 and f_2 are w_1 and w_2, respectively, then:

$$\begin{aligned}
&\phi_2(X_1, X_2) \\
&\quad = \frac{P(X_2|X_1, X_4 = T)}{P(X_2 = T|X_1, X_4 = T)} \\
&\quad = \frac{\exp(w_1(X_1 \wedge \neg X_2) + w_2(X_1 \wedge X_2))/Z(\mathbf{x}^{(2)}_{-o[2]})}{\exp(w_2(X_1 \wedge X_2))/Z(\mathbf{x}^{(2)}_{-o[2]})} \\
&\quad = \exp(w_1(X_1 \wedge \neg X_2) + w_2(X_1 \wedge X_2) - w_2 X_1)
\end{aligned}$$

Note that the final factor ϕ_i is always a log-linear model where each feature is a subset of one of the features in the original conditional distribution. Therefore, converting each conditional distribution in this manner yields an MN represented as a log-linear model.

We summarize our complete method for consistent DNs with the algorithms in Tables 1 and 2. DN2MN takes a set of conditional probability distributions, $\{P_i(X_i|\mathbf{X}_{-i})\}$, a base instance $\mathbf{x}'$, and an inverse variable ordering o^{-1}. The inverse variable ordering is a mapping from variables to their corresponding indices in the desired ordering o, so that $o^{-1}[o[i]] = i$. DN2MN first converts the conditional

Table 2: Feature Simplifying Subroutine

function SIMPLIFYFEATURE($i, f, \mathbf{x}', o^{-1}$, *numerator*)
$f' \leftarrow \emptyset$
for each variable test $(X_j = v_j) \in f$ **do**
  if $o^{-1}[i] < o^{-1}[j]$ OR ($i = j$ AND *numerator*) **then**
    $f' \leftarrow f' \cup (X_j = v_j)$
  else if $v_j \neq x'_j$ **then**
    return $\emptyset$
  end if
end for
return f'

distributions into sets of weighted features. Then it conditions each feature on some of the values in $\mathbf{x}'$, depending on the variable order, to produce the simplified distributions for the numerator and denominator of (4). This is handled by SIMPLIFYFEATURE, which accepts a parameter to indicate if the simplified feature is for the numerator or denominator. Features in the denominator are assigned negative weights, since $1/\exp(wf) = \exp(-wf)$.

5 HANDLING INCONSISTENCIES

When the basic DN2MN algorithm is applied to an inconsistent DN, the result may depend on $\mathbf{x}'$ and o. For example, consider the following conditional distributions:

$$P_1(X_1 = T|X_2 = T) = 4/5$$
$$P_1(X_1 = T|X_2 = F) = 1/5$$
$$P_2(X_2 = T|X_1 = T) = 1/5$$
$$P_2(X_2 = T|X_1 = F) = 4/5$$

P_1 encourages X_1 and X_2 to be equal, while P_2 encourages them to have opposite values. By symmetry, it is easy to see that a Gibbs sampler with these transition probabilities must converge to a uniform distribution. However, the conditional distributions are not consistent with a uniform distribution, so the DN is inconsistent. We see that the choice of $\mathbf{x}'$ here does affect the converted MN:

When $\mathbf{x}' = [T, T]$ and $o = [1, 2]$:

X_1	X_2	$P(X_1, X_2)$
T	T	4/85
T	F	16/85
F	T	1/85
F	F	64/85

When $\mathbf{x}' = [F, F]$ and $o = [1, 2]$:

X_1	X_2	$P(X_1, X_2)$
T	T	64/85
T	F	1/85
F	T	16/85
F	F	4/85

Unsurprisingly, changing the ordering o also changes the resulting probability distribution for this example. However, no single choice of $\mathbf{x}'$ and o leads to the true, uniform distribution. To obtain a uniform distribution, we must average over several conversions.

5.1 AVERAGING OVER ALL BASE INSTANCES

A reasonable way to average is to convert the DN to an MN k times and take the geometric mean of the resulting distributions. With a log-linear model, this can be done by simply combining all features from all models and dividing the weights by k.

However, instead of merely summing over a small number of base instances $\mathbf{x}'$, we can sum over *all* base instances in linear time and space by exploiting the structure of conjunctive features. Given a feature f_j and an intermediate instance $\mathbf{x}^{(i)}_{-o[i]}$, suppose that k binary-valued variables in f_j have constant values in $\mathbf{x}^{(i)}_{-o[i]}$. Out of all possible $\mathbf{x}'$, only the fraction 2^{-k} will be consistent with the instance. Therefore, rather than enumerating all possible $\mathbf{x}'$, we can simply multiply the weight of f_j by 2^{-k}, to reflect the fact that it would be included in only that fraction of the converted MNs.

For inconsistent DNs, a uniform distribution over all $\mathbf{x}'$ may not be a good approach, since the DN may be more inconsistent in less probable regions of the state space, leading to worse behavior. We have found that a product of marginals distribution over $\mathbf{x}'$ works well. We use training data to estimate the marginal distribution of each variable, and then use those marginals to compute the modified weight of each conditioned feature. The uniform approach remains a special case. We also experimented with averaging over all training examples, but found that using the marginals worked slightly better.

5.2 AVERAGING OVER ORDERINGS

Since the conversion depends on the ordering, we would like to average over all possible orderings as well. Since every feature in the final model is a subset of one of the original features, a feature with k variable tests has at most 2^k subsets to consider. When converting the conditional distribution for variable X_i, all variables that come after i in the ordering are constant, so variable tests involving those variables will be removed. Consider a specific conjunctive subfeature that contains only l of the original k variable tests. There are $k!$ total orderings of the variables in the k conditions. There are $(l-1)!$ ways to order the remaining conditions (excluding the target), and $(k-l)$ ways to order the conditions that were removed.[1] Therefore, the fraction of orderings consistent with a particular subfeature is: $(l-1)!(k-l)!/k!$.

In a model with long conjuncts, the exponential number of subfeatures leads to prohibitively large model sizes. Alternately, we can sum over a *linear* number of orderings. Given a base ordering, o, we sum over all n orderings of the form:

$$[o[i], o[i+1], \ldots, o[n], o[1] \ldots, o[i-1]]$$

[1] We assume each variable appears in at most one condition, so they can be ordered independently. This assumption can easily be relaxed.

for all i. When averaging over a linear number of orderings, a conjunctive feature with k variable tests will be converted to k subfeatures, each determined by which of the variables comes first in the ordering.

For example, suppose the conditional distribution for X_7 contains the feature $X_4 \wedge X_6 \wedge X_7 \wedge X_{13}$ and we are summing over all rotations of $[1, 2, \ldots, n]$:

- If X_7 is first in the order, then the conditioned feature is $X_4 \wedge X_6 \wedge X_7 \wedge X_{13}$.
- If X_5 or X_6 comes first, then X_6 is fixed to its value in $\mathbf{x}'$, leaving $X_4 \wedge X_7 \wedge X_{13}$.
- If X_1 to X_4 or X_{14} to X_n comes first, then X_4 and X_6 come before X_7 in the ordering, leaving $X_7 \wedge X_{13}$.
- If X_8 to X_{12} comes first, then all variables come before X_7, so the conditioned feature is just X_7.

Note that these orderings should not be weighted equally. The weights for the four subfeatures should be multiplied by $1/n$, $2/n$, $(4 + (n - 13))/n$, and $5/n$, respectively.

For symmetry, we can average over all rotations of several different orderings. In our experiments, we average over all rotations of the orderings $[1, 2, \ldots, n]$ and $[n, n-1, \ldots, 1]$. For features of length 2, this is equivalent to summing over all orderings, since all subfeatures are included and the symmetry of the opposite orderings ensures balanced weights.

6 EXPERIMENTS

6.1 DATASETS

We used 12 standard MN structure learning datasets collected and prepared by Davis and Domingos [3], omitting EachMovie since it is no longer publicly available. All variables are binary-valued. The datasets vary widely in size and number of variables. See Table 3 for a summary, and Davis and Domingos [3] for more details on their origin. Datasets are in increasing order by number of variables.

Table 3: Dataset characteristics

Dataset	# Train	# Tune	# Test	# Vars
NLTCS	16,181	2,157	3,236	16
MSNBC	291,326	38,843	58,265	17
KDDCup 2000	180,092	19,907	34,955	64
Plants	17,412	2,321	3,482	69
Audio	15,000	2,000	3,000	100
Jester [6]	9,000	1,000	4,116	100
Netflix	15,000	2,000	3,000	100
MSWeb	29,441	3,270	5,000	294
Book	8,700	1,159	1,739	500
WebKB	2,803	558	838	839
Reuters-52	6,532	1,028	1,540	889
20 Newsgroups	11,293	3,764	3,764	910

6.2 METHODS

On each dataset, we learned DNs with decision tree CPDs and logistic regression CPDs. For decision tree CPDs, each probabilistic decision tree was greedily learned to maximize the conditional likelihood of the target variable given the others. To avoid overfitting, we used the structure prior of Heckerman et al. [8], $P(S) \propto \kappa^f$, where f is the number of free parameters in the model and $\kappa > 0$ is a tunable parameter. Parameters were estimated using add-one smoothing. We learned logistic regression CPDs with L1 regularization using the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) method [1]. Both κ and λ (the L1 regularization parameter) were tuned using held-out validation data. For κ, we started from a value of 10^{-4} and increased it by a factor of 10 until the score on the validation set decreased. For λ, we used the values 0.1, 0.2, 0.5, 1, 2, 5, ..., 1000 and selected the model with the highest score on the validation set. To score a DN on a validation set, we computed the product of the conditional probabilities of each variable according to their respective CPDs.

We converted the DNs to MNs with 6 different choices of base instances and orderings:

1. One ordering; one base instance
2. Two opposite orderings; one base instance
3. One ordering; expectation over all instances
4. Two opposite orderings; expectation over all instances
5. All rotations of one ordering; expectation over all instances
6. All rotations of two opposite orderings; expectation over all instances

The expectation over all instances was done using the marginal distribution of the variables, as estimated using the training data. To be as fair as possible, we include the time to read in the entire training data in our timing results, as well as the time to write the model to disk.

We also converted the DNs to MNs using weight learning, similar to the methods of [15, 12]. We first converted all DN CPDs to conjunctive features and removed duplicate features. Then we learned weights for all features to maximize the PLL of the training data, using a Gaussian prior on the weights to reduce overfitting. The standard deviation of the Gaussians was tuned to maximize PLL on the validation set. With decision tree CPDs, we used standard deviations of 0.05, 0.1, 0.2, 0.5, and 1.0, and the best-performing models always had standard deviations between 0.1 and 0.5. For logistic regression CPDs, a slightly wider range of standard deviations was required, ranging from 0.1 to 10. Optimization was done using the limited memory BFGS algorithm, a quasi-Newton method for convex optimization [11]. Weight learning was terminated after 100 iterations if it had not yet converged.

We did not compare to BLM [3] or the algorithm of Della Pietra et al. [4], since those algorithms have been shown to be less accurate and many orders of magnitude slower than Ravikumar et al. [15] on these datasets [12, 7].

In addition to PLL, we computed conditional marginal log-likelihood (CMLL), a common evaluation metric for Markov networks [3, 10, 12]. CMLL is the sum of the conditional log-likelihood of each variable, given a subset of the others as evidence. In our experiments, we followed the approach of Davis and Domingos [3], who partition the variables into four sets, $\mathbf{Q}_1$ through $\mathbf{Q}_4$. The marginal distribution of each variable is computed using the variables from the other three sets as evidence:

$$\text{CMLL}(\mathbf{X}) = \sum_j \sum_{X_i \in \mathbf{Q}_j} \log P(X_i | \mathbf{X} - \mathbf{Q}_j)$$

Marginals were computed using Gibbs sampling. We used a Rao-Blackwellized Gibbs sampler that adds counts to all states of a variable based on its conditional distribution before selecting the next value for that variable. We found that 100 burn-in and 1000 sampling iterations were sufficient for Gibbs sampling, and additional iterations did not affect the results very much.

All of our learning and inference methods are available in the latest version of the open-source Libra toolkit.[2]

6.3 RESULTS

First, we compare the different approaches to converting inconsistent DNs and compare the accuracy of the resulting MNs to the original DNs. Figure 1 shows the result of converting DNs with both decision tree CPDs (left) and logistic regression CPDs (right). We measure the relative accuracy by comparing the PLL of each MN to the original DN on the test data. In order to better show differences among datasets with different numbers of variables, we show the normalized PLL (NPLL), which divides the PLL by the number of variables in the domain. All graphs are rescaled so that the height of the tallest bar in each cluster is one, with its actual height listed above the graph. With consistent DNs, all methods would be equivalent and all differences would be zero. Since the DNs are inconsistent, converting to an MN may result in an altered distribution that is less accurate on test data.

Empirically, we see that using many orders and summing over all base instances leads to more accurate models. For the decision tree DNs, some of the bars are negative, indicating MNs that are more accurate than their source DNs. This can happen because the conditional distributions in the MN combine several conditional distributions from the original DN, leading to more complex dependencies than a single decision tree. With $2n$ orderings, the converted MN was more accurate on 10 out of 12 datasets. For logistic regression DNs, the converted MNs are very, very close in accuracy, but never better. With $2n$ orderings, the difference in NPLL is less than 0.005 for all datasets, and is less than 0.001 for 8 out of 12.

Table 4 shows the relative accuracy of converting DNs to MNs by learning weights (LW) or by direct conversion with $2n$ orderings and a marginal distribution over base instances (DN2MN). The result of the better performing algorithm is marked in bold. Differences on individual datasets are not always statistically significant, but the overall ranking trend shows the relative peroformance of the two methods. LW and DN2MN do similarly well on tree CPDs: neither PLL nor CMLL was significantly different according to a Wilcoxon signed ranks test. With logistic regression CPDs, DN2MN achieves higher PLL on 10 datasets and CMLL on 9. Both differences are significant at $p < 0.05$ under a two-tailed Wilcoxon signed ranks test.

Table 5 shows the time for converting DNs using our closed-form solution and weight learning. DN2MN is 7 to 250 times faster than weight learning with decision tree CPDs, and 160 to 1200 times faster than weight learning with logistic regression CPDs. These results include tuning time, which is necessary to obtain the best results for weight learning. If all tuning time is excluded, weight learning is 12 times slower with tree CPDs and 49 times slower with logistic regression CPDs, on average.

7 CONCLUSION

DN2MN learns an MN with very similar performance to the original DN, and does so very quickly. With decision tree CPDs, the MN is often more accurate than the original DN. With logistic regression CPDs, the MN is significantly more accurate than performing weight learning as done by Ravikumar et al. [15]. This makes DN2MN competitive with or better than state-of-the-art methods for learning MN structure. Furthermore, DN2MN can exploit many types of conditional distributions, including boosted trees and rule sets, and can easily take advantage of any algorithmic improvements in learning conditional distributions. Another important application of DN2MN is when the model is specified by experts, since conditional distributions could be much easier to specify than a joint distribution.

Acknowledgements

We thank Chloé Kiddon, Jesse Davis, and the anonymous reviewers for helpful comments. This research was partly funded by ARO grant W911NF-08-1-0242, NSF grant IIS-1118050, and NSF grant OCI-0960354. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, NSF, or the U.S. Government.

[2] http://libra.cs.uoregon.edu/

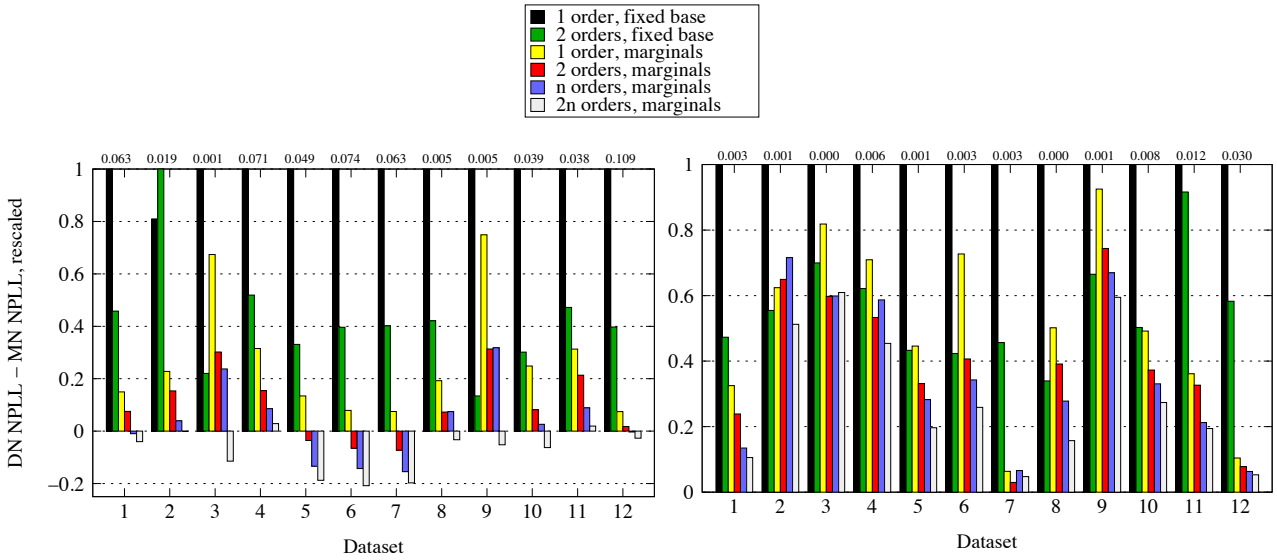


Figure 1: Difference in normalized PLL between the original DNs and converted MNs with different orderings and base instances, divided by largest difference. Results from decision tree DNs are on the left; logistic regression DNs are on the right. Smaller values indicate better MN performance. Largest difference values are listed above each dataset's results, rounded to nearest 1/1000th.

Table 4: Test set PLL and CMLL of converted DNs with tree and logistic regression CPDs.

	Tree CPDs				LR CPDs			
	PLL		CMLL		PLL		CMLL	
Dataset	LW	DN2MN	LW	DN2MN	LW	DN2MN	LW	DN2MN
NLTCS	-5.02	**-4.93**	-5.25	**-5.20**	-4.96	**-4.95**	-5.23	**-5.23**
MSNBC	-4.32	**-4.31**	**-5.75**	-5.80	-6.06	**-6.06**	**-6.28**	-6.28
KDDCup 2000	**-2.05**	-2.05	-2.08	**-2.07**	**-2.06**	-2.07	**-2.11**	-2.12
Plants	**-8.75**	-9.17	**-10.00**	-10.67	**-9.39**	-9.50	**-10.76**	-10.91
Audio	-38.01	**-37.77**	**-38.25**	-38.35	-36.17	**-36.11**	-36.93	**-36.88**
Jester	-51.42	**-50.77**	-51.49	**-51.42**	-49.01	**-48.81**	-49.83	**-49.76**
Netflix	-54.32	**-53.60**	-54.62	**-54.50**	-51.15	**-51.10**	-52.37	**-52.31**
MSWeb	**-8.20**	-8.33	**-8.72**	-8.77	-8.70	**-8.64**	-8.96	**-8.93**
Book	**-34.60**	-35.14	**-34.49**	-35.44	-33.86	**-33.41**	-34.75	**-34.09**
WebKB	-149.37	**-148.42**	**-149.99**	-151.88	-153.13	**-139.39**	-158.51	**-143.05**
Reuters-52	**-82.57**	-82.67	**-82.53**	-85.16	-81.44	**-77.62**	-81.82	**-79.60**
20 Newsgroups	-159.14	**-152.84**	-156.08	**-154.06**	-151.53	**-147.76**	-151.93	**-148.82**

Table 5: Total running time for learning MNs from DNs. DN2MN method uses $2n$ orderings and marginals.

	Tree CPDs			LR CPDs		
Dataset	LW	DN2MN	Speedup	LW	DN2MN	Speedup
NLTCS	4.9s	0.3s	17.3	2.3s	0.01s	162.4
MSNBC	73.3s	9.4s	7.8	7.6s	0.04s	189.6
KDDCup 2000	121.3s	3.8s	31.5	32.0s	0.12s	251.7
Plants	67.9s	1.6s	42.7	71.4s	0.17s	427.6
Audio	147.4s	1.8s	80.3	139.3s	0.40s	344.0
Jester	56.6s	1.2s	46.6	311.7s	0.32s	974.1
Netflix	109.8s	2.0s	54.7	534.3s	0.43s	1245.6
MSWeb	452.8s	16.0s	28.3	308.6s	0.65s	472.0
Book	361.6s	1.4s	252.6	224.8s	1.10s	203.5
WebKB	177.5s	1.8s	128.6	386.3s	1.71s	225.9
Reuters-52	798.2s	3.3s	242.7	951.0s	2.66s	357.7
20 Newsgroups	1952.9s	7.6s	257.9	3830.5s	7.61s	503.0
Geom. mean	149.3s	2.5s	**59.9**	145.4s	0.38s	**363.9**

References

[1] G. Andrew and J. Gao. Scalable training of l1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40. ACM, 2007.

[2] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195, 1975.

[3] J. Davis and P. Domingos. Bottom-up learning of Markov network structure. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, Haifa, Israel, 2010. ACM Press.

[4] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–392, 1997.

[5] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, UK, 1996.

[6] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[7] J. Van Haaren and J. Davis. Markov network structure learning: A randomized feature generation approach. In *Proceedings of the Twenty-Sixth National Conference on Artificial Intelligence*. AAAI Press, 2012.

[8] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[9] G. Hulten, D. M. Chickering, and D. Heckerman. Learning Bayesian networks from dependency networks: A preliminary study. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, FL, 2003.

[10] S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In *Advances in Neural Information Processing Systems 19*, pages 817–824. MIT Press, 2007.

[11] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

[12] D. Lowd and J. Davis. Learning Markov network structure with decision trees. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*, Sydney, Australia, 2010. IEEE Computer Society Press.

[13] D. Lowd and A. Shamaei. Mean field inference in dependency networks: An empirical study. In *Proceedings of the Twenty-Fifth National Conference on Artificial Intelligence*, San Francisco, CA, 2011. AAAI Press.

[14] A. McCallum. Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, Acapulco, Mexico, 2003. Morgan Kaufmann.

[15] P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional ising model selection using L1-regularized logistic regression. *Annals of Statistics*, 2009.

[16] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273–302, 1996.

[17] K. Toutanova, D. Klein, C.D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics, 2003.

[18] E. Yang and P. Ravikumar. On the use of variational inference for learning discrete graphical models. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, Bellevue, WA, 2011. ACM Press.

Bayesian Vote Manipulation: Optimal Strategies and Impact on Welfare

Tyler Lu
Dept. of Computer Science
University of Toronto

Pingzhong Tang
Computer Science Dept.
Carnegie Mellon University

Ariel D. Procaccia
Computer Science Dept.
Carnegie Mellon University

Craig Boutilier
Dept. of Computer Science
University of Toronto

Abstract

Most analyses of manipulation of voting schemes have adopted two assumptions that greatly diminish their practical import. First, it is usually assumed that the manipulators have full knowledge of the votes of the nonmanipulating agents. Second, analysis tends to focus on the probability of manipulation rather than its impact on the social choice objective (e.g., social welfare). We relax both of these assumptions by analyzing *optimal Bayesian manipulation strategies* when the manipulators have only partial probabilistic information about nonmanipulator votes, and assessing the *expected loss in social welfare* (in the broad sense of the term). We present a general optimization framework for the derivation of optimal manipulation strategies given *arbitrary* voting rules and distributions over preferences. We theoretically and empirically analyze the optimal manipulability of some popular voting rules using distributions and real data sets that go well beyond the common, but unrealistic, impartial culture assumption. We also shed light on the stark difference between the loss in social welfare and the probability of manipulation by showing that even when manipulation is likely, impact to social welfare is slight (and often negligible).

1 INTRODUCTION

The use of voting rules to aggregate preferences has become a topic of intense study, and one of great importance in ranking, recommender systems, resource allocation, and other applications of social choice to computational systems. One of the most challenging topics in computational social choice is the study of *manipulation*: given a voting rule, there is usually some set of voter preferences such that one or more voters can obtain a more desirable outcome by misreporting their preferences. Indeed, except under very stringent assumptions, voting rules that are not manipulable do not exist [18, 33]. An important line of research, initiated by Bartholdi *et al.* [2, 3], shows that it can be computationally difficult to manipulate certain rules. However, since worst-case complexity is not itself a significant barrier to manipulation, recent attention has focused on theoretical and empirical demonstration that manipulation is often, or on average, tractable for certain stylized preference distributions [11, 31, 16, 35].

While our understanding of manipulation has improved immensely, some significant deficiencies remain in the state of the art. First, analysis of manipulation is often confined to cases in which the manipulators have complete knowledge of the preferences of sincere voters. While there are reasons for this, such analyses offer too pessimistic a picture of manipulability, since manipulators rarely have access to such information.

The main contribution of our work is a framework for analyzing *optimal Bayesian manipulation strategies* under realistic knowledge assumptions, namely, manipulators with *partial, probabilistic knowledge* of voter preferences. Since success of a manipulation is generally uncertain, the optimal strategy simply maximizes the odds of success. This depends critically on the voting rule, the number of voters, and the preference distribution—as a result, general analytical results are difficult to obtain. We instead present an empirical methodology that, assuming only the ability to sample vote profiles from the preference distribution, allows one to compute an optimal manipulation strategy. We illustrate this methodology on several preference distributions (including real-world data). We also derive sample complexity bounds that provide quality guarantees on the resulting strategies. This framework provides the ability to analyze the manipulability of most voting rules under any probabilistic knowledge assumptions—arbitrary priors, posteriors conditioned on available evidence, even the special case of complete knowledge—requiring only that the preference distribution be sampleable.

While the computational framework is general, we also analytically derive optimal manipulation strategies for

the k-approval rule (and the Borda rule in a limited fashion) under standard *impartial (and anonymous) culture* assumptions. However, since impartial culture is rare in practice [32], these results are primarily of theoretical interest.

A second deficiency of current manipulation analyses pertains to the emphasis on success probability. We adopt a decision-theoretic perspective that provides a more nuanced picture of manipulability of various voting rules. Intuitively, the more preferred an alternative is to the sincere voters, the more likely it is that manipulators can succeed in causing the alternative to be selected. As such, probability of manipulation does not tell the whole story, since alternatives with higher success probability cause less societal dissatisfaction. To this end, we interpret the "score" associated with specific alternatives under a given voting rule as an explicit social choice objective—i.e., a *social welfare function* in the broad sense of the term—and propose analyzing rules in this light. We recognize that voting protocols are often applied in settings where maximizing (some form of) social welfare is not the main objective; but we argue below that this perspective is natural in many applications, and our methodology applies to measures of social welfare different from the "score" used by the voting rule itself.

Along these lines, we derive theoretical bounds on the impact of manipulation on social welfare for the special case of positional scoring rules. More importantly, while distribution-dependent analytical results are difficult to obtain, we can exploit our ability to identify optimal manipulation strategies: by sampling vote profiles from the distribution, computing the loss in welfare under the *optimal* manipulation strategy, and averaging the results, we can readily determine the expected impact on welfare for any specific distribution. Our empirical results show that manipulation for certain rules, under realistic knowledge assumptions, generally causes little damage.

2 BACKGROUND

We briefly review relevant background; see [17, 9] for further detail on (computational) social choice.

2.1 VOTING RULES

We assume a set of n *voters* $N = \{1, \ldots, n\}$ and a set of m *alternatives* $A = \{a_1, \ldots, a_m\}$. A voter i's preferences are represented by a *ranking* or permutation v_i over A. If $v_i(a)$ is the *rank* of a in v_i, we say i prefers a_j to a_k, if $v_i(a_j) < v_i(a_k)$. We refer to v_i as i's *vote*. A *preference profile* is a collection of votes $\mathbf{v} = (v_1, \ldots, v_n)$. Let $\mathcal{V}$ be the set of all such profiles.

A *voting rule* $r : \mathcal{V} \rightarrow A$ selects a "winner" given a preference profile. Common voting rules include plurality, approval, Borda, single transferable vote (STV) and many others. We will use *positional scoring rules* frequently in the sequel. Let $\alpha = (\alpha_1, \ldots, \alpha_m)$ be a non-negative *positional scoring vector*, where $\alpha_i \geq \alpha_{i+1}$ for all $i \geq 1$. Intuitively, a earns score α_i for every vote that places it in position i, and the alternative with the greatest total score wins.[1] More formally, let the $m \times m$ *positional summary matrix (PSM)* $\mathbf{X}$ for profile $\mathbf{v}$ have entry X_{ij} equal to the number of votes that rank a_i in position j. Then the *score* $s_\alpha(a_i; \mathbf{v})$ of a_i is the i-th entry of $\mathbf{X}\alpha'$. Several important rules can be defined using positional scores: plurality by $(1, 0, \ldots, 0)$; k-approval by $(1, 1, \ldots, 0, 0)$, with k 1s; and Borda by $(m-1, m-2, \ldots, 0)$.

Many voting rules—not just positional rules, but rules such as maximin, Copeland, Bucklin, Kemeny, and many others—*explicitly score all alternatives*, assigning a score $s(a, \mathbf{v})$ that defines some measure of the quality of alternative a given a profile $\mathbf{v}$, then choosing the alternative with maximum score. This can be viewed as implicitly defining a *social choice objective* that is optimized by the voting rule, or more broadly as a *social welfare function*, that expresses some "societal utility" for each alternative.[2] While this interpretation may not be valid in all contexts, we will refer to $s(a, \mathbf{v})$ as the *social welfare* of a under rule r for voters with preferences $\mathbf{v}$, and write this as $SW(a, \mathbf{v})$ to emphasize this view. We discuss this further below.

2.2 MANIPULATION

By *manipulation* we refer to a coalition of one or more voters obtaining a more desirable outcome by misreporting their preferences. Indeed, except under very stringent conditions (e.g., single-peaked preferences), the classic Gibbard-Sattherwaite theorem shows that no scheme is immune to manipulation [18, 33]. In other words, there is some preference profile for which at least one voter can obtain a better outcome by misreporting. In the remainder of the paper, we focus on *constructive manipulation*, in which a coalition attempts to cause a single "preferred" candidate to win. We emphasize however that the general principles underlying our approach apply equally well to other forms of manipulation such as destructive manipulation or utility maximization (see below).

[1] In Sec. 3, we assume that tie-breaking works against the desired alternative of the manipulators.

[2] We we use the term "social welfare" in its broadest sense, referring to *any* means of ranking social outcomes. Specifically, we do *not* assume its more restricted definition, commonly used in mechanism design, as "sum of individual voter utilities."

Whether manipulation is a problem in practice depends on: how likely such manipulable preference profiles are; whether manipulators can detect the existence such a profile; and whether computing a suitable misreport is feasible. On this third point, the pioneering work of Bartholdi *et al.* [2, 3] demonstrated that, even given full knowledge of a profile, computing a suitable manipulation is computationally intractable for certain voting rules. This in turn led to the detailed computational analysis of many voting rules (e.g., the Borda rule [13, 4]). Of course, worst-case complexity results offer little comfort if "difficult profiles" are unlikely to arise in practice. Recent work suggests that common voting rules are in fact frequently manipulable, by studying heuristic algorithms that provide theoretical guarantees [31, 41, 39], identifying properties of voting rules that make them easy to manipulate in the typical case [11, 16, 40], and investigating (both theoretically and empirically) the relation between the number of manipulators and the probability of manipulation [30, 38, 35] (see [15] for an overview).

Analyses demonstrating ease of manipulation tend to suffer from two key drawbacks. First, they exclusively analyze manipulation assuming the manipulating coalition has full knowledge of the vote profile. While results showing that manipulation is *difficult* can be justified on these grounds, claiming *easiness* of manipulation has less practical import if the coalition is assumed to have unreasonable access to the preferences of sincere voters.[3] One exception considers manipulators who know only that the vote profile lies within some set [12], but unfortunately this work only analyzes the rather weak notion of *dominating manipulations*. Social choice research on manipulation under probabilistic knowledge is mostly negative in nature [23], or restricted to a single manipulator [1].

A second weakness of many analyses of probability of manipulation (which do assume complete information on the part of the manipulator) is their reliance on specific stylized models such as impartial culture (where every ranking in equally likely) [16, 40]. Much empirical work also considers very stylized distributions such as impartial culture, Polya's urn, and Condorcet (or Mallows) distributions. Some work does consider sub-sampling from real voting data, though with relatively small numbers of votes [36].

2.3 PROBABILISTIC RANKING MODELS

Probabilistic analysis of manipulation—including our Bayesian manipulation problem—requires some probabilistic model of voter preferences. By far the most common model in social choice is *impartial culture (IC)*, which assumes the preference of any voter is drawn from the uniform distribution over the set of permutations of alternatives [32]. A related model is the *impartial anonymous culture (IAC) model* in which each *voting situation* is equally likely [32].[4] Several other models (bipolar, urn, etc.) are considered in both theoretical and empirical social choice research.

Probabilistic models of rankings are widely considered in statistics, econometrics and machine learning as well, including models such as Mallows ϕ-model, Plackett-Luce, and mixtures thereof [25]. We use the *Mallows ϕ-model* [24] in Sec. 6, which is parameterized by a reference ranking σ and a dispersion $\phi \in (0, 1]$, with $P(r) = \frac{1}{Z}\phi^{d(r,\sigma)}$, where r is any ranking, d is Kendall's τ-distance, and Z is a normalizing constant. When $\phi = 1$, this model is exactly the impartial culture model studied widely in social choice—as such it offers considerable modelling flexibility. However, mixtures of Mallows models offer even greater flexibility, allowing (with enough mixture components) accurate modelling of any distribution over preferences. As a consequence, Mallows models, and mixtures thereof, have attracted considerable attention in the machine learning community [27, 8, 20, 26, 21]. We investigate these models empirically below.

3 OPTIMAL BAYESIAN MANIPULATION

We now consider how a manipulating coalition should act given *probabilistic* knowledge of the preferences of the sincere voters. We first formally define our setting, then present several analytical results. Finally, we present a general, sample-based optimization framework for computing optimal manipulation strategies and provide sample complexity results for positional scoring rules and k-approval.

3.1 THE MODEL

We make the standard assumption that voters are partitioned into n sincere voters, who provide their true rankings to a voting mechanism or rule r, and a coalition of c manipulators. We assume the manipulators have a *desired alternative* $d \in A$, and w.l.o.g. we assume $A = \{a_1, \ldots, a_{m-1}, d\}$. We make no assumptions about the manipulators' specific preferences, only that they desire to cast their votes so as to maximize the probability of d winning under r. A vote profile can be partitioned as $\mathbf{v} = (\mathbf{v}_n, \mathbf{v}_c)$, where $\mathbf{v}_n$ reflects the true

[3] This is implicit in [10], which shows that hardness of full-information manipulation implies hardness under probabilistic information.

[4] A voting situation simply counts the *number* of voters who hold each possible ranking of the alternatives.

preferences of the n sincere voters and $\mathbf{v}_c$ the *reported* preferences of the c manipulators.

In contrast to most models, we assume the coalition has only *probabilistic knowledge* of sincere voter preference: a distribution P reflects these *beliefs*, where $P(\mathbf{v}_n)$ is the coalition's degree of belief that the sincere voters will report $\mathbf{v}_n$. We refer to the problem facing the coalition as a *Bayesian manipulation problem*. Manipulator beliefs can take any form: a simple prior based on a standard preference distributions; a mixture model reflecting beliefs about different voter "types;" or a posterior formed by conditioning on evidence the coalition obtains about voter preferences (e.g., through polling, subterfuge, or other means). This latter indeed seems to be the most likely fashion in which manipulation will proceed in practice. Finally, the standard full knowledge assumption is captured by a point distribution that places probability 1 on the actual vote profile. We sometimes refer to P as a distribution over individual *preferences*, which induces a distribution over profiles by taking the product distribution P^n.

The coalition's goal is to cast a collective vote $\mathbf{v}_c$ that maximizes the chance of d winning:

$$\operatorname*{argmax}_{\mathbf{v}_c} \sum_{\mathbf{v}_n:\ r(\mathbf{v}_n,\mathbf{v}_c)=d} P(\mathbf{v}_n).$$

We refer to this $\mathbf{v}_c$ as an *optimal Bayesian manipulation strategy*. For most standard voting rules, this is equivalent to maximizing the *probability of manipulation*, which is the above sum restricted to profiles $\mathbf{v}_n$ such that $r(\mathbf{v}_n) \neq d$.

While we focus on constructive manipulation, our general framework can be applied directly to any reasonable objective on the part of the manipulating coalition. Plausible objectives include: *destructive manipulation*, which attempts to prevent a specific candidate from winning; *safe manipulation*, where a coalitional voter is unsure whether his coalitional colleagues will vote as planned [34, 19]; or *utility maximization*, which attempts to maximize (expected) utility over possible winning candidates. Notice that constructive manipulation can be interpreted as utility maximization with a 0-1 utility for the desired candidate d winning.

3.2 ANALYTICAL RESULTS

Our aim is to determine optimal manipulation strategies given any probabilistic beliefs that the coalition might hold, for arbitrary voting rules. Given this general goal, tight analytical results and bounds are infeasible, a point to which we return below. We do provide here two results for optimal manipulation under impartial (and impartial anonymous) culture. Since this style of analysis under partial information is rare, these results suggest the form that further results (e.g., for additional rules and more general distributions) might take. However, as argued elsewhere [32], this preference model is patently unrealistic, so we view these results as being largely of theoretical interest. Indeed, the difficulty in obtaining decent analytical results even for simple voting rules under very stylized distributions strongly argues for a more general computational approach to manipulation optimization that can be applied broadly—an approach we develop in the next section.

We begin with an analysis of the k-approval rule. When sincere votes are drawn from the uniform distribution over rankings, each alternative will obtain the same number of approvals in expectation. Intuitively, the coalition should cast its votes so that each approves d, and all alternatives apart from d receive the same number of approvals from the coalition (plus/minus 1 if $c(k-1)$ is not divisible by $m-1$): we refer to this as the *balanced strategy*. Indeed, this strategy is optimal:

Theorem 1. *The balanced manipulation strategy is optimal for k-approval under IC and IAC.*[5]

Things are somewhat more complex for the Borda rule, and we provide results only for the case of three candidates under IC and IAC. Apart from the balanced strategy, we use a *near-balanced strategy*, where the coalition's total approval score for d is c, and the scores for the two candidates apart from d differ by at most 2.

Theorem 2. *Let $A = \{x, y, d\}$ be a set of three alternatives, assume c is even. Then the either the balanced strategy or the near-balanced strategy is the optimal manipulation strategy for Borda under both IC and IAC. Furthermore, the balanced strategy is optimal if either: (i) n is even and $c+2$ is divisible by four; or (ii) n is odd and c is divisible by four.*

4 A GENERAL OPTIMIZATION FRAMEWORK

Analytical derivation of optimal Bayesian manipulation strategies is difficult; and even for a *fixed* voting rule, it is not viable for the range of beliefs that manipulators might possess about the voter preferences. For this reason, we develop a general optimization framework that can be used to estimate optimal strategies empirically given only the ability to sample vote profiles from the belief distribution. The model will allow direct estimation of the probability of manipulation

[5]The nontrivial proofs of the results in this section can be found in the appendix of a longer version of this paper; see: `http://www.cs.toronto.edu/~cebly/papers.html`.

(and social cost, see below). The model can be adapted to most voting rules, but we focus our development using positional scoring rules for ease of exposition.

The main idea is straightforward. Suppose we have a sample of T vote profiles from preference distribution P. For each vote profile, a given manipulation will either succeed or not; so we construct an optimization problem, usually in the form of a mixed-integer program (MIP), that constructs the manipulation that succeeds on the greatest number of sampled profiles. If enough samples are used, this approximately maximizes the probability of d winning, or equivalently, the probability of successful manipulation by the coalition. The formulation of the optimization problem—including the means by which one summarizes a sampled vote profile and formulates the objective—depends critically on the voting rule being used. We illustrate the method by formulating the problem for positional scoring rules.

Assume a positional scoring rule using score vector α. A sampled vote profile can be summarized by a *(summary) score vector* $\mathbf{s} = (s_1, \ldots, s_m)$, where s_i is the total score of a_i in that profile; hence we will treat a profile and its score vector interchangeably. Assume T sampled profiles $S = \{\mathbf{s}^1, \ldots, \mathbf{s}^T\}$. A *manipulation strategy* $\mathbf{v}_c$ can be represented by a PSM $\mathbf{X}$, where X_{ij} denotes the number manipulators who rank candidate a_i in jth position. The total score of each candidate for a given profile $\mathbf{s}$ is then $\mathbf{s} + \mathbf{X}\alpha'$.

This strategy representation simplifies the formulation of the optimization problem significantly (by avoiding search over of all possible collections of rankings). Moreover, it is not difficult to recover a set of manipulator votes $\mathbf{v}_c$ that induce any such $\mathbf{X}$, using properties of perfect matchings on c-regular bipartite graphs:

Lemma 3. *A matrix* $\mathbf{X}$ *is the PSM for some manipulation strategy* $\mathbf{v}_c$ *iff* $\mathbf{X} \in \mathbb{N}_{\geq 0}^{m \times m}$ *and* $\mathbf{X1} = \mathbf{X'1} = c\mathbf{1}$.

Our aim then reduces to finding a PSM $\mathbf{X}$ satisfying the above properties such that $\mathbf{X}\alpha'$ maximizes the probability of manipulation. We can recover the optimal manipulation strategy $\mathbf{v}_c^*$ (i.e., a set of c votes) in polynomial time using an algorithm to find c edge-disjoint perfect matchings in a c-regular bipartite graph. Specifically, we construct a bipartite graph with candidates forming one set of nodes and "vote positions" forming the second set. We connect these two sets of nodes with a multi-set of edges, with exactly X_{ij} (duplicate) edges connecting candidate i to position j. We find a perfect matching in this graph to determine one manipulator vote, remove the corresponding edges, and repeat the process (decreasing each row and column sum by one at each iteration).

We formulate the problem of finding an (approximately) optimal Bayesian manipulation strategy as a MIP which constructs a PSM $\mathbf{X}$ maximizing the number of sampled profiles in S on which d wins. We assume, for ease of exposition only, that α has integral entries. First note that in any optimal strategy, $X_{d1} = c$ and $X_{dj} = 0$ for all $j > 1$, which implies $X_{i1} = 0$ for all $a_i \neq d$. Otherwise we require $X_{ij} \in \{0, \ldots, c\}$ and row and column sum constraints:

$$\sum_{j=2}^{m} X_{ij} = c \quad \forall a_i \neq d, \qquad \sum_{\substack{i=1 \\ i \neq d}}^{m} X_{ij} = c \qquad \forall j > 1.$$

We use variables $I_i^t \in [0,1]$ for all $t \leq T, i \neq d$, where $I_i^t = 1$ iff candidate a_i's total score with manipulators is strictly less than d's total score, constrained as:

$$s_d^t + c\alpha_1 - s_i^t - \sum_{j=2}^{m} X_{ij}\alpha_j \geq \alpha_1(n+c)(I_i^t - 1) + 1 \quad \forall t, a_i \neq d. \quad (1)$$

The left-hand side of Eq. (1) is the score difference between d and a_i, bounded (strictly) from below by $-\alpha_1(n+c)$. If it is less than 0, then $I_i^t < 1$; otherwise, I_i^t is unconstrained by Eq. (1), and will take value 1 (due to the maximization objective below). Finally, we use variables $I^t \in \{0,1\}$ to indicate whether d wins under $\mathbf{X}$ on profile $\mathbf{s}^t$, requiring:

$$\sum_{a_i \neq d} I_i^t \geq (m-1)I^t \quad \forall t. \quad (2)$$

If d's score is less than that of some a_i, then the sum in Eq. (2) is smaller than $m-1$, forcing $I^t = 0$ (otherwise the maximization objective will force it to 1). We use the following natural maximization objective:

$$\max_{\mathbf{I},\mathbf{X}} \sum_{t=1}^{T} I_t. \quad (3)$$

If d wins in sample $\mathbf{s}$ prior to manipulation (see below), d still wins after manipulator votes are counted, but we do not consider this to be "successful manipulation." Thus, the estimated *probability of manipulation* (distinct from the probability of d winning) is the MIP objective value less the number of profiles in S where d would have won anyway.

The MIP can be simplified greatly. First notice that d cannot win, even with manipulation, in profile $\mathbf{s}^t$ if:

$$s_d^t + c\alpha_1 \leq c\alpha_m + \max_i s_i^t. \quad (4)$$

Any such profiles—and all corresponding variables and constraints—can be pruned from the MIP. Similarly, we can prune any profile where d wins regardless of the manipulation. This occurs when d wins without manipulator votes or is very close to winning:

$$s_d^t + c\alpha_1 > c\alpha_2 + \max_i s_i^t. \quad (5)$$

This pruning can greatly reduce the size of the MIP in practice, indeed, in expectation by a factor equal to the probability P that a random profile satisfies condition (4) or (5). The MIP has at most a total of $(T+2)m-2$ constraints, $(m-1)^2+T$ integer variables and $T(m-1)$ continuous variables, where T is the number of non-pruned profiles.

While pruning has a tremendous practical impact, the optimal Bayesian manipulation problem for scoring rules remains NP-hard: this follows from the NP-hardness of Borda manipulation with a known profile [13, 4], and the observation that a single known profile corresponds to a special case of our problem.[6]

The remaining question has to do with sample complexity: in order to have confidence in our estimate, how many samples T should we use? Specifically, if we set a PAC target, obtaining an ε-accurate estimate of the probability of d winning with confidence $1-\delta$, the required number of samples depends on the *VC dimension* D_α of the class of boolean-valued functions over vote profiles (or more generally the corresponding score vectors $\mathbf{s}=(s_1,\ldots,s_m)$):

$$\mathcal{F}_\alpha = \{\mathbf{s} \mapsto \mathbf{1}[d \text{ unique max of } \mathbf{X}\alpha' + \mathbf{s}] \mid \forall \mathbf{X}\}.$$

Using known results [14], on counting $|\mathcal{F}_\alpha|$ one obtains $\sup_\alpha D_\alpha \in O(cm\ln(cm)+c^2)$. Standard sample complexity results then apply directly:

Proposition 4. *There exists a constant $C>0$ such that if $T \geq C(cm\ln(cm)+c^2+\ln(1/\delta))/\varepsilon^2$ then for any distribution P, with probability $1-\delta$ over sample S of size T, we have $\hat{q} \leq q^*+\varepsilon$, where q^* is the probability of manipulation of the best strategy, and $\hat{q}$ is the probability of manipulation given the optimal solution to the MIP.*

For specific positional rules, the sample complexity may be smaller. For example, using standard results on compositions of integers, k-approval gives rise to a VC dimension of $D_{\text{kappr}} \leq \log_2 \binom{m+ck-1}{ck-1}$, giving the following sample complexity result:

Proposition 5. *If $T \geq 256(2\log_2 \binom{m+ck-1}{ck-1} + \ln(4/\delta))/\varepsilon^2$ then for any P, with probability $1-\delta$ over sample S of size T, we have $\hat{q} \leq q^*+\varepsilon$, where q^* is the probability of manipulation under the best strategy for k-approval and $\hat{q}$ is the probability of manipulation given the optimal solution to the MIP for k-approval.*

Furthermore, tighter results could be obtained with specific knowledge or constraints on the distribution P. Of course, such sample complexity results are conservative, and in practice good predictions can be realized with far fewer samples. Note also that this sample complexity is only indirectly related to the complexity of the MIP, due to the pruning of (typically, a great many) sampled profiles.

[6] A partial LP relaxation of the MIP may be valuable in practice: allowing entries of $\mathbf{X}$ to be continuous on $[0,1]$ provides an upper bound on (optimal Bayesian) success probability. Our computational experiments did not require this approximation, but it may be useful for larger problems.

5 IMPACT OF MANIPULATION ON SOCIAL WELFARE

As discussed above, characterizing the impact of a manipulating coalition's action solely in terms of its probability of succeeding can sometimes be misleading. This is especially true when one moves away from political domains—where a utility-theoretic interpretation of voting may run afoul of policy, process and fairness considerations—into other settings where voting is used, such as resource allocation, consumer group recommendations, hiring decisions, and team decision making [7]. In such domains, it is often natural to consider the utility that a group member ("voter") derives from the choice or decision that is made for the group as a whole. However, even in "classical" voting situations, most voting protocols are defined using an explicit score, social choice objective, or social welfare function; as such, analyzing the expected loss in this objective due to (optimal) manipulation is a reasonable approach to characterizing the manipulability of different voting rules.

Intuitively, manipulation is more likely to succeed when the desired candidate d is "closer to winning" under a specific voting rule (in the absence of manipulation) than if the candidate is "further from winning." In a (very loose) sense, if candidates that are "closer to winning" are those that are generally ranked more highly by group members, this means that such candidates are generally more desirable. As a consequence, social welfare for alternative d must be close to that of the optimal (non-manipulated) alternative if d has a reasonable chance of winning, which in turn means that the *damage*, or loss in social welfare, caused by manipulation will itself be limited. In this section we formalize this intuition and provide some simple bounds on such damage. We investigate this empirically in the next section.

Assume a voting rule r based on some social welfare measure $SW(a,\mathbf{v})$ over alternatives a and (reported) preference profiles $\mathbf{v}$; hence $r(\mathbf{v}) \in \text{argmax}_a\, SW(a,\mathbf{v})$. As above, we partition the vote profile: $\mathbf{v}=(\mathbf{v}_n,\mathbf{v}_c)$. We are interested in the loss in social welfare, or *regret*, imposed on the honest voters by a manipulation, so

define this to be:

$$R(\mathbf{v}_n, \mathbf{v}_c) = SW(r(\mathbf{v}_n), \mathbf{v}_n) - SW(r(\mathbf{v}), \mathbf{v}_n).^7 \quad (6)$$

The *expected regret* of a specific manipulation, given distribution P over preference profiles $\mathbf{v}_n$, is then:

$$ER(P, \mathbf{v}_c) = \mathop{\mathbb{E}}_{\mathbf{v}_n \sim P}[R(\mathbf{v}_n, \mathbf{v}_c)] \ . \quad (7)$$

Notice that any social welfare function SW that determines the quality of a candidate a given the sincere vote profile $\mathbf{v}_n$ can be used in Eq. 6; we need not commit to using r's scoring function itself. However, assessing loss does require the use of *some* measure of societal utility or welfare. If one is concerned only with whether the "true winner" is selected, then probability of manipulation, as discussed above, is the only sensible measure.

We illustrate the our framework by deriving several bounds on loss due to manipulation using positional scoring rules to measure social welfare.[8] We can derive theoretical bounds on expected regret for positional scoring rules. First, notice that expected regret can be bounded for arbitrary distributions:

Proposition 6. *Let r be a positional scoring rule with score vector α. Then for any distribution P, and any optimal manipulation strategy $\mathbf{v}_c$ w.r.t. P, we have*

$$\begin{aligned} ER(P, \mathbf{v}_c) < c[&(\alpha_1 - \alpha_m)P(r(\mathbf{v}_n) \neq d \wedge r(\mathbf{v}) = d) \\ &+ (\alpha_2 - \alpha_m)P(r(\mathbf{v}_n) \neq r(\mathbf{v}) \wedge r(\mathbf{v}) \neq d)]. \end{aligned} \quad (8)$$

Intuitively, this follows by considering the maximum increase in score the manipulating coalition can cause for d relative to an alternative a that would have won without the manipulators.

Proof of Prop. 6. Consider any $\mathbf{v}_n$. Clearly, d is ranked first by all votes in $\mathbf{v}_c$. Case 1: if d wins in $\mathbf{v}_n$ then d also wins on $\mathbf{v}$. Case 2: if a_i wins in $\mathbf{v}_n$ but d wins on $\mathbf{v}$ then $SW(a_i) + \alpha_m c \leq SW(a_i) + SW(a_i, \mathbf{v}_c) < SW(d) + \alpha_1 c$ implying $R(\mathbf{v}_n, \mathbf{v}_c) < c(\alpha_1 - \alpha_m)$. Case 3: if a_i wins in $\mathbf{v}_n$ and $a_j \neq a_i$ wins in $\mathbf{v}_c$ then $SW(a_i) + \alpha_m c \leq SW(a_i) + SW(a_i, \mathbf{v}_c) \leq SW(a_j) + \alpha_2 c$ implying $R(\mathbf{v}_n, \mathbf{v}_c) \leq c(\alpha_2 - \alpha_m)$. Case 4: if $r(\mathbf{v}_n) = r(\mathbf{v}_c)$ then regret is zero. Summing 2 and 3 gives the upper bound on expected regret. □

The proposition applies when P reflects full knowledge of $\mathbf{v}_n$ as well; and while this P-dependent bound will be crude for some P, it is in fact tight in the worst-case (which includes full knowledge distributions):

Proposition 7. *Suppose $\alpha_1 + \cdots + \alpha_m = M$, $m - 1$ divides c and $n - c \geq 0$ is even. Then*

$$\sup_{n,c,\alpha} \sup_{P} ER(P, \mathbf{v}_c) = cM. \quad (9)$$

Proof. The upper bound on the LHS follows from the RHS of Eq. 8 since it is at most $c(\alpha_1 - \alpha_m) \leq cM$. For the lower bound on the LHS, let P be a point mass on $\{\mathbf{v}_n\}$: in $\mathbf{v}_n$, the first c votes rank a_1 first and the remaining alternatives $a_2, \ldots, a_{m-1}, p$ in such a way that the number of times any is ranked i-th ($i \geq 2$) is $c/(m-1)$. Of the remaining $n-c$ votes, half rank a_1 first and d second, and half do the opposite (the remaining candidates are ranked in any manner). Thus $SW(a_1) - SW(d) = (\alpha_1 - \frac{\alpha_2 + \cdots + \alpha_m}{m-1})c$. Let $\alpha_2 = \delta + \xi$ and $\alpha_i = \delta$, for some $\delta, \xi > 0$, for all $i \geq 3$. One optimal strategy $\mathbf{v}_c$ is to always place d first and a_1 last, resulting in a score difference of $c(\alpha_1 - \alpha_m) = c(\alpha_1 - \delta)$ which is strictly larger than the above social welfare difference of $c(\alpha_1 - \delta - \xi/(m-1))$ within $\mathbf{v}_n$. Hence $\mathbf{v}_c$ causes d to win, inducing regret of $c(M - (m-1)\delta - \xi m/(m-1))$, which can be made arbitrarily close to cM using a small enough δ, ξ. □

We can obtain a tighter bound than that offered by Prop. 6 if, for a given P and r, we know the optimal manipulation strategy. For instance, exploiting Thm. 1 we obtain:

Proposition 8. *Consider the k-approval rule, where $SW(a, \mathbf{v}_n)$ is the approval score of a. Let P be impartial culture. Then*

$$\begin{aligned} ER(\mathbf{v}_n, BAL) \leq \left[\left\lceil \frac{ck}{m} \right\rceil - 1\right] \cdot [&P(r(\mathbf{v}_n) \neq d \wedge r(\mathbf{v}) = d) \\ &+ P(r(\mathbf{v}_n) \neq r(\mathbf{v}) \wedge r(\mathbf{v}) \neq d)]. \end{aligned} \quad (10)$$

Proof. Consider any $\mathbf{v}_n$. We use a case analysis similar to that in Prop. 6. Case 1 applies directly. For case 3, a_i must have received one more veto vote than a_j from the manipulators, and thus $SW(a_i) - SW(a_j) \leq 1$. For case 2, BAL implies that $SW(a_i) - \lceil \frac{ck}{m} \rceil + 1 \leq SW(p)$ (a_i might have received $\lfloor ck/m \rfloor$ veto votes, but the bound holds in any case). Thus $SW(a_i) - SW(d) \leq \lceil \frac{ck}{m} \rceil - 1$. Case 4 also applies directly. Summing cases 2 and 3 gives the required inequality. □

[7] We assume a voting rule and welfare measure that can accept variable numbers of voters, as is typical.

[8] One reason to consider positional scoring rules like Borda in analyzing impact on social welfare is the tight connection between scoring rules and social welfare maximization in its narrow sense (i.e., sum of individual utilities). In models where we desire to maximize sum of utilities relative to some underlying utility profile, voting is a relatively simple and low-cost way (i.e., with minimal communication) of eliciting partial preference information from voters. Analysis of the *distortion* of utilities induced by restricting voters to expressing ordinal rankings shows that, with carefully crafted positional rules, one can come close to maximizing (this form of) social welfare [37, 29, 6]. Borda scoring, in particular, seems especially robust in this sense.

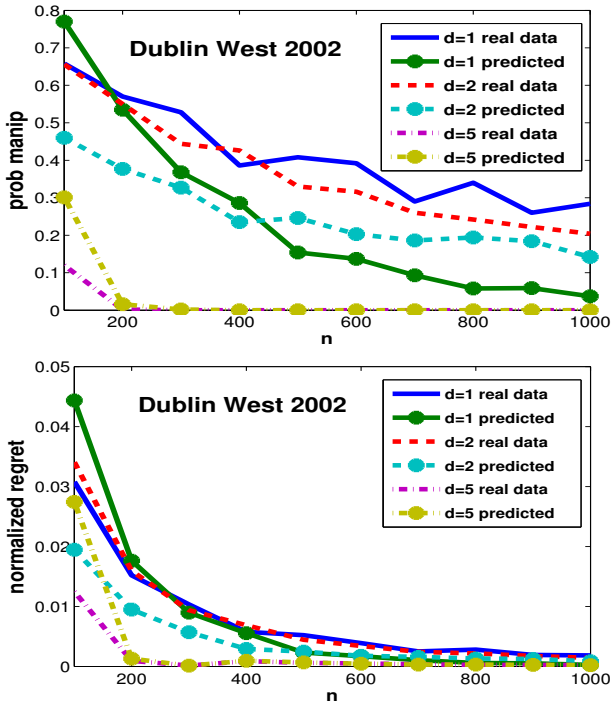


Fig. 1: Probability of manipulation and expected normalized regret for Irish election data.

We can also exploit our empirical optimization framework to assess the expected regret. While deriving optimal manipulation strategies is analytically intractable in general, we can exploit the ability to empirically determine (approximately) optimal Bayesian strategies to great effect. Given a collection of samples, we can—in the same MIP used to compute the optimal strategy—measure expected regret empirically.[9] Sample complexity results related to those above can be derived (we omit details). Notice the ability to accurately estimate the behavior of the manipulators is critical to being able to estimate expected regret.

6 EMPIRICAL EVALUATION

We experiment with several common preference distributions, as well as real voting data, to test the effectiveness of our approach. We are primaily interested in: (a) determining the computational effectiveness of our empirical framework for computing optimal manipulation strategies when manipulators have incomplete knowledge; and (b) measuring both the probability of manipulation and expected regret caused by manipulation in some prototypical scenarios. We focus on Mallows models, and mixtures of Mallows models, because of their flexibility and ability to represent a wide class of preferences (including impartial culture).

[9]Pruning of samples must be less aggressive when estimating regret, since damage may be caused by manipulation even in profiles where d cannot win.

However, it should be clear that our framework is not limited to such models. Space precludes a systematic investigation of multiple voting rules, so we focus here on the Borda rule.

Our first set of experiments uses 2002 Irish electoral data from the Dublin West constituency, with 9 candidates and 29,989 ballots of top-t form, of which 3800 are complete rankings.[10] We learn a Mallows mixture with three components (using the techniques of [21]) using a subsample of 3000 random (not necessarily complete) ballots: this represents a prior that manipulators might develop with intense surveying. We fix $c = 10$ manipulators and vary the number of sincere voters $n \in \{100, 200, \ldots, 1000\}$. The MIP is solved using 500 random profiles sampled from the learned Mallows mixture, which provides the manipulators with an approximately optimal strategy, as well as predicted probability of success and expected regret. We test the predictions by simulating the manipulators' strategy on the real data set, drawing 1000 random profiles (of the appropriate size n) from the set of 3800 full rankings to estimate true probability of manipulation and expected regret (normalized by $SW(r(\mathbf{v}_n), \mathbf{v}_n)$). Since manipulability varies greatly with the "expected rank" of a candidate, we show results for the candidates whose *expected* ranks in the learned model are first, second, and fifth (below this, probability of manipulation is extremely small).

Fig. 1 shows that the probability of manipulation is in fact quite high when d is the first- or second-ranked candidate (in expectation), but is much smaller when d is the fifth-ranked. Not surprisingly, probabilities gradually drop as n grows. The predicted probabilities based on the learned model are reasonably accurate, but of course have some error due to to imperfect model fit. Despite the high probability of manipulation, the second plot shows that expected regret (normalized to show percentage loss) is in fact extremely small. Indeed, maximal (average) loss in social welfare is just over 3%, when d is candidate 2 and $n = 100$, which means means nearly 10% of the voters are manipulators. Expected regret drops rapidly with increasing n. Notice that success probability and expected regret are greatest when the manipulators' desired candidate has expected rank 1 or 2: while the odds of 1 winning are higher than 2, 1 is also more likely to win without manipulator intervention.

Our second set of experiments use Mallows models over six alternatives with different variance ϕ (recall that with $\phi = 1$, Mallows is exactly IC). The reference ranking is $\sigma = 123456$ (i.e., alternative 1 is most preferred, 2 next most, etc.). We fix $c = 10$ and vary n from 100

[10]See www.dublincountyreturningofficer.com.

ϕ	d	n:100	200	300	400	500	600
.6	1	.03	.00	0	0	0	0
	2	.46	.06	.00	0	0	0
.8	1	.19	.09	.05	.03	.01	.01
	2	.68	.41	.25	.13	.08	.05
	3	.41	.07	.01	0	0	0
1	*	.46	.34	.26	.21	.17	.17

ϕ	d	n:100	200	300	400	500
.6	1	5.3E-4	1.7E-5	0	0	0
	2	3.2E-2	2.1E-3	7.1E-5	0	0
.8	1	5.5E-3	1.7E-3	8.0E-4	2.2E-4	7.5E-5
	2	4.0E-2	1.4E-2	6.0E-3	2.5E-3	1.3E-3
	3	9.6E-3	7.6E-4	7.4E-5	1.4E-5	1.2E-5
1	*	1.9E-2	7.1E-3	3.6E-3	2.5E-3	1.5E-3

Fig. 2: Prob. of manipulation (top) and expected normalized regret (bottom), Mallows models.

sec.	n:100	200	300	400	500	600
avg	28.67	0.35	0.20	0.18	0.06	0.07
max	205.64	2.56	1.17	1.16	0.12	0.15

sec.	d	ϕ:0.5	0.6	0.7	0.8	0.9	1
avg	2	0.02	0.04	0.02	0.02	.05	2.44
	4	0.02	0.03	0.02	0.02	0.14	
max	2	0.03	0.15	0.03	0.07	0.14	30.22
	4	0.03	0.08	0.05	0.04	1.16	

Fig. 3: MIP solution times for Dublin (top) and Mallows (bottom) on an 8-core 2.66GHz/core machine.

to 1000 as above. Results in Fig. 2 show manipulation probability and expected regret as we vary ϕ and consider desired alternatives 1, 2 and 3.[11] While manipulation probability is high for these near-top alternatives (when n is small), expected regret (normalized in percentage terms) is negligible, with a maximum of 4%, and then only when the distribution is close to impartial culture ($\phi = 0.8$) and $n = 100$ (nearly 10% manipulators). As above, when manipulators want $d = 2$, expected regret is highest. Of some interest is the connection to both theoretical and empirical work that shows phase transitions often occur when the number of manipulators is roughly the square root of the number of sincere voters: any less makes manipulation very unlikely, while any more makes manipulation likely. While most of this work analyzes complete information settings, our results above show that *with realistic preference distributions*—even with restricted knowledge on the part of manipulators—the probability of manipulation is sometimes quite significant with far fewer manipulators than suggested by past work. Despite this, expected regret remains relatively small.

Fig. 3 shows the average and maximum running times of the MIP required to compute the optimal manipulation for the problems described above. As can be seen, even with a large number of sampled profiles, the MIP can be solved quickly across a range of problem sizes and distributions.

[11]Under IC (i.e., when $\phi = 1$), alternatives are probabilistically indistinguishable, so we show one row only.

7 CONCLUDING REMARKS

Our primary contribution is an empirical framework for the computation of optimal manipulation strategies when manipulators have incomplete information about voter preferences. This is an important methodology for the analysis of the manipulability of voting rules in realistic circumstances, without the need to restrict the analysis to specific voting rules or priors. Our experiments indicate that our algorithms are quite tractable. Furthermore, our results suggest that manipulation may not be as serious a problem as is commonly believed when realistic informational models are used, or when the quality of the outcome, rather than societal justice, is the main objective. Our empirical results, which exploit several innovations introduced in this paper, demonstrate this in the case of Borda voting; but our approach is easily adapted to different types of manipulation under different scoring rules, and can be applied to any "utility-based" voting rule with appropriate formulation of the optimization problem. Thus, our approach provides a compelling framework for the comparison of voting rules.

One nonstandard aspect of our approach is the use of the score under a voting rule as a proxy for social welfare. A similar regret-based approach is taken in preference elicitation [22]; and it is implicit in work on approximating voting rules [28, 5], which assumes that approximating the score also gives an approximation to the desirability of an alternative. One strong argument in favor of this view is that scores of certain voting rules, such as Borda, are provably good proxies for utilitarian social welfare when utilities are drawn from specific distributions [37]. That said, our framework can be used, in principle, to analyze any measure of impact or loss.

Our work suggests a number of interesting future directions. Of course, we must study additional voting rules, social welfare measures, and manipulator objectives within our framework to further demonstrate its viability. While our results suggest that incomplete knowledge limits the ability of a manipulating coalition to impact the results of an election, our framework can also be used to directly study the relationship between the "amount of information" (e.g., using entropy or equivalent sample size metrics) and the probability of manipulation. Finally, interesting computational questions arise within our approach: e.g., deriving tighter complexity results for optimal manipulation given a *collection* of vote profiles; or deriving simpler classes of manipulation policies (e.g., uncertainty-sensitive variants of the balanced manipulation strategy) that can be more readily optimized by a manipulating coalition.

References

[1] M. Á. Ballester and P. Rey-Biel. Does uncertainty lead to sincerity? Simple and complex voting mechanisms. *Social Choice and Welfare*, 33(3):477–494, 2009.

[2] John Bartholdi III, Craig Tovey, and Michael Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[3] John J. Bartholdi III and James B. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[4] Nadja Betzler, Rolf Niedermeier, and Gerhard J. Woeginger. Unweighted coalitional manipulation under the Borda rule is NP-hard. In *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 55–60, Barcelona, Spain, 2011.

[5] E. Birrell and R. Pass. Approximately strategy-proof voting. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 67–72, 2011.

[6] Craig Boutilier, Ioannis Caragiannis, Simi Haber, Tyler Lu, Ariel D. Procaccia, and Or Sheffet. Optimal social choice functions: A utilitarian view. In *Proceedings of the Thirteenth ACM Conference on Electronic Commerce (EC'12)*, pages 197–214, Barcelona, 2012.

[7] Craig Boutilier and Tyler Lu. Probabilistic and utility-theoretic models in social choice: Challenges for learning, elicitation, and manipulation. In *IJCAI Workshop on Social Choice and Artificial Intelligence*, pages 7–9, Barcelona, 2011.

[8] Ludwig M. Busse, Peter Orbanz, and Joachim M. Buhmann. Cluster analysis of heterogeneous rank data. In *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML-07)*, pages 113–120, Corvallis, OR, 2007.

[9] Yann Chevaleyre, Ulle Endriss, Jérôme Lang, and Nicolas Maudet. A short introduction to computational social choice. In *Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM-07)*, pages 51–69, Harrachov, Czech Republic, 2007.

[10] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):1–33, 2007.

[11] Vincent Conitzer and Tuomas Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, pages 627–634, Boston, 2006.

[12] Vincent Conitzer, Toby Walsh, and Lirong Xia. Dominating manipulations in voting wih partial information. In *Proceedings of the Twenty-fifth AAAI Conference on Artificial Intelligence (AAAI-11)*, pages 638–643, San Francisco, 2011.

[13] Jessica Davies, George Katsirelos, Nina Narodytska, and Toby Walsh. Complexity of and algorithms for Borda manipulation. In *Proceedings of the Twenty-fifth AAAI Conference on Artificial Intelligence (AAAI-11)*, pages 657–662, San Francisco, 2011.

[14] C.J. Everett and P.R. Stein. The asymptotic number of integer stochastic matrices. *Discrete Mathematics*, 1(1):55–72, 1971.

[15] P. Faliszewski and A. D. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[16] Ehud Friedgut, Gil Kalai, and Noam Nisan. Elections can be manipulated often. In *Proceedings of the 49th Annual IEEE Symposium on the Foundations of Computer Science (FOCS'08)*, pages 243–249, Philadelphia, 2008.

[17] Wulf Gaertner. *A Primer in Social Choice Theory*. LSE Perspectives in Economic Analysis. Oxford University Press, USA, August 2006.

[18] Allan Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, 1973.

[19] Noam Hazon and Edith Elkind. Complexity of safe strategic voting. In *Symposium on Algorithmic Game Theory (SAGT-10)*, pages 210–221, 2010.

[20] Guy Lebanon and Yi Mao. Non-parametric modeling of partially ranked data. *Journal of Machine Learning Research*, 9:2401–2429, 2008.

[21] Tyler Lu and Craig Boutilier. Learning Mallows models with pairwise preferences. In *Proceedings of the Twenty-eighth International Conference on Machine Learning (ICML-11)*, pages 145–152, Bellevue, WA, 2011.

[22] Tyler Lu and Craig Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proceedings of the Twenty-second In-*

ternational Joint Conference on Artificial Intelligence (IJCAI-11), pages 287–293, Barcelona, Spain, 2011.

[23] D. Majumdar and A. Sen. Bayesian incentive compatible voting rules. *Econometrica*, 72(2):523–540, 2004.

[24] Colin L. Mallows. Non-null ranking models. *Biometrika*, 44:114–130, 1957.

[25] John I. Marden. *Analyzing and Modeling Rank Data*. Chapman and Hall, 1995.

[26] Marina Meila and Harr Chen. Dirichlet process mixtures of generalized Mallows models. In *Proceedings of the Twenty-sixth Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 358–367. AUAI Press, 2010.

[27] Thomas Brendan Murphy and Donal Martin. Mixtures of distance-based models for ranking data. *Computational Statistics & Data Analysis*, 41(3-4):645–655, 2003.

[28] A. D. Procaccia. Can approximation circumvent Gibbard-Satterthwaite? In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 836–841, 2010.

[29] A. D. Procaccia and J. S. Rosenschein. The distortion of cardinal preferences in voting. In *Proceedings of the 10th International Workshop on Cooperative Information Agents, LNAI 4149*, pages 317–331. Springer, 2006.

[30] A. D. Procaccia and J. S. Rosenschein. Average-case tractability of manipulation in elections via the fraction of manipulators. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 718–720, 2007.

[31] A. D. Procaccia and J. S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of Artificial Intelligence Research*, 28:157–181, 2007.

[32] Michel Regenwetter, Bernard Grofman, A. A. J. Marley, and Ilia Tsetlin. *Behavioral Social Choice: Probabilistic Models, Statistical Inference, and Applications*. Cambridge University Press, Cambridge, 2006.

[33] Mark A. Satterthwaite. Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, April 1975.

[34] Arkadii Slinko and Shaun White. Nondictatorial social choice rules are safely manipulable. In *Proceedings of the Second International Workshop on Computational Social Choice (COMSOC-2008)*, pages 403–414, 2008.

[35] Toby Walsh. Where are the really hard manipulation problems? the phase transition in manipulating the veto rule. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 324–329, Pasadena, California, 2009.

[36] Toby Walsh. An empirical study of the manipulability of single transferable voting. In *Proceedings of the Nineteenth European Conference on Artificial Intelligence (ECAI-10)*, pages 257–262, Lisbon, 2010.

[37] R. J. Weber. Reproducing voting systems. Cowles Foundation Discussion Paper No. 498, 1978.

[38] L. Xia and V. Conitzer. Generalized scoring rules and the frequency of coalitional manipulability. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC)*, pages 109–118, 2008.

[39] L. Xia, V. Conitzer, and A. D. Procaccia. A scheduling approach to coalitional manipulation. In *Proceedings of the 11th ACM Conference on Electronic Commerce (EC)*, pages 275–284, 2010.

[40] Lirong Xia and Vincent Conitzer. A sufficient condition for voting rules to be frequently manipulable. In *Proceedings of the Ninth ACM Conference on Electronic Commerce (EC'08)*, pages 99–108, Chicago, 2008.

[41] Michael Zuckerman, Ariel D. Procaccia, and Jeffrey S. Rosenschein. Algorithms for the coalitional manipulation problem. *Artificial Intelligence*, 173(2):392–412, 2009.

Heuristic Ranking in Tightly Coupled Probabilistic Description Logics

Thomas Lukasiewicz[1] **Maria Vanina Martinez**[1] **Giorgio Orsi**[1,2] **Gerardo I. Simari**[1]
[1] Department of Computer Science, University of Oxford, UK
[2] Institute for the Future of Computing, Oxford Martin School, University of Oxford, UK
email: firstname.lastname@cs.ox.ac.uk

Abstract

The Semantic Web effort has steadily been gaining traction in the recent years. In particular, Web search companies are recently realizing that their products need to evolve towards having richer semantic search capabilities. Description logics (DLs) have been adopted as the formal underpinnings for Semantic Web languages used in describing ontologies. Reasoning under uncertainty has recently taken a leading role in this arena, given the nature of data found on the Web. In this paper, we present a probabilistic extension of the DL $\mathcal{EL}^{++}$ (which underlies the OWL2 EL profile) using Markov logic networks (MLNs) as probabilistic semantics. This extension is tightly coupled, meaning that probabilistic annotations in formulas can refer to objects in the ontology. We show that, even though the tightly coupled nature of our language means that many basic operations are data-intractable, we can leverage a sublanguage of MLNs that allows to rank the atomic consequences of an ontology relative to their probability values (called ranking queries) even when these values are not fully computed. We present an anytime algorithm to answer ranking queries, and provide an upper bound on the error that it incurs, as well as a criterion to decide when results are guaranteed to be correct.

1 Introduction

Recently, it has become apparent that Semantic Web formalisms must be able to cope with uncertainty in a principled manner. The Web contains many examples where uncertainty comes in [22]: as an inherent aspect of Web data (such as in reviews of products or services, comments in blog posts, weather forecasts, etc.), as the result of automatically processing Web data (for instance, analyzing a document's HTML Document Object Model usually involves some degree of uncertainty), and as the result of integrating information from many different heterogeneous sources (such as in aggregator sites, which allow users to query multiple sites at once to save time). Finally, inconsistency and incompleteness are also ubiquitous as the result of over- and under-specification, respectively. To be applicable to Web-sized data sets, any machinery developed for dealing with uncertainty in these settings must be scalable. In this paper, we develop an extension of $\mathcal{EL}^{++}$ [1] by means of a probabilistic semantics based on Markov logic networks [20]. $\mathcal{EL}^{++}$ is a DL that combines tractability of several key reasoning problems with enough expressive power to model a variety of ontologies; for instance, it is expressive enough to model real-world ontologies such as the well-known SNOMED CT, large segments of the Galen medical knowledge base, as well as the Gene Ontology. Moreover, $\mathcal{EL}^{++}$ underlies the OWL2 EL profile, in which basic reasoning problems are solvable in polynomial time, and highly scalable implementations are available. One of the key aspects of the extension presented here is that it is *tightly coupled*, meaning that probabilistic annotations can refer to objects in the ontology, providing greater expressive power than similar efforts in the literature.

In the area of Web data extraction [16], uncertainty comes into play even for very basic tasks. Consider, as an example, the *form-labeling* problem, consisting of the association of blocks of text (i.e., the *labels*) to the corresponding *fields* in a Web form [6], illustrated in Figure 1, where three possible instances of the problem with decreasing likelihood of occurrence are shown. In the first case, the fields f_1 and f_2 have horizontally aligned blocks of text on their left and on their right; this case exemplifies the most typical situation where the blocks of text on the left of a field (denoted l_1 and l_2) are the actual labels, while those on the right (denoted t_1 and t_2) are either unrelated or carry additional information about the fields. The second case represents the symmetric situation, where the labels are on the right of the fields. This situation is typical for eastern Web sites where the content has right-to-left reading order (e.g., for Arabic). Another possibility is exemplified by the third case, where the labels occur in the north-west region of the field. In this

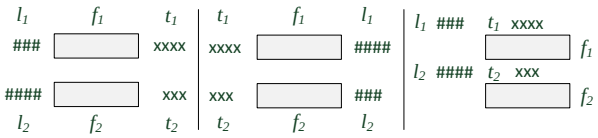


Figure 1: Uncertainty in form labeling.

setting, probabilistic DL formalisms are well-suited to the task, since they couple powerful modeling capabilities with sound and complete reasoning procedures.

Example 1. Consider the following $\mathcal{EL}^{++}$ formulas describing a simple ontology of Web forms. The ontology forces each field to be associated with a block of text representing its label. Fields and text blocks are disjoint sets.

$$\mathit{Field} \sqsubseteq \exists \mathit{label}.\mathit{Text}; \quad \mathit{dom}(\mathit{label}) \sqsubseteq \mathit{Field};$$
$$\mathit{Field} \sqcap \mathit{Text} \sqsubseteq \bot; \quad \mathit{ran}(\mathit{label}) \sqsubseteq \mathit{Text}. \qquad \blacksquare$$

The set of formulas above simply defines what a correct labeling of the form should be, without providing a measure of the likelihood that the labeling is correct. Example 1 shows the need for probabilistic modeling languages for reasoning over structured data on the Web, e.g., to leverage the statistical evidence that certain patterns are more likely to occur than others.

Markov logic networks [20] (MLNs), which were developed in recent years, are a simple approach to generalizing classical logic; their relative simplicity and lack of restrictions has recently caused them to be well-received in the reasoning under uncertainty community. To our knowledge, this work is the first to develop an MLN-based probabilistic extension of DLs, and in particular of $\mathcal{EL}^{++}$.

The following are the main contributions of this paper:

(i) Introduction of tightly coupled probabilistic (TCP) DLs, an expressive class of probabilistic ontology languages that allow probabilistic annotations to refer to objects in the ontology. The probabilistic semantics is based on MLNs.

(ii) Complexity results establishing the intractability of basic computations over TCP ontologies necessary to rank atomic consequences based on their probabilities.

(iii) The proposal of *conjunctive MLNs* (cMLNs), a subset of the MLN formalism. Though we prove that this model restriction does not alleviate the complexity issues of the general case for TCP ontologies, we propose the analysis of a special kind of *equivalence classes* of possible worlds, for which we prove useful properties: deciding emptiness and generation of members can both be done in polynomial time in the data complexity, all worlds in a class are guaranteed to have the same probability, and the highest-probability classes can be identified in polynomial time in the data complexity.

(iv) An anytime algorithm for answering *ranking queries* over TCP ontologies with cMLNs that leverages the properties of equivalence classes and is guaranteed to inspect worlds in decreasing order of probability. We provide an upper bound on the error that is incurred by this algorithm based on the number of worlds and equivalence classes inspected, and a criterion to decide when results are guaranteed to be correct.

The rest of this paper is organized as follows. Section 2 describes the preliminaries on the DL $\mathcal{EL}^{++}$ and MLNs. Section 3 presents tightly coupled probabilistic DLs, and a complexity result showing the intractability of computing probabilities of atoms. In Section 4, we present conjunctive MLNs, and we analyze how their structure can be leveraged in the definition of well-behaved equivalence classes over possible worlds, as well as in an anytime algorithm that exploits this equivalence class approach in order to tractably compute a heuristic answer to the ranking of atoms according to their probabilities. Finally, Sections 5 and 6 discuss related work and conclusions, respectively.

2 Preliminaries

In this section, we briefly recall the description logic (DL) $\mathcal{EL}^{++}$ and Markov logic networks (MLNs).

2.1 The DL $\mathcal{EL}^{++}$

We now recall the syntax and the semantics of $\mathcal{EL}^{++}$, a tractable DL especially suited for representing large amounts of data. Intuitively, DLs model a domain of interest in terms of concepts and roles, which represent classes of individuals and binary relations between individuals, respectively. While we restrict ourselves to $\mathcal{EL}^{++}$ here, the general approach continues to be valid for any DL or ontology language for which instance checking is data-tractable (for instance, the closely related Datalog+/– family of ontology languages contains such tractable subsets [3]). However, note that all results in this paper were derived for $\mathcal{EL}^{++}$, and thus certain results may not hold for other logics.

Syntax and Semantics. We first define concepts and then knowledge bases and instance checking in $\mathcal{EL}^{++}$. We assume pairwise disjoint sets $\mathbf{A}$, $\mathbf{R}$, and $\mathbf{I}$ of *atomic concept names*, *role names*, and *individual names*, respectively. Concepts are defined inductively via the constructors shown in the first five rows of the table in Figure 2; this table adopts the usual conventions of using C and D to refer to concepts, r to refer to a role, and a and b to refer to individuals. The semantics of these concepts is given, as usual in first-order logics, in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The *domain* $\Delta^{\mathcal{I}}$ comprises a non-empty set of individuals and the *interpretation function* $\cdot^{\mathcal{I}}$ maps each concept name $A \in \mathbf{A}$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each role name $r \in \mathbf{R}$ to a binary relation $r^{\mathcal{I}}$ over $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name $a \in \mathbf{I}$ to an individual $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The ex-

Name	Syntax	Semantics
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Existential Restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
Concrete Domain	$p(f_1,...,f_k)$ for $p \in \mathcal{P}^{\mathcal{D}_j}$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y_1,...,y_k \in \Delta^{\mathcal{D}_j} : f_i^{\mathcal{I}}(x) = y_i, 1 \leq i \leq k \wedge (y_1,...,y_k) \in p^{\mathcal{D}_j}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RI	$r_1 \circ ... \circ r_k \sqsubseteq r$	$r_1^{\mathcal{I}} \circ ... \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$
Domain Restriction	$dom(r) \sqsubseteq C$	$r^{\mathcal{I}} \subseteq C^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Range Restriction	$ran(r) \sqsubseteq C$	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times C^{\mathcal{I}}$
Concept Assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Role Assertion	$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Figure 2: Syntax and Semantics of $\mathcal{EL}^{++}$ (rep. from [1]).

tension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is defined via the constructors in Figure 2.

Reference to concrete data objects is accomplished via the parameterization of $\mathcal{EL}^{++}$ by concrete domains $\mathcal{D}_1, ..., \mathcal{D}_n$, which correspond to OWL data types. Such concrete domains are pairs $(\Delta^{\mathcal{D}}, \mathcal{P}^{\mathcal{D}})$, where $\Delta^{\mathcal{D}}$ is a set and $\mathcal{P}^{\mathcal{D}}$ is a set of predicate names; each $p \in \mathcal{P}^{\mathcal{D}}$ has an arity $n > 0$ and an extension $p^{\mathcal{D}} \in (\Delta^{\mathcal{D}})^n$. The link between the DL and concrete domains is accomplished via the introduction of *feature names* $\mathbf{F}$ and the concrete domain constructor included in Figure 2; p is used to denote a predicate of a concrete domain, and $f_1, ..., f_k$ to denote feature names. The interpretation function maps each feature name f to a partial function from $\Delta^{\mathcal{I}}$ to $\bigcup_{1 \leq i \leq n} \Delta^{\mathcal{D}_i}$, where in general it is assumed that $\Delta^{\mathcal{D}_i} \cap \Delta^{\mathcal{D}_j} = \emptyset$ for $1 \leq i < j \leq n$.

$\mathcal{EL}^{++}$ Knowledge Bases. A KB consists of two sets, referred to as the ABox and the TBox, respectively containing the *extensional* knowledge about individual objects and *intensional* knowledge about the general notions for the domain in question. The ABox formally consists of a finite set of concept assertions and role assertions, while the TBox is comprised of a finite set of *constraints*, which can be general concept inclusions (GCIs), role inclusions (RIs), domain restrictions (DRs), or range restrictions (RRs) (cf. Figure 2). An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ (resp., ABox $\mathcal{A}$) iff, for each contained constraint (resp., assertion), the conditions in the "Semantics" column of Figure 2 are satisfied. We note that the expressive power of $\mathcal{EL}^{++}$ allows the expression of role hierarchies, role equivalences, transitive roles, reflexive roles, left- and right-identity rules, disjointness of complex concept descriptions, and the identity and distinctness of individuals.

Finally, here we adopt the syntactic restriction presented in [1] to avoid intractability/undecidability, which prevents intricate interplay between role inclusions and range restrictions; we do not describe it here for reasons of space.

2.2 Markov Logic Networks

Markov logic networks (MLNs) [20] combine first-order logic with Markov networks (abbreviated MNs; they are also known as Markov random fields) [19]. We now provide a brief introduction first to MNs, and then to MLNs.

Markov Networks. A Markov network (MN) is a probabilistic model that represents a joint probability distribution over a (finite) set of random variables $X = \{X_1, \ldots, X_n\}$. Each random variable X_i may take on *values* from a finite *domain* $Dom(X_i)$. A *value* for $X = \{X_1, \ldots, X_n\}$ is a mapping $x \colon X \to \bigcup_{i=1}^{n} Dom(X_i)$ such that $x(X_i) \in Dom(X_i)$; the *domain* of X, denoted $Dom(X)$, is the set of all values for X. An MN is similar to a Bayesian network (BN) in that it includes a graph $G = (V, E)$ in which each node corresponds to a variable, but, differently from a BN, the graph is undirected; in an MN, two variables are connected by an edge in G iff they are conditionally dependent. Furthermore, the model contains a *potential function* ϕ_i for each (maximal) clique in the graph; potential functions are non-negative real-valued functions of the values of the variables in each clique (called the *state* of the clique). In this work, we assume the *log-linear* representation of MNs, which involves defining a set of *features* of such states; a feature is a real-valued function of the state of a clique (we only consider binary features in this work). Given a value $x \in Dom(X)$ and a feature f_j for clique j, the probability distribution represented by an MN is given by $P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j \cdot f_j(x)\right)$, where j ranges over the set of cliques in the graph G, and $w_j = \log \phi_j(x_{\{j\}})$ (here, $x_{\{j\}}$ is the state of the j-th clique). The term Z is a normalization constant to ensure that the values given by the equation above are in $[0, 1]$; it is given by $Z = \sum_{x \in Dom(X)} \exp\left(\sum_j w_j \cdot f_j(x)\right)$. Probabilistic inference in MNs is intractable; however, approximate inference mechanisms, such as Markov Chain Monte Carlo, have been developed and successfully applied.

Markov Logic Networks. In the following, let Δ_{MLN}, $\mathcal{V}_{MLN}$, and $\mathcal{R}_{MLN}$ denote the set of constants, variables, and predicate symbols. The main idea behind Markov logic networks (MLNs) is to provide a way to soften the constraints imposed by a set of classical logic formulas. Instead of considering worlds that violate some formulas to be impossible, we wish to make them less probable. An MLN is a finite set L of pairs (F_i, w_i), where F_i is a formula in first-order logic over $\mathcal{V}_{MLN}$, Δ_{MLN}, and $\mathcal{R}_{MLN}$, and w_i is a real number. Such a set L, along with a finite set of constants Δ_{MLN}, defines a Markov network M that contains: (i) one binary node corresponding to each element of the Herbrand base of the formulas in L (i.e., all possible ground instances of the atoms), where the node's value is 1 iff the atom is true; and (ii) one feature for every possible

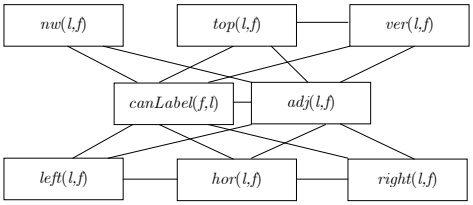


Figure 3: The graph representation of a simple grounding of the MLN from Example 2.

ground instance of a formula in L. The value of the feature is 1 iff the ground formula is true, and the weight of the feature is the weight corresponding to the formula in L. From this characterization and the description above of the graph corresponding to an MN, it follows that M has an edge between any two nodes corresponding to ground atoms that appear together in at least one formula in L. Furthermore, the probability of $x \in Dom(X)$ given this ground MLN is

$$P(X = x) = \frac{1}{Z} \cdot \exp\left(\sum_j w_j \cdot n_j(x)\right), \qquad (1)$$

where $n_i(x)$ is the *number of ground instances* of F_i made true by x, and Z is defined as above. This formula can be used in a generalized manner to compute the probability of any setting of a subset of random variables $X' \subseteq X$.

Example 2. Consider again the form-labeling problem of Example 1. The fact that certain text blocks are more likely to represent field labels than others (which is part of their *phenomenology*) can be described by assigning weights to the following first-order formulas:

$\phi_1 : canLabel(Y, X) \wedge hor(X, Y) \wedge left(X, Y) \wedge adj(X, Y)$;

$\phi_2 : canLabel(Y, X) \wedge hor(X, Y) \wedge right(X, Y) \wedge adj(X, Y)$;

$\phi_3 : canLabel(Y, X) \wedge ver(X, Y) \wedge top(X, Y) \wedge adj(X, Y)$;

$\phi_4 : canLabel(Y, X) \wedge nw(X, Y) \wedge adj(X, Y)$.

Formula ϕ_1 describes labels that are left-adjacent and horizontally aligned with the field. The dual case, with right-adjacent labels, is captured by ϕ_2. The case of top-adjacent labels that are vertically-aligned to the field is expressed by formula ϕ_3, while the last expression (ϕ_4) describes the case of labels appearing in the north-west area of the field.

The formulas above are part of the MLN associated with the $\mathcal{EL}^{++}$ formulas describing the form ontology. As an example, the following MLN formulas reflect the likelihood of the particular labeling phenomenology.

$\psi_1 : (\phi_1, 9), \;\; \psi_2 : (\phi_2, 6), \;\; \psi_3 : (\phi_3, 5), \;\; \psi_4 : (\phi_4, 1)$. ∎

Figure 3 shows a simple grounding of the MLN described in Example 2, relative to the set of constants $\{f, \ell\}$ (resp., one field and one label). This grounding also assumes that we have additional constraints (not shown here for simplicity) that only allow arguments of predicates to take the type of argument they expect; this sort of constraints is provided in certain implementations of MLNs such as Tuffy[1], where they are called *predicate scoping rules*.

3 Tightly Coupled Probabilistic DLs

Considering the basic setup from Sections 2.1 and 2.2, we now present the language of probabilistic $\mathcal{EL}^{++}$.

3.1 Syntax

Recall that we have (as discussed in Sections 2.1 and 2.2) an infinite universe of individual names $\mathbf{I}$, a finite set of concept names $\mathbf{C}$ and role names $\mathbf{R}$, a finite set of constants Δ_{MLN}, an infinite set of variables $\mathcal{V}_{MLN}$, and a finite set of predicate names $\mathcal{R}_{MLN}$ (such that $\mathcal{R}_{MLN} \cap \mathbf{C} = \emptyset$ and $\mathcal{R}_{MLN} \cap \mathbf{R} = \emptyset$). Finally, recall that the set of random variables in the MLN coincides with the set of ground atoms over $\mathcal{R}_{MLN}$ and Δ_{MLN}; alternatively, we denote this set with $X = \{X_1, \ldots, X_n\}$, as in Section 2.2.

Substitutions and Unifiers. We adopt the usual definitions from classical logic. A substitution is a function from variables to variables or constants. Two sets S and T *unify* via a substitution θ iff $\theta S = \theta T$, where θA denotes the application of θ to all variables in all elements of A (here, θ is a unifier). A *most general unifier* (mgu) is a unifier θ such that for all other unifiers ω, there exists a substitution σ such that $\omega = \sigma \circ \theta$.

Translation into FOL. In the rest of this work, we assume that $\mathcal{EL}^{++}$ TBoxes and ABoxes are translated into their equivalent first-order logic formulas; therefore, when clear from the context, we refer to *axioms* or *assertions* without distinguishing them from their translations into FOL. This is required for technical reasons, such as the need to be able to explicitly refer to the variables in the axioms in order to have the possibility of linking such variables with those in probabilistic annotations. Note that this does not affect the expressiveness of our formalism, nor its tractability, since the translation to FOL (and back to $\mathcal{EL}^{++}$) can be done in polynomial time in the size of the ontology.

Informally, probabilistic ontologies consist of a finite set of first-order logic formulas that correspond to the translation of $\mathcal{EL}^{++}$ axioms; each such formula is associated with a probabilistic annotation, as described next.

Definition 1. A *probabilistic annotation* λ relative to an MLN M defined over $\mathcal{R}_{MLN}$, $\mathcal{V}_{MLN}$, and Δ_{MLN} is a (finite) set of pairs $\langle A_i, x_i \rangle$, where: (i) A_i is an atom over $\mathcal{R}_{MLN}$, $\mathcal{V}_{MLN}$, and Δ_{MLN}; (ii) $x_i \in \{0, 1\}$; and (iii) for any two pairs $\langle A, x \rangle, \langle B, y \rangle \in \lambda$, there does not exist substitution θ that unifies A and B. If $|\lambda| = |X|$ and all $\langle A, x_i \rangle \in \lambda$ are such that A is ground, then λ is called a *(possible) world*.

[1] http://research.cs.wisc.edu/hazy/tuffy/

Intuitively, a probabilistic annotation λ is used to describe the class of events in which the random variables in an MLN are compatible with the settings of the random variables described by λ, i.e., each X_i has the value x_i.

Definition 2. Let F be the FOL translation of an $\mathcal{EL}^{++}$ axiom, and λ be a probabilistic annotation; a *probabilistic $\mathcal{EL}^{++}$ axiom* is of the form $F : \lambda$. We also refer to probabilistic axioms as *annotated formulas*.

Essentially, probabilistic axioms hold whenever the events associated with their annotations occur. Note that whenever a random variable's value is left unspecified in an annotation, the variable is *unconstrained*; in particular, an empty annotation means that the formula holds in every possible world (we sometimes refer to these axioms as *crisp*).

Definition 3. Let O be a set of (FOL translations of) probabilistic $\mathcal{EL}^{++}$ axioms and M be an MLN. A *tightly coupled probabilistic $\mathcal{EL}^{++}$ ontology* (TCP ontology, or knowledge base) is of the form $KB = (O, M)$, where the probabilistic annotations of formulas in O are relative to M.

Recall that random variables in our MLN setting are Boolean and written in the form of atoms over $\mathcal{R}_{MLN}$, $\mathcal{V}_{MLN}$, and Δ_{MLN}; if a is such an atom, $a = 1$ (resp., $a = 0$) denotes that the variable is *true* (resp., *false*); we also use the notation a and $\neg a$, respectively.

Definition 4. Let $KB = (O, M)$ be a *probabilistic $\mathcal{EL}^{++}$ ontology*, and λ be a possible world. The (non-probabilistic) $\mathcal{EL}^{++}$ ontology *induced* from KB by λ, denoted O_λ, is the set $\{\theta_i F_i \mid F_i : \lambda_i \in O \text{ and } \theta_i\lambda_i \subseteq \lambda\}$, where θ_i is an mgu for λ and λ_i.

The annotation of ontological axioms offers a clear modeling advantage by enabling a *clear separation of concerns* between the task of ontological modeling and the task of modeling the uncertainty around the axioms in the ontology. More precisely, in our formalism, it is possible to express the fact that the probabilistic nature of an ontological axiom is determined by elements that are *outside of the domain modeled by the ontology*.

Example 3. Consider the form-labeling ontology of Example 1 and the MLN of Example 2. In general, we expect only certain axioms to be probabilistic, while others are necessarily crisp. In our case, the fact that fields and text blocks are disjoint sets of objects, and that fields are labeled by text blocks are considered crisp axioms, since we do not want these assumptions to be violated by any model. However, we want to accept models where some field is left unlabeled, and therefore violating the axiom $Field \sqsubseteq \exists label.Text$ is possible. In addition, we want to link the probability that this axiom holds to the heuristics used to produce the actual labeling of the field, which are out of the domain of the form-labeling ontology. The following is a possible probabilistic $\mathcal{EL}^{++}$ ontology modeling this setup, where the first-order representation of the $\mathcal{EL}^{++}$ axioms discussed above is used to explicitly state the relationships between variables and constants in the MLN and in the ontology.

$$\forall X.\mathit{field}(X) \rightarrow \exists Y.\mathit{label}(X,Y) \wedge \mathit{text}(Y) : \{\langle \mathit{canLabel}(Y,X), 1\rangle\};$$

$$\forall X.\mathit{label}(X,Y) \rightarrow \mathit{field}(X) : \{\};$$

$$\forall X.\mathit{label}(X,Y) \rightarrow \mathit{text}(Y) : \{\};$$

$$\forall X.\mathit{field}(X) \wedge \mathit{text}(X) \rightarrow \bot : \{\}. \qquad \blacksquare$$

Data Complexity. In this setting, we extend the usual concept of data complexity as follows: the set of formulas in the MLN are considered to be fixed, as is the TBox; on the other hand, the sets $\mathbf{I}$ and Δ_{MLN} are not, and therefore the ground ABox on the ontology side and the set of random variables on the MLN side *are not fixed*, either.

3.2 Semantics

The semantics of TCP ontologies is given relative to probabilistic distributions over *interpretations* of the form $\mathcal{I}_{MLN} = \langle D, w\rangle$, where D is a database over $\mathbf{I} \cup \Delta_N$, and w is a world. We usually abbreviate "$true : \lambda$" with "λ".

Definition 5. An interpretation $\mathcal{I}_{MLN} = \langle D, w\rangle$ *satisfies* an annotated formula $F : \lambda$, denoted $\mathcal{I}_{MLN} \models F : \lambda$, iff whenever there exists an mgu θ such that for all $\langle V_i, x_i\rangle \in \lambda$ it holds that $X_i = \theta V_i$ and $w[i] = x_i$, then $D \models \theta F$.

A *probabilistic interpretation* is then a probability distribution *Pr* over the set of all possible interpretations such that only a finite number of interpretations are mapped to a non-zero value. The probability of an annotated formula $F : \lambda$, denoted $\mathit{Pr}(F : \lambda)$, is the sum of all $\mathit{Pr}(\mathcal{I}_{MLN})$ such that $\mathcal{I}_{MLN}$ satisfies $F : \lambda$.

Definition 6. Let *Pr* be a probabilistic interpretation, and $F : \lambda$ be an annotated formula. We say that *Pr satisfies* (or is a *model* of) $F : \lambda$ iff $\mathit{Pr}(F : \lambda) = 1$. Furthermore, *Pr* is a model of a probabilistic $\mathcal{EL}^{++}$ ontology $KB = (O, M)$ iff: (i) *Pr* satisfies all annotated formulas in O, and (ii) $1 - \mathit{Pr}(\mathit{false} : \lambda) = \mathit{Pr}_M(\lambda)$ for all possible worlds λ, where $\mathit{Pr}_M(\lambda)$ is the probability of $\bigwedge_{\langle V_i, x_i\rangle \in \lambda}(V_i = x_i)$ in the MLN M (and computed in the same way as $P(X = x)$ in Section 2.2).

In Definition 6 above, condition (ii) is stating that the probability values that *Pr* assigns are in accordance with those of MLN M (note that the equality in the definition implies that $\mathit{Pr}(true : \lambda) = \mathit{Pr}_M(\lambda)$) and that they are adequately distributed (since $\mathit{Pr}(true : \lambda) + \mathit{Pr}(false : \lambda) = 1$).

In the following, we are interested in computing the probabilities of atoms in a TCP ontology, working towards ranking of entailed atoms based on their probabilities.

Definition 7. Let $KB = (O, M)$ be a TCP ontology, and a be a ground atom that is constructed from predicates and

individuals in KB. The *probability* of a in KB, denoted $Pr^{KB}(a)$, is the infimum of $Pr(a:\{\})$ subject to all probabilistic interpretations Pr such that $Pr \models KB$.

Intuitively, an atom has the probability that results from summing the probabilities of all possible worlds under which the induced ontology entails the atom.

3.3 Ranking of Atoms based on Probabilities

In this paper, we focus on queries requesting the ranking of atoms based on their probability values.

Definition 8. Let $KB = (O, M)$ be a TCP ontology; the *answer* to a *ranking query* $Q = rank(KB)$ is a tuple $ans(Q) = \langle a_1, \ldots, a_n \rangle$ such that $\{a_1, \ldots, a_n\}$ are all of the atomic consequences of O_λ for any possible world λ, and $i < j \Rightarrow Pr^{KB}(a_i) \geq Pr^{KB}(a_j)$.

The straightforward approach to answering ranking queries is to compute the probabilities of all atoms inferred by the knowledge base; unfortunately, computing exact probabilities of atoms is intractable.

Theorem 1. *Let $KB = (O, M)$ be a TCP ontology. Computing $\Pr^{KB}(a)$ is $\#P$-hard in the data complexity.*

In the next section, we explore how this negative result can be avoided by considering a special kind of MLN that allows us to compute *scores* instead of probability values. Such scores are closely related to probabilities, and allow us to rank answers with respect to actual probability values.

4 Tractably Answering Ranking Queries

In this section, we consider a *special case of MLNs* that proves to be useful towards more tractable methods to answer ranking queries.

4.1 Conjunctive MLNs and Equivalence Classes

We now introduce a simple class of MLNs:

Definition 9. A *conjunctive* Markov logic network (cMLN) is an MLN in which all formulas (F, w) in the set are such that F is a conjunction of atoms, and $w \in \mathbb{R}$.

Informally, a cMLN is an MLN in which formulas are restricted to conjunctions of atoms. This restriction allows us to define an equivalence relation over the set of worlds. Given a cMLN M and its grounding $gr(M, \Delta_{MLN})$, we use the notation $Sat(\lambda, M)$ to denote the set of all ground formulas F in $gr(M, \Delta_{MLN})$ that are satisfied by world λ. It is easy to see that $gr(M, \Delta_{MLN})$ can be computed in polynomial time in the data complexity; we therefore work with ground cMLNs in the rest of the paper.

Definition 10. Let M be a ground cMLN, and λ_1 and λ_2 be possible worlds. We say that λ_1 and λ_2 are *equivalent* with respect to M, denoted $\lambda_1 \sim_M \lambda_2$, iff $Sat(\lambda_1, M) = Sat(\lambda_2, M)$.

Clearly, $\sim_M$ is an equivalence relation; we denote the equivalence classes induced by this relation with $C_1, \ldots, C_N$; note that, in general, there are $2^{|gr(M, \Delta_{MLN})|}$ such classes, making it intractable to inspect all of them.

Example 4. Consider the MLN in Example 2, and the grounding discussed above (cf. Figure 3). In this case, each of ψ_1 to ψ_4 has one possible grounding, which we call f_1 to f_4. There are therefore 16 equivalence classes in this case (each ground formula can be negated or not), which form a partition of the $2^8 = 256$ possible worlds. ■

There are several properties of equivalence classes that we can leverage. First, equivalence classes are not guaranteed to be non-empty, but it is simple to check for this condition.

Theorem 2. *Let M be a ground cMLN, and C be a $\sim_M$-equivalence class. Deciding $C = \emptyset$ can be done in polynomial time.*

Furthermore, generating worlds for a given equivalence class is also tractable:

Theorem 3. *Let M be a ground cMLN, and C be a $\sim_M$-equivalence class. All elements in C can be obtained in linear time with respect to the size of the output.*

Proof sketch. Let C be described by formula *pos* $\wedge$ *neg*, where *pos* is the conjunction of all formulas from M that are true in C, while *neg* is the conjunction of the negations of all formulas not true in C. The set of all atoms in the Herbrand base can be divided into two sets: *det* and *undet*; the former contains all (possibly negated) atoms that are determined by *pos*, while the latter contains the rest. The worlds in C are obtainable by traversing *neg* and assigning truth values to atoms in this formula in all possible ways that make C true, without contradicting the atoms in *det*. □

Finally, we point out an important characteristic of equivalence classes with respect to probability values:

Theorem 4. *Let M be a ground cMLN, and λ_1 and λ_2 be possible worlds. If $\lambda_1 \sim_M \lambda_2$, then $\Pr_M(\lambda_1) = \Pr_M(\lambda_2)$.*

Proof sketch. Follows from Definition 10. Since both worlds belong to the same class, they satisfy (and do not satisfy) the same formulas, and so the value $\exp\left(\sum_j w_j \cdot n_j(\lambda_i)\right)$ from Equation 1 is the same for both worlds; the denominator (factor Z) is equal across all worlds, which means that their probabilities are equal. □

However, computing exact probabilities remains intractable in cMLNs.

Theorem 5. *Let $KB = (O, M)$ be a* TCP *ontology, where M is a cMLN. Deciding $\Pr^{KB}(a) \geq k$ is PP-hard in the data complexity.*

The complexity class *PP* contains problems decidable by a probabilistic Turing machine in polynomial time, with error probability less than 1/2; like $\#P$, a polynomial time Turing machine with a *PP* oracle can solve all problems in the polynomial hierarchy [21]. This negative result does not prevent us, however, from tractably *comparing* the probabilities of two worlds.

Proposition 1. *Let M be a cMLN, and λ_1 and λ_2 be possible worlds. Deciding whether $\Pr_M(\lambda_1) \leq \Pr_M(\lambda_2)$ is in PTIME in the data complexity.*

The basic intuition behind this result is that we can compute the term $n_i(x)$ in Equation 1 in polynomial time; since the denominator in this equation is the same for all worlds, $\Pr_M(\lambda_1) \leq \Pr_M(\lambda_2)$ can be decided by only computing the numerators in this equation.

Example 5. Consider the following worlds relative to the MLN in Example 2 and the grounding from Example 4:

$$\begin{aligned}\lambda_1 &= \{\textit{canLabel}(f,\ell),\ \textit{hor}(\ell,f),\ \neg\textit{left}(\ell,f),\ \textit{adj}(\ell,f),\\ &\quad \textit{right}(\ell,f),\ \neg\textit{top}(\ell,f),\ \neg\textit{nw}(\ell,f),\ \neg\textit{ver}(\ell,f)\};\\ \lambda_2 &= \{\textit{canLabel}(f,\ell),\ \neg\textit{hor}(\ell,f),\ \neg\textit{left}(\ell,f),\ \textit{adj}(\ell,f),\\ &\quad \neg\textit{right}(\ell,f),\ \textit{top}(\ell,f),\ \neg\textit{nw}(\ell,f),\ \textit{ver}(\ell,f)\}.\end{aligned}$$

A quick inspection of the formulas in the MLN allows us to conclude that $\lambda_1 \models \neg f_1 \wedge f_2 \wedge \neg f_3 \wedge \neg f_4$, while $\lambda_2 \models \neg f_1 \wedge \neg f_2 \wedge f_3 \wedge \neg f_4$. Therefore, taking into account the weights of each formula, we can see that $\Pr(\lambda_1) > \Pr(\lambda_2)$; in fact, even though we cannot (tractably) compute the actual probabilities, we can conclude that $\Pr(\lambda_1) = \frac{e^6}{e^5}\Pr(\lambda_2)$, or that λ_1 is approximately 2.7 times more probable than λ_1. ■

4.2 An Anytime Algorithm

The results in the previous section point the way towards Algorithm anytimeRank for answering ranking queries; the pseudocode for this algorithm is shown in Figure 4; in the rest of this section, we will discuss its properties.

Algorithm anytimeRank takes a probabilistic ontology in which the MLN is assumed to be a ground cMLN (recall that the grounding can be computed in polynomial time in the data complexity); the other input corresponds to a *stopping condition* that can be based on whatever the user considers important (time, number of steps, number of inspected worlds, etc); cf. Section 4.2.2 for a discussion on ways to define the stopping condition based on properties of the output offering correctness guarantees.

The main `while` loop in line 3 iterates through the set of equivalence classes relative to M. Subroutine *compMostProbEqClass*, invoked in line 4, computes the i-th most probable equivalence class. Note that this can simply be done by taking the formulas in M and sorting them with respect to their weights; the classes are then generated by keeping track of a Boolean vector of which formulas are true and which are false. The next `while` loop, in line 7, is in charge of going through the current equivalence class. Subroutine *computePossWorld* takes the current class and a set of already inspected worlds and computes a new world (not in S). This can be done as described in the proof sketch of Theorem 3; in particular, the possible combinations of atoms in formula *neg* can be traversed in order, without the need to explicitly keep track of a set like S. The final lines of this loop take the computed world and obtain the atomic consequences from the (non-probabilistic) induced subontology (line 10), and adds the *score* to each such atom (lines 11 and 12). The *score* of a class consists of e to the power of the sum of the weights of formulas that are true in that class. Line 13 updates the output set of atoms. Finally, set *out* is returned in decreasing order of score.

Example 6. Consider the probabilistic $\mathcal{EL}^{++}$ ontology $\Phi = (O, M)$, where O is given as follows:

$$\begin{array}{ll}\forall X p(X) \rightarrow q(X) & : \{\langle m(X), 1\rangle, \langle n(X), 0\rangle\};\\ p(a) & : \{\langle m(a), 1\rangle\};\\ p(b) & : \{\langle n(a), 1\rangle\};\\ p(c) & : \{\langle m(c), 1\rangle, \langle n(c), 0\rangle\},\end{array}$$

and M is given by $\{(m(X), 1.5), (n(X), 0.8)\}$. Suppose we ground M with the set of constants $\{a, b, c\}$, yielding:

$$\begin{array}{lll} f_1 : (m(a), 1.5), & f_2 : (n(a), 0.8), & f_3 : (m(b), 1.5),\\ f_4 : (n(b), 0.8), & f_5 : (m(c), 1.5), & f_6 : (n(c), 0.8).\end{array}$$

This setup therefore yields $2^6 = 64$ equivalence classes (in this case, we have exactly one world per class). Figure 5 shows a subset of these classes in decreasing order of score (as they will be inspected by Algorithm anytimeRank). The algorithm will proceed as follows; cf. Figure 5 for the description of the classes (we use classes C_i to denote the single world λ_i in that class):

$O_{C_1} \models \{p(a), p(b)\}$; add $e^{6.9}$ to *score*$(p(a))$ and *score*$(p(b))$;

$O_{C_2} \models \{p(a), p(b), p(c), q(c)\}$; add $e^{6.1}$ to *score*$(p(a))$, *score*$(p(b))$, *score*$(p(c))$, and *score*$(q(c))$;

$O_{C_3} \models \{p(a)\}$; add $e^{6.1}$ to *score*$(p(a))$;

$O_{C_4} \models \{p(a), p(c), q(c)\}$; add $e^{6.1}$ to *score*$(p(a))$, *score*$(p(c))$, and *score*$(q(c))$;

$O_{C_5} \models \{p(a), p(b), q(a)\}$; add $e^{5.3}$ to *score*$(p(a))$, *score*$(p(b))$, and *score*$(q(a))$.

If the algorithm is stopped at this point, the scores are as follows (approximate, and in descending order):

$p(a) : 2530.18, p(b) : 1638.47, p(c) : 891.71, q(c) : 891.71$ ■

4.2.1 Correctness and Running Time of anytimeRank

The correctness of this algorithm lies in the fact that if all classes are inspected, the returned output set is clearly the answer to the ranking query. In the general case, only a

Algorithm anytimeRank($KB = (O, M)$, *stopCond*)
// M is assumed to be a ground cMLN
1. *score*:= empty mapping from atoms to $\mathbb{R}$ (default 0);
2. *out*:= empty set of atoms; $i := 1$;
3. while $\left(i \leq 2^{|M|}\right)$ and !*stopCond* do begin
   // i ranges over classes of possible worlds
4. $C := compMostProbEqClass(M, i)$;
5. $S := \emptyset$;
6. $i := i + 1$;
7. while $\left(|S| \neq |C|\right)$ and !*stopCond* do
8. $\lambda := computePossWorld(C, S)$;
   // compute world s.t. $\lambda \in C$ and $\lambda \notin S$
9. $S := S \cup \{\lambda\}$;
10. $O_\lambda := getInducedOnt(O, \lambda)$;
11. for all atoms $a \in$ *atomicCons*(O_λ) do
12. *score*(a)+= $\exp\left(\sum_{F_j \in M, C \models F_j} w_j\right)$;
13. *out*:= *out* $\cup$ *atomicCons*(O_λ);
14. end;
15. return *out* sorted in dec. order according to *score*.

Figure 4: An anytime algorithm to compute the answer to a ranking query over a TCP ontology (refer to text for description of subroutines).

Class	f_1	f_3	f_5	f_2	f_4	f_6	$\sum_j w_j$
C_1	1	1	1	1	1	1	6.9
C_2	1	1	1	1	1	0	6.1
C_3	1	1	1	1	0	1	6.1
C_4	1	1	1	0	1	1	6.1
C_5	1	1	1	1	0	0	5.3
...							

Figure 5: Equivalence classes from Example 6, sorted in descending order of the *score* assigned to possible worlds that belong to them (e to the power of the value in the last column); only 5 of the 64 classes are shown here.

subset of the worlds will be inspected; since the probabilities of worlds in a given equivalence class are all equal (Theorem 4), and this value depends directly on the formulas in the cMLN that are satisfied by the class, the iteration through the equivalence classes in decreasing order of probability is the optimal path to take. Though the total running time of course depends on the stopping condition, Theorem 3, along with the way in which equivalence classes are manipulated (as described above), guarantee a running time that is polynomial in the data complexity as long as the combined number of inspected worlds and equivalence classes is bounded by a polynomial as well.

4.2.2 Bounding the Error of anytimeRank

The following proposition provides a bound on the total "mass" of score that remains unassigned by our algorithm after a certain number of iterations.

Proposition 2. *Let $KB = (O, M)$ be a* TCP *ontology where M is a ground cMLN with n ground atoms, and let $C_1, ..., C_{2^{|M|}}$ be the set of equivalence classes of M sorted in decreasing order of their score. Then, after analyzing s worlds and t classes with Algorithm* anytimeRank*, the total unassigned class score mass is bounded by above by $U = (2^n - s) \cdot \exp\left(\sum_{F_j \in M, C_{t+1} \models F_j} w_j\right)$.*

This result is useful, for instance, in determining a *provably correct partial order* over the output of the algorithm. For example, if the output is $\{(a, 120), (b, 90), (c, 80), (d, 10)\}$ and $U = 30$, we can safely conclude that $\Pr^{KB}(a) > \Pr^{KB}(c)$, $\Pr^{KB}(a) > \Pr^{KB}(d)$, $\Pr^{KB}(b) > \Pr^{KB}(d)$, and $\Pr^{KB}(c) > \Pr^{KB}(d)$.

Theorem 6. *Let out be the output of Algorithm* anytimeRank *and U be the bound on the unassigned score mass as computed in Proposition 2. The partial order $\leq_U$ defined as: $a \leq_U b$ iff $s_a + U \leq s_b$, where $(a, s_a), (b, s_b) \in$ out, is such that if $a \leq_U b$ then $\Pr(a) \leq \Pr(b)$.*

Therefore, Theorem 6 allows us to glimpse into the total order over the set of atoms as established by the true probability values, without actually computing them.

5 Related Work

Ontology languages, rule-based systems, and their integrations are central for the Semantic Web [2]. Although many approaches exist to tight, loose, or hybrid integrations of ontology languages and rule-based systems, to our knowledge there is very little work on the combination of tractable description logics with MLNs. Probabilistic ontology languages in the literature can be classified according to the underlying ontology language, the supported forms of probabilistic knowledge, and the underlying probabilistic semantics (see [14] for a recent survey). Some early approaches [11] generalize the description logic $\mathcal{ALC}$ and are based on propositional probabilistic logics, while others [12] generalize the tractable DL CLASSIC and $\mathcal{FL}$, and are based on Bayesian networks as underlying probabilistic semantics. The fairly recent approach in [13], generalizing the expressive DL $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ behind the sublanguages *OWL Lite* and *OWL DL*, respectively, of the Web ontology language *OWL* [18], is based on probabilistic default logics, and allows for rich probabilistic terminological and assertional knowledge. Other recent approaches [23] generalize OWL by probabilistic uncertainty using Bayesian networks.

In the probabilistic description logics literature, the most closely related work is that of Prob-$\mathcal{EL}$ [15, 10], a probabilistic extension to $\mathcal{EL}$ that belongs to a family of probabilistic DLs derived in a principled way from Halpern's probabilistic first-order logic [4]. One limitation in this line of research is that probabilistic annotations are somewhat restricted, leading to the inability to express uncertainty about certain kinds of general knowledge; note that our formalism does not suffer from this drawback, since probabilistic annotations can be associated with any axiom. In [15], the authors study various logics in this family and

show complexity of reasoning, ranging from PTIME for weak variants of $\mathcal{EL}$ to undecidable for expressive variants of $\mathcal{ALC}$. Prob-$\mathcal{EL}$ is more closely studied in [10], where the authors show that reasoning is PTIME as long as (i) probability values are restricted to 0 and 1, and (ii) probabilistic annotations are only allowed on concepts; if (i) is dropped, then it becomes EXPTIME-complete, while if (ii) is dropped it becomes PSPACE-hard. The complexity results in our work are further testament to how intractable simple tasks become when probabilistic computations are involved, even when the starting point is a tractable logic.

Other recent efforts focused on extending $\mathcal{EL}$ DLs with probabilistic uncertainty include [5], [9], and [17]. In [5], a formalism is presented in which probability assessments are only allowed on ABoxes. The authors study the problem of satisfiability of KBs, which involves determining whether there exists a probability distribution that satisfies all the assignments over the ABoxes. The work of [9] is similar in spirit to [5], but there an MLN is employed in order to infer the probabilities of atoms in the ABox as a means to generate explanations as part of abductive inference. Though they use MLNs, this work is quite different from ours since only the ABox is assumed to be probabilistic, and the assertions are themselves part of the MLN instead of being annotated by external events as in our formalism. In [17], the authors present ELOG, which is $\mathcal{EL}^{++}$ without nominals or concrete domains combined with probabilistic log-linear models (a class which contains MLNs). The resulting probabilistic formalism basically assigns weights to axioms reflecting how likely the axiom is to hold. The main problem then corresponds to finding the most probable coherent ontology, a problem that is essentially different from the one tackled here.

Finally, a related formalism from the recent databases literature is that of probabilistic Datalog+/– [8, 7]. Datalog+/– is a language that arose from the generalization of rule-based constraints with the goal of expressing ontological axioms. The probabilistic extension in [8, 7] also makes use of MLNs, though the integration is loose in the sense that probabilistic annotations cannot refer to objects in the ontology, which leads to data-tractable algorithms but also limits the expressive power of the formalism.

6 Discussion, Conclusions, and Future Work

In this work, we have extended the DL $\mathcal{EL}^{++}$ with probabilistic uncertainty, based on the annotation of axioms. Such annotations refer to events whose probabilities are described by an associated MLN; one of the advantages of this formalism is that it is tightly coupled, which means that probabilistic annotations can refer to objects in the ontology. The proposed application of our formalism is in managing uncertainty in the Semantic Web, showing examples of how it can be applied in the analysis of Web forms, an important task in information extraction efforts.

Our focus here is on ranking queries, which request the set of atomic inferences sorted in descending order of probability. The algorithm we developed works in an anytime fashion, and therefore allows partial computations depending on the available resources; most importantly, we provide bounds on the error incurred by runs of this algorithm and conditions that allow to conclude when certain pairs in the output are correctly ordered. This algorithm works over cMLNs, in which only conjunctions of atoms are allowed. Regarding the expressivity of cMLNs, we can say that: (i) They are rich enough to simulate disjunction for a specific propositional subset. For instance, if the formula $p(X) \vee q(X)$ with a given weight needs to be enforced for the subset of individuals $\{a, b\}$, MLN learning algorithms can be directed to give corresponding weights to the specific worlds in which $\big(p(a)$ or $q(a)\big)$ and $\big(p(b)$ or $q(b)\big)$ hold. So, it is possible to represent certain special cases that may need to be handled. (ii) For the case of (atomic) negation, it is known to be representable via negative weights. (iii) Though material implications cannot be represented, this sort of constraint is more adequately placed on the ontology side, and the TBox is capable of representing them.

Regarding the practical applicability of the formalism, we can say that, despite probabilistic instance-checking being already intractable for cMLNs, it is often possible to bound the number of scenarios in practice. Consider, for instance, the problem of reasoning over data structures on the Web (related to the running example). Web pages are usually processed in isolation, since structured data do not usually span across pages and are often delimited by a certain DOM sub-tree. This implies that the number of possible worlds is bounded by a function of the constants appearing in a subset of the page. This bound can work together with the good computational behavior of cMLNs to allow tractability of reasoning in practice. Other applications where cMLNs can be leveraged are semantic and natural language-based Web search.

Future work involves investigating other DLs that can be extended in this manner, and pushing the known line between tractability and expressivity. We also need to empirically evaluate our approach both on synthetic and real-world data, as well as studying the application of other techniques such as random sampling, which may provide increased scalability at the cost of lost guarantees.

Acknowledgments

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/J008346/1 ("PrOQAW"), the Oxford Martin School grant LC0910-019, the European Research Council (ERC) grant FP7/246858 ("DIADEM"), a Google Research Award, and a Yahoo! Research Fellowship.

References

[1] BAADER, F., BRANDT, S., AND LUTZ, C. Pushing the el envelope further. In *Proc. of OWLED* (2008), K. Clark and P. F. Patel-Schneider, Eds.

[2] BERNERS-LEE, T., HENDLER, J., AND LASSILA, O. The Semantic Web. *Scientific American 284* (2002), 34–43.

[3] CALÌ, A., GOTTLOB, G., AND LUKASIEWICZ, T. A general Datalog-based framework for tractable query answering over ontologies. In *Journal of Web Semantics* (2012).

[4] FAGIN, R., HALPERN, J. Y., AND MEGIDDO, N. A logic for reasoning about probabilities. *Information and Computation 87*, 1/2 (1990), 78–128.

[5] FINGER, M., WASSERMANN, R., AND COZMAN, F. G. Satisfiability in EL with sets of probabilistic aboxes. In *Proc. of DL* (2011).

[6] FURCHE, T., GOTTLOB, G., GRASSO, G., GUO, X., ORSI, G., AND SCHALLHART, C. OPAL: Automated form understanding for the deep Web. In *Proc. of WWW* (2012).

[7] GOTTLOB, G., LUKASIEWICZ, T., AND SIMARI, G. I. Answering threshold queries in probabilistic Datalog+/– ontologies. In *Proc. of SUM* (2011), LNCS, pp. 401–414.

[8] GOTTLOB, G., LUKASIEWICZ, T., AND SIMARI, G. I. Conjunctive query answering in probabilistic Datalog+/– ontologies. In *Proc. of RR* (2011), LNCS, pp. 77–92.

[9] GRIES, O., MÖLLER, R., NAFISSI, A., ROSENFELD, M., SOKOLSKI, K., AND WESSEL, M. A probabilistic abduction engine for media interpretation based on ontologies. In *Proc. of RR* (2010), pp. 182–194.

[10] GUTIÉRREZ-BASULTO, V., JUNG, J. C., LUTZ, C., AND SCHRÖDER, L. A closer look at the probabilistic description logic Prob-EL. In *Proc. of AAAI* (2011).

[11] HEINSOHN, J. Probabilistic description logics. In *Proc. of UAI* (1994), pp. 311–318.

[12] KOLLER, D., LEVY, A., AND PFEFFER, A. P-CLASSIC: A tractable probabilistic description logic. In *Proc. of AAAI* (1997), pp. 390–397.

[13] LUKASIEWICZ, T. Expressive probabilistic description logics. *Artificial Intelligence 172* (2008), 852–883.

[14] LUKASIEWICZ, T., AND STRACCIA, U. Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics 6* (2008), 291–308.

[15] LUTZ, C., AND SCHRÖDER, L. Probabilistic description logics for subjective uncertainty. In *Proc. of KR* (2010), pp. 393–403.

[16] MADHAVAN, J., KO, D., KOT, L., GANAPATHY, V., RASMUSSEN, A., AND HALEVY, A. Google's deep Web crawl. In *Proc. of VLDB* (2008), pp. 1241–1252.

[17] NOESSNER, J., AND NIEPERT, M. ELOG: a probabilistic reasoner for OWL EL. In *Proc. of RR* (2011), pp. 281–286.

[18] PATEL-SCHNEIDER, P. F., HAYES, P., AND HORROCKS, I. OWL Web Ontology Language semantics and abstract syntax. W3C Recommendation, 2004. Available at `http://www.w3.org/TR/owl-semantics/`.

[19] PEARL, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. 1988.

[20] RICHARDSON, M., AND DOMINGOS, P. Markov logic networks. *Machine Learning 62* (2006), 107–136.

[21] TODA, S. On the computational power of PP and $\oplus$P. In *Proc. of FOCS* (1989), pp. 514–519.

[22] WWW CONSORTIUM. Uncertainty Reasoning for the World Wide Web – W3C Incubator Group Report. Available at: `http://www.w3.org/2005/Incubator/urw3/XGR-urw3/`.

[23] YANG, Y., AND CALMET, J. OntoBayes: An ontology-driven uncertainty model. In *Proc. of IAWTIC* (2005), pp. 457–463.

Sparse Q-learning with Mirror Descent

Sridhar Mahadevan and **Bo Liu**
Computer Science Department
University of Massachusetts, Amherst
Amherst, Massachusetts, 01003
{mahadeva, boliu}@cs.umass.edu

Abstract

This paper explores a new framework for reinforcement learning based on *online convex optimization*, in particular *mirror descent* and related algorithms. Mirror descent can be viewed as an enhanced gradient method, particularly suited to minimization of convex functions in high-dimensional spaces. Unlike traditional gradient methods, mirror descent undertakes gradient updates of weights in both the dual space and primal space, which are linked together using a Legendre transform. Mirror descent can be viewed as a proximal algorithm where the distance generating function used is a Bregman divergence. A new class of *proximal-gradient* based temporal-difference (TD) methods are presented based on different Bregman divergences, which are more powerful than regular TD learning. Examples of Bregman divergences that are studied include p-norm functions, and Mahalanobis distance based on the covariance of sample gradients. A new family of sparse mirror-descent reinforcement learning methods are proposed, which are able to find sparse fixed points of an l_1-regularized Bellman equation at significantly less computational cost than previous methods based on second-order matrix methods. An experimental study of mirror-descent reinforcement learning is presented using discrete and continuous Markov decision processes.

Reinforcement learning (RL) models the interaction between the agent and an environment as a *Markov decision process* (MDP), a widely adopted model of sequential decision-making. The major aim of this paper is to investigate a new framework for reinforcement learning and sequential decision-making, based on ideas from *online convex optimization*, in particular mirror descent [NY83] and related algorithms [BT03, Nes09]. Mirror descent generalizes classical first-order gradient descent to non-Euclidean geometries by using a distance-generating function specific to a particular geometry. Mirror descent can be viewed as a proximal algorithm [BT03], where the distance generating function used is a Bregman divergence [Bre67]. Mirror descent is a powerful framework for convex optimization in high-dimensional spaces: an early success of this approach was its use in positron-emission tomography (PET) imaging involving an optimization problem with millions of variables [BTMN01]. The mirror descent algorithm has led to new methods for sparse classification and regression [DHS11, SST11a]. Mirror descent has also been extended from its original deterministic setting [NY83] to a stochastic approximation setting [NJLS09], which makes it highly appropriate for RL, as the standard temporal-difference (TD) learning method for solving MDPs also has its origins in stochastic approximation theory [Bor08].

l_1 regularized methods for solving MDPs have become a topic of recent attention, including a technique combining least-squares TD (LSTD) with least-angle regression (LARS) [KN09]; another method for combining l_1 regularization with approximate linear programming [PTPZ10]; finally, a linear complementarity formulation of l_1 regularization [JPWP10]. These methods involve matrix inversion, requiring near cubic complexity in the number of (active) features. In contrast, mirror-descent based RL methods promise similar performance guarantees involving only linear complexity in the number of features. Recent work in online learning [DHS11, SST11a] has explored the application of mirror descent in developing sparse methods for regularized classification and regression. This paper investigates the use of mirror-descent for sparse reinforcement learning.

1 Technical Background

1.1 Reinforcement Learning

The most popular and widely used RL method is temporal-difference (TD) learning [Sut88]. TD is a stochastic approximation approach [Bor08] to solving Markov decision

processes (MDPs), comprised of a set of states S, a set of (possibly state-dependent) actions A (A_s), a dynamical system model comprised of the transition probabilities $P^a_{ss'}$ specifying the probability of transition to state s' from state s under action a, and a reward model R. A policy $\pi : S \rightarrow A$ is a deterministic mapping from states to actions. Associated with each policy π is a value function V^π, which is a fixed point of the Bellman equation:

$$V^\pi = T^\pi(V^\pi) = R^\pi + \gamma P^\pi V^\pi \quad (1)$$

where $0 \leq \gamma < 1$ is a discount factor. Any optimal policy π^* defines the same unique optimal value function V^* that satisfies the nonlinear system of equations:

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'} \left(R^a_{ss'} + \gamma V^*(s')\right)$$

The *action value* $Q^*(s,a)$ represents a convenient reformulation of the value function, defined as the long-term value of performing a first, and then acting optimally according to V^*:

$$Q^*(s,a) = E\left(r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\right) \quad (2)$$

where r_{t+1} is the actual reward received at the next time step, and s_{t+1} is the state resulting from executing action a in state s_t. The (optimal) action value formulation is convenient because it can be approximately solved by a temporal-difference (TD) learning technique called Q-learning [Wat89]. The simplest TD method, called TD(0), estimates the value function associated with the fixed policy using a normal stochastic gradient iteration:

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t(r_t + \gamma V_t(s_{t+1}) - V_t(s_t))$$

TD(0) converges to the optimal value function V^π for policy π as long as the samples are "on-policy", namely following the stochastic Markov chain associated with the policy; and the learning rate α_t is decayed according to the Robbins-Monro conditions in stochastic approximation theory: $\sum_t \alpha_t = \infty, \sum_t \alpha_t^2 < \infty$ [BT96]. When the set of states S is large, it is often necessary to approximate the value function V using a set of handcrafted basis functions (e.g., polynomials, radial basis functions, wavelets etc.) or automatically generated basis functions [Mah09]. In linear value function approximation, the value function is assumed to lie in the linear span of the basis function matrix Φ of dimension $|S| \times p$, where it is assumed that $p \ll |S|$. Hence, $V^\pi \approx \hat{V}^\pi = \Phi w$. The equivalent TD(0) rule for linear function approximated value functions is given as:

$$w_{t+1} = \alpha_t \left(r_t + \gamma \phi(s_{t+1})^T w_t - \phi(s_t)^T w_t\right) \phi(s_t) \quad (3)$$

where the quantity in the parenthesis is the TD error. TD can be shown to converge to the fixed point of the composition of the projector Π^Φ onto the column space of Φ and the Bellman operator T^π.

l_1 regularized least-squares methods for solving MDPs attempt to find a fixed point of the l_1 penalized Bellman equation:[1] [KN09, PTPZ10, JPWP10]

$$w = f(w) = \text{argmin}_u \left(\|(R^\pi + \gamma P^\pi \Phi w - \Phi u)\|^2 + \beta \|u\|_1\right) \quad (4)$$

Unfortunately, the above l_1 regularized fixed point is not a convex optimization problem. Another way to introduce l_1 regularization is to penalize the projected Bellman residual [GS11], which yields a convex optimization problem:

$$\begin{aligned} w_\theta &= \text{argmin}_w \|R^\pi + \gamma P^\pi \Phi\theta - \Phi w\|^2 \\ \theta^* &= \text{argmin}_\theta \|\Phi w_\theta - \Phi\theta\|_2^2 + \beta\|\theta\|_1 \end{aligned} \quad (5)$$

where the first step is the projection step and the second is the fixed point step. Note that in the on-policy setting, algorithms derived from the motivation of minimizing projected Bellman residual asymptotically converge to the solution of conventional TD-learning [SMP+09]. Recent l_1-regularized RL methods, such as LARS-TD and LCP-TD, involve matrix inversion, requiring at least cubic complexity in the number of (active) features. In contrast, mirror-descent based RL methods can provide similar performance guarantees involving only linear complexity in the number of features.

1.2 Online Convex Optimization

Online convex optimization [Zin03] explores the use of first-order gradient methods for solving convex optimization problems. In this framework, at each step, the learner picks an element w_t of a convex set, in response to which an adversary picks a convex loss function $f_t(w_t)$. The aim of the learner is to select elements in such a way as to minimize its long-term regret

$$\sum_{t=1}^T f_t(w_t) - \min_w \sum_{t=1}^T f_t(w)$$

with respect to the choice it would have made in hindsight.

A fundamental problem addressed in this paper is to minimize a *sum* of two functions, as in Equation 4 and Equation 5:

$$w^* = \text{argmin}_{w \in X} \left(f(w) + g(w)\right) \quad (6)$$

where $f(w)$ is a smooth differentiable convex function, whereas g is a (possibly non-smooth) convex function that is subdifferentiable over its domain. A common example is l_1 regularized classification or regression:

$$w^* = \text{argmin}_{w \in \mathbb{R}^d} \sum_{t=1}^m L(\langle w, x_t\rangle, y_t) + \beta\|w\|_1 \quad (7)$$

[1] In practice, the full Φ matrix is never constructed, and only the p-dimensional embedding $\phi(s)$ of sampled states are explicitly formed. Also, R^π, Φ, and $P^\pi\Phi$ are approximated by $\tilde{R}$, $\tilde{\Phi}$, and $\tilde{\Phi}'$ over a given set of samples.

where $L(a, b)$ is a convex loss function, and β is a sparsity-controlling parameter. The simplest online convex algorithm is based on the classic gradient descent procedure for minimizing a function, given as:

$$w_0 \in X, w_t = \Pi_X \left(w_{t-1} - \alpha_t \nabla f(w_{t-1}) \right) : t \geq 1 \quad (8)$$

where $\Pi_X(x) = \text{argmin}_{y \in X} \|x - y\|^2$ is the projector onto set X, and α_t is a stepsize. If f is not differentiable, then the subgradient ∂f can be substituted instead, resulting in the well-known projected subgradient method, a workhorse of nonlinear programming [Ber99]. We discuss a general framework for minimizing Equation 6 next.

1.3 Proximal Mappings and Mirror Descent

The proximal mapping associated with a convex function h is defined as:

$$\text{prox}_h(x) = \text{argmin}_u \left(h(u) + \|u - x\|_2^2 \right)$$

If $h(x) = 0$, then $\text{prox}_h(x) = x$, the identity function. If $h(x) = I_C(x)$, the indicator function for a convex set C, then $\text{prox}_{I_C}(x) = \Pi_C(x)$, the projector onto set C. For learning sparse representations, the case when $h(w) = \beta \|w\|_1$ is particularly important. In this case, the entry-wise proximal operator is:

$$\text{prox}_h(w)_i = \begin{cases} w_i - \beta, & \text{if } w_i > \beta \\ 0, & \text{if } |w_i| \leq \beta \\ w_i + \beta, & \text{otherwise} \end{cases} \quad (9)$$

An interesting observation follows from noting that the projected subgradient method (Equation 8) can be written equivalently using the proximal mapping as:

$$w_{t+1} = \text{argmin}_{w \in X} \left(\langle w, \partial f(w_t) \rangle + \frac{1}{2\alpha_t} \|w - w_t\|_2^2 \right) \quad (10)$$

An intuitive way to understand this equation is to view the first term as requiring the next iterate w_{t+1} to move in the direction of the (sub) gradient of f at w_t, whereas the second term requires that the next iterate w_{t+1} not move too far away from the current iterate w_t. Note that the (sub)gradient descent is a special case of Equation (10) with Euclidean distance setup.

With this introduction, we can now introduce the main concept of *mirror descent* [NY83]. We follow the treatment in [BT03] in presenting the mirror descent algorithm as a nonlinear proximal method based on a distance generating function that is a Bregman divergence [Bre67].

Definition 1: A distance generating function $\psi(x)$ is defined as a continuously differentiable strongly convex function (with modulus σ) which satisfies:

$$\langle x' - x, \nabla \psi(x') - \nabla \psi(x) \rangle \geq \sigma \|x' - x\|^2 \quad (11)$$

Given such a function ψ, the Bregman divergence associated with it is defined as:

$$D_\psi(x, y) = \psi(x) - \psi(y) - \langle \nabla \psi(y), x - y \rangle \quad (12)$$

Intuitively, the Bregman divergence measures the difference between the value of a strongly convex function $\psi(x)$ and the estimate derived from the first-order Taylor series expansion at $\psi(y)$. Many widely used distance measures turn out to be special cases of Bregman divergences, such as Euclidean distance (where $\psi(x) = \frac{1}{2}\|x\|_2^2$) and Kullback Liebler divergence (where $\psi(x) = \sum_i x_i \log_2 x_i$, the positive entropy function). In general, Bregman divergences are non-symmetric, but projections onto a convex set with respect to a Bregman divergence is well-defined.

The general mirror descent procedure can be written as:

$$w_{t+1} = \text{argmin}_{w \in X} \left(\langle w, \partial f(w_t) \rangle + \frac{1}{\alpha_t} D_\psi(w, w_t) \right) \quad (13)$$

Notice that the squared distance term in Equation 10 has been generalized to a Bregman divergence. The solution to this optimization problem can be stated succinctly as the following generalized gradient descent algorithm, which forms the core procedure in mirror descent:

$$w_{t+1} = \nabla \psi^* \left(\nabla \psi(w_t) - \alpha_t \partial f(w_t) \right) \quad (14)$$

Here, ψ^* is the Legendre transform of the strongly convex function ψ, which is defined as

$$\psi^*(y) = \sup_{x \in X} \left(\langle x, y \rangle - \psi(x) \right)$$

It can be shown that $\nabla \psi^* = (\nabla \psi)^{-1}$ [BT03]. Mirror descent is a powerful first-order optimization method that been shown to be "universal" in that if a problem is online learnable, it leads to a low-regret solution using mirror descent [SST11b]. It is shown in [BTMN01] that the mirror descent procedure specified in Equation 14 with the Bregman divergence defined by the *p-norm* function [Gen03], defined below, can outperform regular projected subgradient method by a factor $\frac{n}{\log n}$ where n is the dimensionality of the space. For high-dimensional spaces, this ratio can be quite large.

2 Proposed Framework: Mirror Descent RL

Algorithm 1 describes the proposed mirror-descent TD(λ) method.[2] Unlike regular TD, the weights are updated using the TD error in the dual space by mapping the primal weights w using a gradient of a strongly convex function ψ. Subsequently, the updated dual weights are converted

[2] All the algorithms described extend to the action-value case where $\phi(s)$ is replaced by $\phi(s, a)$.

Algorithm 1 Adaptive Mirror Descent TD(λ)

Let π be some fixed policy for an MDP M, and s_0 be the initial state. Let Φ be some fixed or automatically generated basis.

1: **repeat**
2: Do action $\pi(s_t)$ and observe next state s_{t+1} and reward r_t.
3: Update the eligibility trace $e_t \leftarrow e_t + \lambda\gamma\phi(s_t)$
4: Update the dual weights θ_t for a linear function approximator:

$$\theta_{t+1} = \nabla\psi_t(w_t) + \alpha_t(r_t + \gamma\phi(s_{t+1})^T w_t - \phi(s_t)^T w_t)e_t$$

where ψ is a distance generating function.
5: Set $w_{t+1} = \nabla\psi_t^*(\theta_{t+1})$ where ψ^* is the Legendre transform of ψ.
6: Set $t \leftarrow t+1$.
7: **until done**.
Return $\hat{V}^\pi \approx \Phi w_t$ as the value function associated with policy π for MDP M.

back into the primal space using the gradient of the Legendre transform of ψ, namely $\nabla\psi^*$. Algorithm 1 specifies the mirror descent TD(λ) algorithm wherein each weight w_i is associated with an eligibility trace $e(i)$. For $\lambda = 0$, this is just the features of the current state $\phi(s_t)$, but for nonzero λ, this corresponds to a decayed set of features proportional to the recency of state visitations. Note that the distance generating function ψ_t is a function of time.

2.1 Choice of Bregman Divergence

We now discuss various choices for the distance generating function in Algorithm 1. In the simplest case, suppose $\psi(w) = \frac{1}{2}\|w\|_2^2$, the Euclidean length of w. In this case, it is easy to see that mirror descent TD(λ) corresponds to regular TD(λ), since the gradients $\nabla\psi$ and $\nabla\psi^*$ correspond to the identity function. A much more interesting choice of ψ is $\psi(w) = \frac{1}{2}\|w\|_q^2$, and its conjugate Legendre transform $\psi^*(w) = \frac{1}{2}\|w\|_p^2$. Here, $\|w\|_q = \left(\sum_j |w_j|^q\right)^{\frac{1}{q}}$, and p and q are conjugate numbers such that $\frac{1}{p} + \frac{1}{q} = 1$. This $\psi(w)$ leads to the p-norm link function $\theta = f(w)$ where $f : \mathbb{R}^d \to \mathbb{R}^d$ [Gen03]:

$$f_j(w) = \frac{\text{sign}(w_j)|w_j|^{q-1}}{\|w\|_q^{q-2}}, \quad f_j^{-1}(\theta) = \frac{\text{sign}(\theta_j)|\theta_j|^{p-1}}{\|\theta\|_p^{p-2}} \tag{15}$$

The p-norm function has been extensively studied in the literature on online learning [Gen03], and it is well-known that for large p, the corresponding classification or regression method behaves like a multiplicative method (e.g., the p-norm regression method for large p behaves like an exponentiated gradient method (EG) [KW95, Lit88]).

Another distance generating function is the negative entropy function $\psi(w) = \sum_i w_i \log w_i$, which leads to the entropic mirror descent algorithm [BT03]. Interestingly, this special case has been previously explored [PS95] as the exponentiated-gradient TD method, although the connection to mirror descent and Bregman divergences were not made in this previous study, and EG does not generate sparse solutions [SST11a]. We discuss EG methods vs. p-norm methods in Section 6.

2.2 Sparse Learning with Mirror Descent TD

Algorithm 2 Sparse Mirror Descent TD(λ)

1: **repeat**
2: Do action $\pi(s_t)$ and observe next state s_{t+1} and reward r_t.
3: Update the eligibility trace $e_t \leftarrow e_t + \lambda\gamma\phi(s_t)$
4: Update the dual weights θ_t:

$$\tilde{\theta}_{t+1} = \nabla\psi_t(w_t) + \alpha_t\left(r_t + \gamma\phi(s_{t+1})^T w_t - \phi(s_t)^T w_t\right)e_t$$

(e.g., $\psi(w) = \frac{1}{2}\|w\|_q^2$ is the p-norm link function).
5: Truncate weights:

$$\forall j, \;\; \theta_j^{t+1} = \text{sign}(\tilde{\theta}_j^{t+1})\max(0, |\tilde{\theta}_j^{t+1}| - \alpha_t\beta)$$

6: $w_{t+1} = \nabla\psi_t^*(\theta_{t+1})$ (e.g., $\psi^*(\theta) = \frac{1}{2}\|\theta\|_p^2$ and p and q are dual norms such that $\frac{1}{p} + \frac{1}{q} = 1$).
7: Set $t \leftarrow t+1$.
8: **until done**.
Return $\hat{V}^\pi \approx \Phi w_t$ as the l_1 penalized sparse value function associated with policy π for MDP M.

Algorithm 2 describes a modification to obtain sparse value functions resulting in a sparse mirror-descent TD(λ) algorithm. The main difference is that the dual weights θ are truncated according to Equation 9 to satisfy the l_1 penalty on the weights. Here, β is the sparsity parameter defined in Equation 7. An analogous approach was suggested in [SST11a] for l_1 penalized classification and regression.

2.3 Composite Mirror Descent TD

Another possible mirror-descent TD algorithm uses as the distance-generating function a Mahalanobis distance derived from the subgradients generated during actual trials. We base our derivation on the composite mirror-descent approach proposed in [DHS11] for classification and regression. The composite mirror-descent solves the following optimization problem at each step:

$$w_{t+1} = \text{argmin}_{x \in X}\left(\alpha_t\langle x, \partial f_t\rangle + \alpha_t\mu(x) + D_{\psi_t}(x, w_t)\right) \tag{16}$$

Here, μ serves as a fixed regularization function, such as the l_1 penalty, and ψ_t is the time-dependent distance generating function as in mirror descent. We now describe a dif-

ferent Bregman divergence to be used as the distance generating function in this method. Given a positive definite matrix A, the Mahalanobis norm of a vector x is defined as $\|x\|_A = \sqrt{\langle x, Ax\rangle}$. Let $g_t = \partial f(s_t)$ be the subgradient of the function being minimized at time t, and $G_t = \sum_t g_t g_t^T$ be the covariance matrix of outer products of the subgradients. It is computationally more efficient to use the diagonal matrix $H_t = \sqrt{\text{diag}(G_t)}$ instead of the full covariance matrix, which can be expensive to estimate. Algorithm 3 describes the adaptive subgradient mirror descent TD method.

Algorithm 3 Composite Mirror Descent TD(λ)

1: **repeat**
2: Do action $\pi(s_t)$ and observe next state s_{t+1} and reward r_t.
3: Set TD error $\delta_t = r_t + \gamma\phi(s_{t+1})^T w_t - \phi(s_t)^T w_t$
4: Update the eligibility trace $e_t \leftarrow e_t + \lambda\gamma\phi(s_t)$
5: Compute TD update $\xi_t = \delta_t e_t$.
6: Update feature covariance

$$G_t = G_{t-1} + \phi(s_t)\phi(s_t)^T$$

7: Compute Mahalanobis matrix $H_t = \sqrt{\text{diag}(G_t)}$.
8: Update the weights w:

$$w_{t+1,i} = \text{sign}(w_{t,i} - \frac{\alpha_t \xi_{t,i}}{H_{tt,i}})(|w_{t,i} - \frac{\alpha_t \xi_{t,i}}{H_{tt,i}}| - \frac{\alpha_t \beta}{H_{tt,i}})$$

9: Set $t \leftarrow t + 1$.
10: **until done**.
Return $\hat{V}^\pi \approx \Phi w_t$ as the l_1 penalized sparse value function associated with policy π for MDP M.

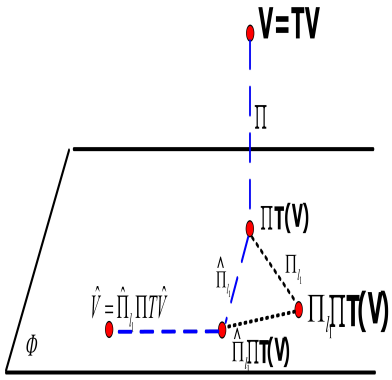
3 Convergence Analysis

Definition 2 [GLMH11]: Π_{l_1} is the l_1-regularized projection defined as: $\Pi_{l_1} y = \Phi\alpha$ such that $\alpha = \arg\min_w \|y - \Phi w\|^2 + \beta\|w\|_1$, which is a non-expansive mapping w.r.t weighted l_2 norm induced by the on-policy sample distribution setting, as proven in [GLMH11]. Let the approximation error $f(y, \beta) = \|y - \Pi_{l_1} y\|^2$.

Definition 3 (Empirical l_1-regularized projection): $\hat{\Pi}_{l_1}$ is the empirical l_1-regularized projection with a specific l_1 regularization solver, and satisfies the non-expansive mapping property. It can be shown using a direct derivation that $\hat{\Pi}_{l_1}\Pi T$ is a γ-contraction mapping. Any unbiased l_1 solver which generates intermediate sparse solution before convergence, e.g., SMIDAS solver after t-th iteration, comprises an empirical l_1-regularized projection.

Theorem 1 The approximation error $||V - \hat{V}||$ of Algorithm 2 is bounded by (ignoring dependence on π for simplicity):

Figure 1: Error Bound and Decomposition



$$||V - \hat{V}|| \leq \frac{1}{1-\gamma} \times \left(\|V - \Pi V\| + f(\Pi V, \beta) + (M-1)P(0) + \|w^*\|_1^2 \frac{M}{\alpha_t N}\right) \quad (17)$$

where $\hat{V}$ is the approximated value function after N-th iteration, i.e., $\hat{V} = \Phi w_N$, $M = \frac{2}{2 - 4\alpha_t(p-1)e}$, α_t is the step-size, $P(0) = \frac{1}{N}\sum_{i=1}^{N} \|\Pi V(s_i)\|_2^2$, s_i is the state of i-th sample, $e = d^{\frac{p}{2}}$, d is the number of features, and finally, w^* is l_1-regularized projection of ΠV such that $\Phi w^* = \Pi_{l_1}\Pi V$.

Proof: In the on-policy setting, the solution given by Algorithm 2 is the fixed point of $\hat{V} = \hat{\Pi}_{l_1}\Pi T\hat{V}$ and the error decomposition is illustrated in Figure 1. The error can be bounded by the triangle inequality

$$||V - \hat{V}|| = ||V - \Pi TV|| + ||\Pi TV - \hat{\Pi}_{l_1}\Pi TV|| + ||\hat{\Pi}_{l_1}\Pi TV - \hat{V}|| \quad (18)$$

Since $\hat{\Pi}_{l_1}\Pi T$ is a γ-contraction mapping, and $\hat{V} = \hat{\Pi}_{l_1}\Pi T\hat{V}$, we have

$$||\hat{\Pi}_{l_1}\Pi TV - \hat{V}|| = ||\hat{\Pi}_{l_1}\Pi TV - \hat{\Pi}_{l_1}\Pi T\hat{V}|| \leq \gamma||V - \hat{V}|| \quad (19)$$

So we have

$$(1-\gamma)||V - \hat{V}|| \leq ||V - \Pi TV|| + ||\Pi TV - \hat{\Pi}_{l_1}\Pi TV||$$

$\|V - \Pi TV\|$ depends on the expressiveness of the basis Φ, where if V lies in $span(\Phi)$, this error term is zero. $||\Pi TV - \Pi_{l_1}\hat{\Pi}TV||$ is further bounded by the triangle inequality

$$||\Pi TV - \hat{\Pi}_{l_1}\Pi TV|| \leq ||\Pi TV - \Pi_{l_1}\Pi TV|| + ||\Pi_{l_1}\Pi TV - \hat{\Pi}_{l_1}\Pi TV||$$

where $\|\Pi TV - \Pi_{l_1}\Pi TV\|$ is controlled by the sparsity parameter β, i.e., $f(\Pi TV, \beta) = ||\Pi TV - \Pi_{l_1}\Pi TV||$, where $\varepsilon = ||\hat{\Pi}_{l_1}\Pi TV - \Pi_{l_1}\Pi TV||$ is the approximation error depending on the quality of the l_1 solver employed. In Algorithm 2, the l_1 solver is related to the SMIDAS l_1 regularized mirror-descent method for regression and classification [SST11a]. Note that for a squared loss function

$L(\langle w, x_i \rangle, y_i) = ||\langle w, x_i \rangle - y_i||_2^2$, we have $|L'|^2 \leq 4L$. Employing the result of Theorem 3 in [SST11a], after the N-th iteration, the l_1 approximation error is bounded by

$$\varepsilon \leq (M-1)P(0) + ||w^*||_1^2 \frac{M}{\alpha_t N}, M = \frac{2}{2 - 4\alpha_t(p-1)e}$$

By rearranging the terms and applying $V = TV$, Equation (17) can be deduced.

4 Experimental Results: Discrete MDPs

Figure 2 shows that mirror-descent TD converges more quickly with far smaller Bellman errors than LARS-TD [KN09] on a discrete "two-room" MDP [MM07]. The basis matrix Φ was automatically generated as 50 proto-value functions by diagonalizing the graph Laplacian of the discrete state space connectivity graph[MM07]. The figure also shows that Algorithm 2 (sparse mirror-descent TD) scales more gracefully than LARS-TD. Note LARS-TD is unstable for $\gamma = 0.9$. It should be noted that the computation cost of LARS-TD is $O(Ndm^3)$, whereas that for Algorithm 2 is $O(Nd)$, where N is the number of samples, d is the number of basis functions, and m is the number of active basis functions. If p is linear or sublinear w.r.t d, Algorithm 2 has a significant advantage over LARS-TD.

Figure 3 shows the result of another experiment conducted to test the noise immunity of Algorithm 2 using a discrete 10×10 grid world domain with the goal set at the upper left hand corner. For this problem, 50 proto-value basis functions were automatically generated, and 450 random Gaussian mean 0 noise features were added. The sparse mirror descent TD algorithm was able to generate a very good approximation to the optimal value function despite the large number of irrelevant noisy features, and took a fraction of the time required by LARS-TD.

Figure 4 compares the performance of mirror-descent Q-learning with a fixed p-norm link function vs. a decaying p-norm link function for a 10×10 discrete grid world domain with the goal state in the upper left-hand corner. Initially, $p = O(\log d)$ where d is the number of features, and subsequently p is decayed to a minimum of $p = 2$. Varying p-norm interpolates between additive and multiplicative updates. Different values of p yield an interpolation between the truncated gradient method [LLZ09] and SMIDAS [SsT09].

Figure 5 illustrates the performance of Algorithm 3 on the two-room discrete grid world navigation task.

5 Experimental Results: Continuous MDPs

Figure 6 compares the performance of Q-learning vs. mirror-descent Q-learning for the mountain car task, which converges more quickly to a better solution with much

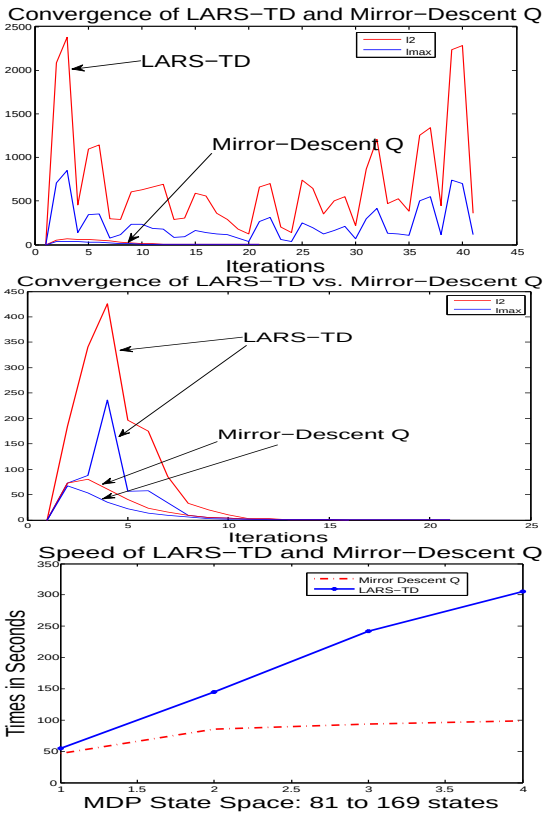


Figure 2: Mirror-descent Q-learning converges significantly faster than LARS-TD on a "two-room" grid world MDP for $\gamma = 0.9$ (top left) and $\gamma = 0.8$ (top right). The y-axis measures the l_2 (red curve) and l_∞ (blue curve) norm difference between successive weights during policy iteration. Bottom: running times for LARS-TD (blue solid) and mirror-descent Q (red dashed). Regularization $\beta = 0.01$.

lower variance. Figure 7 shows that mirror-descent Q-learning with learned diffusion wavelet bases converges quickly on the 4-dimensional Acrobot task. We found in our experiments that LARS-TD did not converge within 20 episodes (its curve, not shown in Figure 6, would be flat on the vertical axis at 1000 steps). Finally, we tested the mirror-descent approach on a more complex 8-dimensional continuous MDP. The triple-link inverted pendulum [SW01] is a highly nonlinear time-variant underactuated system, which is a standard benchmark testbed in the control community. We base our simulation using the system parameters described in [SW01], except that the action space is discretized because the algorithms described here are restricted to policies with discrete actions. There are three actions, namely $\{0, 5\text{Newton}, -5\text{Newton}\}$. The state space is 8-dimensional, consisting of the angles made to the horizontal of the three links in the arm as well as their angular velocities, the position and velocity of the cart used to balance the pendulum. The goal is to learn a policy that can balance the system with the minimum num-

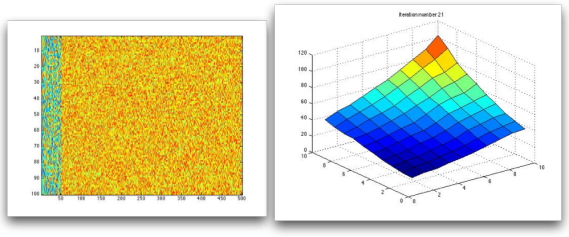


Figure 3: Sensitivity of sparse mirror-descent TD to noisy features in a grid-world domain. Left: basis matrix with the first 50 columns representing proto-value function bases and the remainder 450 bases representing mean-0 Gaussian noise. Right: Approximated value function using sparse mirror-descent TD.

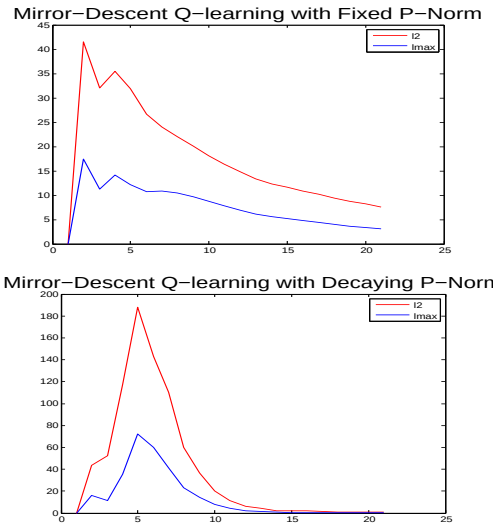


Figure 4: Left: convergence of mirror-descent Q-learning with a fixed p-norm link function. Right: decaying p-norm link function.

ber of episodes. A run is successful if it balances the inverted pendulum for the specified number of steps within 300 episodes, resulting in a reward of 0. Otherwise, this run is considered as a failure and yields a negative reward -1. The first action is chosen randomly to push the pendulum away from initial state. Two experiments were conducted on the triple-link pendulum domain with 20 runs for each experiment. As Table 1 shows, Mirror Descent Q-learning is able to learn the policy with fewer episodes and usually with reduced variance compared with regular Q-learning.

The experiment settings are Experiment 1: Zero initial state and the system receives a reward 1 if it is able to balance 10,000 steps. Experiment 2: Zero initial state and the system receives a reward 1 if it is able to balance 100,000 steps. Table 1 shows the comparison result between regular Q-learning and Mirror Descent Q-learning.

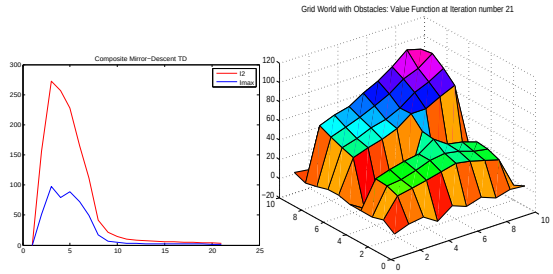


Figure 5: Left: Convergence of composite mirror-descent Q-learning on two-room gridworld domain. Right: Approximated value function, using 50 proto-value function bases.

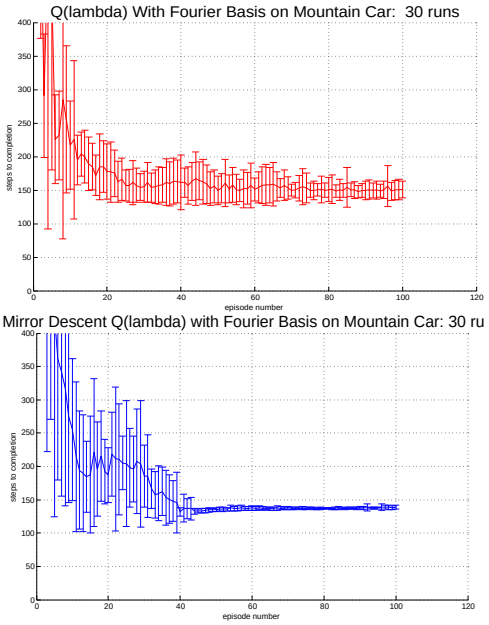


Figure 6: Top: Q-learning; Bottom: mirror-descent Q-learning with p-norm link function, both with 25 fixed Fourier bases [KOT08] for the mountain car task.

6 Comparison of Link Functions

The two most widely used link functions in mirror descent are the p-norm link function [BT03] and the relative entropy function for exponentiated gradient (EG) [KW95]. Both of these link functions offer a multiplicative update rule compared with regular additive gradient methods. The differences between these two are discussed here. Firstly, the loss function for EG is the relative entropy whereas that of the p-norm link function is the square l_2-norm function. Second and more importantly, EG does not produce sparse solutions since it must maintain the weights away from zero, or else its potential (the relative entropy) becomes unbounded at the boundary.

Another advantage of p-norm link functions over EG is that

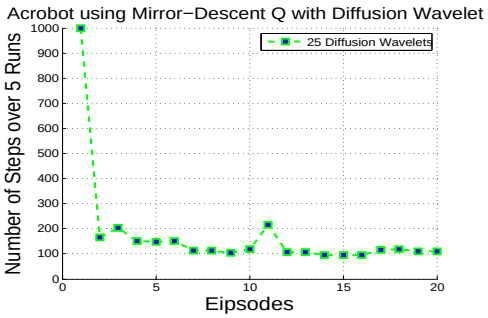


Figure 7: Mirror-descent Q-learning on the Acrobot task using automatically generated diffusion wavelet bases averaged over 5 trials.

Table 1: Results on Triple-Link Inverted Pendulum Task.

# of Episodes\Experiment	1	2
Q-learning	6.1 ± 5.67	15.4 ± 11.33
Mirror Descent Q-learning	5.7 ± 9.70	11.8 ± 6.86

the p-norm link function offers a flexible interpolation between additive and multiplicative gradient updates. It has been shown that when the features are dense and the optimal coefficients θ^* are sparse, EG converges faster than the regular additive gradient methods [KW95]. However, according to our experience, a significant drawback of EG is the overflow of the coefficients due to the exponential operator. To prevent overflow, the most commonly used technique is rescaling: the weights are re-normalized to sum to a constant. However, it seems that this approach does not always work. It has been pointed out [PS95] that in the EG-Sarsa algorithm, rescaling can fail, and replacing eligible traces instead of regular additive eligible traces is used to prevent overflow. EG-Sarsa usually poses restrictions on the basis as well. Thanks to the flexible interpolation capability between multiplicative and additive gradient updates, the p-norm link function is more robust and applicable to various basis functions, such as polynomial, radial basis function (RBF), Fourier basis [KOT08], proto-value functions (PVFs), etc.

7 Summary and Future Work

We proposed a novel framework for reinforcement learning using mirror-descent online convex optimization. Mirror Descent Q-learning demonstrates the following advantage over regular Q learning: faster convergence rate and reduced variance due to larger stepsizes with theoretical convergence guarantees [NJLS09]. Compared with existing sparse reinforcement learning algorithms such as LARS-TD, Algorithm 2 has lower sample complexity and lower computation cost, advantages accrued from the first-order mirror descent framework combined with proximal mapping [SST11a]. There are many promising future research topics along this direction. We are currently exploring a mirror-descent fast-gradient RL method, which is both convergent off-policy and quicker than fast gradient TD methods such as GTD and TDC [SMP$^+$09]. To scale to large MDPs, we are investigating hierarchical mirror-descent RL methods, in particular extending SMDP Q-learning. We are also undertaking a more detailed theoretical analysis of the mirror-descent RL framework, building on existing analysis of mirror-descent methods [DHS11, SST11a]. Two types of theoretical investigations are being explored: regret bounds of mirror-descent TD methods, extending previous results [SW94] and convergence analysis combining robust stochastic approximation [NJLS09] and RL theory [BT96, Bor08].

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research (AFOSR) under grant FA9550-10-1-0383, and the National Science Foundation under Grant Nos. NSF CCF-1025120, IIS-0534999, and IIS-0803288. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the AFOSR or the NSF.

References

[Ber99] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, September 1999.

[Bor08] V. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

[Bre67] L. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217, 1967.

[BT96] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Massachusetts, 1996.

[BT03] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 2003.

[BTMN01] A. Ben-Tal, T. Margalit, and A. Nemirovski. The ordered subsets mirror descent optimization method with applications to tomography. *SIAM Journal of Optimization*, Jan 2001.

[DHS11] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, July 2011.

[Gen03] Claudio Gentile. The robustness of the p-norm algorithms. *Mach. Learn.*, 53:265–299, December 2003.

[GLMH11] Mohammad Ghavamzadeh, Alessandro Lazaric, Remi Munos, and Matthew Hoffman. Finite-Sample Analysis of Lasso-TD . In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, 2011.

[GS11] M. Geist and B. Scherrer. L1-penalized projected bellman residual. In *Proceedings of the European Workshop on Reinforcement Learning*, 2011.

[JPWP10] J. Johns, C. Painter-Wakefield, and R. Parr. Linear complementarity for regularized policy evaluation and improvement. *In Proceedings of Advances in Neural Information Processing Systems*, 23, 2010.

[KN09] J. Zico Kolter and Andrew Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 521–528, New York, NY, USA, 2009. ACM.

[KOT08] G. Konidaris, S. Osentoski, and PS Thomas. Value function approximation in reinforcement learning using the fourier basis. *Computer Science Department Faculty Publication Series*, page 101, 2008.

[KW95] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132, 1995.

[Lit88] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. In *Machine Learning*, pages 285–318, 1988.

[LLZ09] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

[Mah09] S. Mahadevan. Learning Representation and Control in Markov Decision Processes: New Frontiers. *Foundations and Trends in Machine Learning*, 1(4):403–565, 2009.

[MM07] S. Mahadevan and M. Maggioni. Proto-Value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes. *Journal of Machine Learning Research*, 8:2169–2231, 2007.

[Nes09] Y Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, Jan 2009.

[NJLS09] A Nemirovski, A Juditsky, G Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 14(4):1574–1609, 2009.

[NY83] A. Nemirovksi and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley Press, 1983.

[PS95] D. Precup and R. Sutton. Exponentiated gradient methods for reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[PTPZ10] M. Petrik, G. Taylor, R. Parr, and S. Zilberstein. Feature selection using regularization in approximate linear programs for markov decision processes. In *ICML*, pages 871–878, 2010.

[SMP+09] Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvri, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *In Proceedings of the 26th International Conference on Machine Learning*, 2009.

[SsT09] Shai Shalev-shwartz and Ambuj Tewari. Stochastic methods for l1 regularized loss minimization. In *International Conference on Machine Learning*, pages 117–936, 2009.

[SST11a] S. Shalev-Shwartz and A. Tewari. Stochastic methods for l1 regularized loss minimization. *Journal of Machine Learning Research*, 2011.

[SST11b] N. Srebro, K. Sridharan, and A. Tewari. On the universality of online mirror descent. *Arxiv preprint arXiv:1107.4080*, Jan 2011.

[Sut88] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[SW94] Robert Schapire Schapire and Manfred K. Warmuth. On the worst-case analysis of

temporal-difference learning algorithms. In *Machine Learning*, pages 266–274. Morgan Kaufmann, 1994.

[SW01] J. Si and Y.T. Wang. Online learning control by association and reinforcement. *Neural Networks, IEEE Transactions on*, 12(2):264–276, 2001.

[Wat89] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.

[Zin03] M Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the* 20^{th} *International Conference on Machine Learning (ICML)*, 2003.

Multi-objective Influence Diagrams*

Radu Marinescu
IBM Research – Ireland
radu.marinescu@ie.ibm.com

Abdul Razak and **Nic Wilson**
Cork Constraint Computation Centre
University College Cork, Ireland
{a.razak,n.wilson}@4c.ucc.ie

Abstract

We describe multi-objective influence diagrams, based on a set of p objectives, where utility values are vectors in $\mathbb{R}^p$, and are typically only partially ordered. These can still be solved by a variable elimination algorithm, leading to a set of maximal values of expected utility. If the Pareto ordering is used this set can often be prohibitively large. We consider approximate representations of the Pareto set based on ϵ-coverings, allowing much larger problems to be solved. In addition, we define a method for incorporating user tradeoffs, which also greatly improves the efficiency.

1 INTRODUCTION

Influence diagrams [1] are a powerful formalism for reasoning with sequential decision making problems under uncertainty. They involve both chance variables, where the outcome is determined randomly based on the values assigned to other variables, and decision variables, which the decision maker can choose the value of, based on observations of some other variables. Uncertainty is represented (like in a Bayesian network) by a collection of conditional probability distributions, one for each chance variable. The overall value of an outcome is represented as the sum of a collection of utility functions.

In many situations, a decision maker has more than one objective, and mapping several objectives to a single utility scale can be problematic, since the decision maker may be unwilling or unable to provide precise tradeoffs between objectives [2, 3, 4]. We consider multi-objective influence diagrams, where utility values are now vectors in $\mathbb{R}^p$, with p being the number of objectives. Since utility values are now only partially ordered (for instance by the Pareto ordering) we no longer have a unique maximal value of expected utility, but a set of them.

The Pareto ordering on multi-objective utility is a rather weak one; the effect of this is that the set of maximal values of expected utility can often become huge. We discuss how the notion of ϵ-covering, which approximates the Pareto set, can be applied for the case of multi-objective influence diagrams. As we demonstrate experimentally, this has a major effect on the size of the undominated utility vector sets and hence on the computational efficiency and feasibility for larger problems.

We also define a simple formalism for imprecise tradeoffs; this allows the decision maker, during the elicitation stage, to specify a preference for one multi-objective utility vector over another, and uses such inputs to infer other preferences. The induced preference relation then is used to eliminate dominated utility vectors during the computation. Our experimental results indicate that the presence of even a few such imprecise tradeoffs greatly reduces the undominated set of expected utility values.

The paper is organized as follows. We first discuss the related work. Then Section 2 describes standard influence diagrams and multi-objective utility values. Section 3 defines multi-objective influence diagrams, and how variable elimination can be used to compute the set of maximal values of expected utility, up to a form of equivalence. Section 4 describes the use of the ϵ-covering of the Pareto set. Section 5 defines the formalism for tradeoffs and how the induced notion of preference dominance can be computed. Section 6 describes our experimental testing of multi-objective influence diagrams, based on the Pareto ordering, on the ϵ-covering approximation, and based on the dominance relation induced by the user tradeoffs. Section 7 concludes.

RELATED WORK: Our approach to multi-objective influence diagram computation is based on our general framework [5].

Diehl and Haimes [6] describe a computational technique for influence diagrams with multiple objectives, with the

*Abdul Razak is funded by IRCSET and IBM through the IRCSET Enterprise Partnership Scheme. This work was also supported in part by the Science Foundation Ireland under grant no. 08/PI/I1912

restriction of just a single value node (utility function). The solution method is based on influence diagram transformations [7]. Pareto dominance is used to prune sub-optimal utility vectors during the computation (with tradeoffs being taken into account at the end of the computation).

Papadimitriou and Yannakakis [8] proposed the use of the logarithmic grid in $(1+\epsilon)$ to generate an ϵ-covering of the Pareto set. Dubus et al. [9] also developed a variable elimination algorithm that uses ϵ-dominance over a graphical model (GAI network) that represents a (multi-objective) generalized decomposable additive utility function.

Zhou et al. [10] consider influence diagrams (with a unique value node) based on interval-valued utility, which is similar to bi-objective utility. They adapt Cooper's approach for solving influence diagrams based on Bayesian network algorithms [11]. Other extensions of influence diagrams include work in [12, 13, 14] which allow interval probability (but precise single-objective utility), and in [15] which considers generalized influence diagrams based on fuzzy random variables.

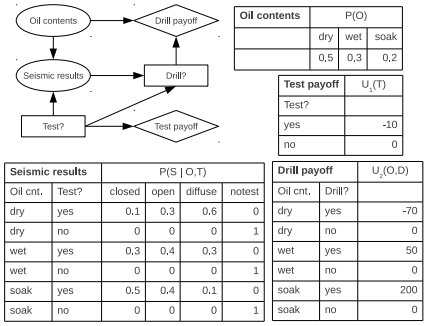
Maua et al. [16] proposed a variable elimination algorithm for solving Limited Memory Influence Diagrams (LIMIDs). Although they consider standard totally ordered utilities, their algorithm also manipulates partially ordered sets: in their case, sets of functions that are used to represent the effect of different undominated policies.

Our method for incorporating tradeoffs relates to *convex cones*, as shown by Theorem 3. A general approach to multi-objective preferences based on cones has been developed by Yu and others [17, 18].

2 BACKGROUND

2.1 INFLUENCE DIAGRAMS

An *influence diagram* (ID) [1] is defined by a tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U} \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of *chance variables* which specify the uncertain decision environment and $\mathbf{D} = \{D_1, \ldots, D_m\}$ is a set of *decision variables* which specify the possible decisions to be made in the domain. The chance variables are further divided into *observable* meaning they will be observed during execution, or *unobservable*. As in Bayesian networks [19], each chance variable $X_i \in \mathbf{X}$ is associated with a *conditional probability table* (CPT) $P_i = P(X_i | pa(X_i))$, where $P_i \in \mathbf{P}$ and $pa(X_i) \subseteq \mathbf{X} \cup \mathbf{D} \setminus \{X_i\}$. Each decision variable $D_k \in \mathbf{D}$ has a parent set $pa(D_k) \subseteq \mathbf{X} \cup \mathbf{D} \setminus \{D_k\}$, denoting the variables whose values will be known at the time of the decision and may affect directly the decision. *Non-forgetting* is typically assumed for an influence diagram, meaning that a decision node and its parents are parents to all subsequent decisions. The *utility* (or *reward*) *functions* $\mathbf{U} = \{U_1, \ldots, U_r\}$ are defined over subsets of



Oil contents	P(O)		
	dry	wet	soak
	0,5	0,3	0,2

Test payoff	U_1(T)
Test?	
yes	-10
no	0

Seismic results		P(S | O,T)			
Oil cnt.	Test?	closed	open	diffuse	notest
dry	yes	0.1	0.3	0.6	0
dry	no	0	0	0	1
wet	yes	0,3	0,4	0,3	0
wet	no	0	0	0	1
soak	yes	0.5	0.4	0.1	0
soak	no	0	0	0	1

Drill payoff		U_2(O,D)
Oil cnt.	Drill?	
dry	yes	-70
dry	no	0
wet	yes	50
wet	no	0
soak	yes	200
soak	no	0

Figure 1: A simple influence diagram with two decisions.

variables $Q = \{Q_1, \ldots, Q_r\}$, $Q_i \subseteq \mathbf{X} \cup \mathbf{D}$, called *scopes*, and represent the preferences of the decision maker.

The graph of an ID contains nodes for chance variables (circled), decision variables (boxed) and for the utility components (diamond). For each chance or decision node there is an arc directed from each of its parent variables toward it, and there is a directed arc from each variable in the scope of a utility function toward its utility node.

The decision variables in an influence diagram are typically assumed to be temporally ordered (also known as *regularity*). Let $D_1, D_2, ..., D_m$ be the order in which the decisions are to be made. The chance variables can be partitioned into a collection of disjoint sets $\mathbf{I}_0, \mathbf{I}_1, \ldots, \mathbf{I}_m$. For each k, where $0 < k < m$, $\mathbf{I}_k$ is the set of chance variables that are observed between D_k and D_{k+1}. $\mathbf{I}_0$ is the set of initial evidence variables that are observed before D_1. $\mathbf{I}_m$ is the set of chance variables left unobserved when the last decision D_m is made. This induces a strict partial order $\prec$ over $\mathbf{X} \cup \mathbf{D}$, given by: $\mathbf{I}_0 \prec D_1 \prec \mathbf{I}_1 \prec \cdots \prec D_m \prec \mathbf{I}_m$ [20].

A *policy* (or *strategy*) for an influence diagram is a list of decision rules $\Delta = (\delta_1, \ldots, \delta_m)$ consisting of one rule for each decision variable. A *decision rule* for the decision $D_k \in \mathbf{D}$ is a mapping $\delta_k : \Omega_{pa(D_k)} \rightarrow \Omega_{D_k}$, where for a set $S \subseteq \mathbf{X} \cup \mathbf{D}$, Ω_S is the Cartesian product of the individual domains of the variables in S. Therefore, a policy Δ determines a value for each decision variable D_i (which depends on the parents set $pa(D_i)$). Given a utility function U, a policy Δ yields a utility function $[U]_\Delta$ that involves no decision variables, by assigning their values using Δ. The expected utility given policy Δ is $EU_\Delta = \sum_{\mathbf{X}} [(\prod_{i=1}^{n} P_i \times \sum_{j=1}^{r} U_j)]_\Delta$. Solving an influence diagram means finding an *optimal policy* that maximizes the expected utility, i.e., to find $\arg\max_\Delta EU_\Delta$. The maximum expected utility can be shown to be equal to:

$$\sum_{\mathbf{I}_0} \max_{D_1} \cdots \sum_{\mathbf{I}_{m-1}} \max_{D_m} \sum_{\mathbf{I}_m} \left(\prod_{i=1}^{n} P_i \times \sum_{j=1}^{r} U_j \right) \quad (1)$$

Example 1 *Figure 1 shows the influence diagram of the oil wildcatter problem (adapted from [21]). An oil wildcatter must decide either to* drill *or* not to drill *for oil at a*

specific site. Before drilling, a seismic test *could help determine the geological structure of the site. The test results can show a* closed *reflection pattern (indication of significant oil), an* open *pattern (indication of some oil), or a* diffuse *pattern (almost no hope of oil). The special value* notest *is used if no seismic test is performed. There are therefore two decision variables, T (Test) and D (Drill), and two chance variables S (Seismic results) and O (Oil contents). The probabilistic knowledge consists of the conditional probability tables $P(O)$ and $P(S|O,T)$, while the utility function is the sum of $U_1(T)$ and $U_2(O,D)$. The optimal policy is to perform the test and to drill only if the test results show an open or a closed pattern. The expected utility of this policy is 22.5.*

Several exact methods have been proposed over the past decades for solving influence diagrams using local computations [7, 22, 23, 20, 24, 25]. These methods adapted classical *variable elimination* techniques, which compute a type of marginalization over a combination of local functions, in order to handle the multiple types of information (probabilities and utilities), marginalization ($\sum$ and $\max$) and combination ($\times$ for probabilities, $+$ for utilities) involved in influence diagrams. Since the alternation of $\sum$ and $\max$ in Equation 1 does not commute in general, it prevents the solution technique from eliminating variables in any ordering. Therefore, the computation dictated by Equation 1 must be performed along a *legal elimination ordering* that respects $\prec$, namely the reverse of the elimination ordering is some extension of $\prec$ to a total order [20, 24].

2.2 MULTI-OBJECTIVE UTILITY VALUES

In many real-world situations it may not be possible for the decision maker to map the various possible consequences of a set of actions to the same scale of utility in a way that avoids essentially arbitrary decisions. Hence, it is natural to consider multi-objective or multi-attribute utility functions to cope with multiple and non-commensurate utility scales on which the decision maker's preferences are expressed.

Consider a situation with p objectives (or attributes). A *multi-objective utility value* is characterized by a vector $\vec{u} = (u_1, \ldots, u_p) \in \mathbb{R}^p$, where u_i represents the utility with respect to objective $i \in \{1, \ldots, p\}$. We assume the standard pointwise arithmetic operations, namely $\vec{u} + \vec{v} = (u_1 + v_1, \ldots, u_p + v_p)$ and $q \times \vec{u} = (q \times u_1, \ldots, q \times u_p)$, where $q \in \mathbb{R}$. The comparison of utility values reduces to that of their corresponding p-dimensional vectors.

We are interested in partial orders $\succcurlyeq$ on $\mathbb{R}^p$ satisfying the following two monotonicity properties, where $\vec{u}, \vec{v}, \vec{w} \in \mathbb{R}^p$ are arbitrary vectors.

[Independence:] If $\vec{u} \succcurlyeq \vec{v}$ then $\vec{u} + \vec{w} \succcurlyeq \vec{v} + \vec{w}$

[Scale-Invariance:] If $\vec{u} \succcurlyeq \vec{v}$ and $q \in \mathbb{R}$, $q \geq 0$ then $q \times \vec{u} \succcurlyeq q \times \vec{v}$.

An important example of such a partial order is the weak Pareto order.

DEFINITION **1 (weak Pareto ordering)** *Let $\vec{u}, \vec{v} \in \mathbb{R}^p$ such that $\vec{u} = (u_1, \ldots, u_p)$ and $\vec{v} = (v_1, \ldots, v_p)$. We define the binary relation $\geq$ on $\mathbb{R}^p$ by $\vec{u} \geq \vec{v} \iff \forall i \in \{1, \ldots, p\}\ u_i \geq v_i$.*

Given $\vec{u}, \vec{v} \in \mathbb{R}^p$, if $\vec{u} \succcurlyeq \vec{v}$ then we say that $\vec{u}$ *dominates* $\vec{v}$. As usual, the symbol $\succ$ refers to the asymmetric part of $\succcurlyeq$, namely $\vec{u} \succ \vec{v}$ if and only if $\vec{u} \succcurlyeq \vec{v}$ and it is not the case that $\vec{v} \succcurlyeq \vec{u}$. In particular, relation $\geq$ (resp. $>$) is also called *weak Pareto dominance* (resp. *Pareto dominance*).

DEFINITION **2 (maximal/Pareto set)** *Given a partial order $\succcurlyeq$ and a finite set of utility vectors $\mathcal{U} \subseteq \mathbb{R}^p$, we define the* maximal set, *denoted by $\max_{\succcurlyeq}(\mathcal{U})$, to be the set consisting of the undominated elements in $\mathcal{U}$, i.e., $\max_{\succcurlyeq}(\mathcal{U}) = \{\vec{v} \in \mathcal{U} \mid \nexists \vec{v} \in \mathcal{U}, \vec{v} \succ \vec{u}\}$. When $\succcurlyeq$ is the weak Pareto ordering $\geq$, we call $\max_{\succcurlyeq}(\mathcal{U})$ the* Pareto set.

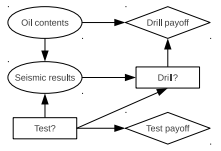
3 MULTI-OBJECTIVE INFLUENCE DIAGRAMS

In this section, we introduce the extension of the standard influence diagram model to include multiple objectives. Towards this goal we consider a multi-objective utility function that is additively decomposable [26]. For simplicity and without loss of generality we assume that all objectives are to be maximized. We next define formally the graphical model and then derive a sequential variable elimination algorithm for evaluating the model.

3.1 THE GRAPHICAL MODEL

A *multi-objective influence diagram* (MOID) extends the standard influence diagram by allowing a multi-objective utility function defined on $p > 1$ objectives. The graphical structure of a MOID is identical to that of a standard ID, namely it is a directed acyclic graph containing *chance nodes* (drawn as circles) for the random discrete variables $\mathbf{X}$, *decision nodes* (drawn as rectangles) for the decision variables $\mathbf{D}$, and *utility nodes* (drawn as diamonds) for the local utility functions $\mathbf{U}$ of the decision maker. The directed arcs in the MOID represent the same dependencies between the variables as in the standard model. Each chance node $X_i \in \mathbf{X}$ is associated with a conditional probability table $P(X_i|pa(X_i)) : \Omega_{X_i \cup pa(X_i)} \rightarrow [0, 1]$. The utility functions $U_j \in \mathbf{U}$ represent the decision maker's preferences with respect to each of the p objectives, namely $U_j : \Omega_{Q_j} \rightarrow \mathbb{R}^p$, where Q_j is the scope of U_j.

A *policy* for a MOID is a sequence $\Delta = (\delta_1, \ldots, \delta_m)$, where each δ_i is a function from $\Omega_{pa(D_i)} \rightarrow \Omega_{D_i}$. Clearly, given a policy Δ we have that $EU_\Delta \subseteq \mathbb{R}^p$. Solving a multi-objective influence diagram means finding the set of



Oil contents	P(O)		
	dry	wet	soak
	0,5	0,3	0,2

Test payoff	U_1(T)
Test?	
yes	{(-10,10)}
no	{(0,0)}

Seismic results		P(S | O,T)			
Oil cnt.	Test?	closed	open	diffuse	notest
dry	yes	0.1	0.3	0.6	0
dry	no	0	0	0	1
wet	yes	0.3	0.4	0.3	0
wet	no	0	0	0	1
soak	yes	0.5	0.4	0.1	0
soak	no	0	0	0	1

Drill payoff		U_2(O,D)
Oil cnt.	Drill?	
dry	yes	{(-70,18)}
dry	no	{(0,0)}
wet	yes	{(50,12)}
wet	no	{(0,0)}
soak	yes	{(200,8)}
soak	no	{(0,0)}

Figure 2: A bi-objective influence diagram.

policies that generate maximal values of expected utility, i.e., values of utility in the set $\max_{\succcurlyeq}\{EU_\Delta \mid \text{policies } \Delta\}$. We say that a policy Δ is *optimal* if the corresponding expected utility EU_Δ is undominated.

Example 2 *Figure 2 displays a bi-objective influence diagram for the decision problem from Example 1. We consider a utility function with two attributes representing the testing/drilling* payoff *and* damage *to the environment, respectively. The utility of testing is* $(-10, 10)$*, whereas the utility of drilling is* $(-70, 18)$*,* $(50, 12)$*,* $(200, 8)$ *for a dry, wet and soaking hole, respectively. The aim is to find optimal policies that maximize the payoff and minimize the damage to environment. The dominance relation is defined in this case by* $\vec{u} \geq \vec{v} \Leftrightarrow u_1 \geq v_1$ *and* $u_2 \leq v_2$ *(e.g.,* $(10, 2) \geq (8, 4)$ *and* $(10, 2) \not\geq (8, 1)$*). The Pareto set* $\max_{\geq}\{EU_\Delta \mid$ *policies* $\Delta\}$ *contains 4 elements, i.e.,* $\{(22.5, 17.56),\ (20, 14.2),\ (11, 12.78),\ (0, 0)\}$*, corresponding to the four optimal policies shown below (we show how to obtain these in Section 3.4):*

	Δ_1	Δ_2	Δ_3	Δ_4
δ_T	*yes*	*no*	*yes*	*no*
δ_D	*yes (S = closed)* *yes (S = open)* *no (S = diffuse)*	*yes (S = notest)*	*yes (S = closed)* *no (S = open)* *no (S = diffuse)*	*no (S = notest)*
EU_{Δ_i}	$\{(22.5, 17.56)\}$	$\{(20, 14.2)\}$	$\{(11, 12.78)\}$	$\{(0, 0)\}$

Because we are considering partially ordered utilities, the $\max$ operator does not necessarily lead to a unique element. This means that we need to extend the arithmetic operations, addition, multiplication and $\max$, to finite sets of utility values.

3.2 ARITHMETIC OPERATIONS AND DISTRIBUTIVITY PROPERTIES

In this section we assume a partial order $\succcurlyeq$ on $\mathbb{R}^p$ that satisfies Independence and Scale-Invariance. In particular, $\succcurlyeq$ can be the weak Pareto ordering.

Given two finite sets $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$ and $q \geq 0$, we define the summation and multiplication operations as $\mathcal{U} + \mathcal{V} = \{\vec{u} + \vec{v} \mid \vec{u} \in \mathcal{U}, \vec{v} \in \mathcal{V}\}$ and $q \times \mathcal{U} = \{q \times \vec{u} \mid \vec{u} \in \mathcal{U}\}$, respectively. The maximization operation is defined as $\max(\mathcal{U}, \mathcal{V}) = \max_{\succcurlyeq}(\mathcal{U} \cup \mathcal{V})$. It is easy to see that $+$, $\times$ and $\max$ are commutative and associative.

To ensure the correctness of a variable elimination computation (such as the one in Algorithm 1) we need some distributivity properties. However, the important property $(q_1 + q_2) \times \mathcal{U} = q_1 \times \mathcal{U} + q_2 \times \mathcal{U}$ does not always hold. As a simple example, let $q_1 = q_2 = 0.5$ and consider a set of bi-objective utility vectors $\mathcal{U} = \{(1, 0), (0, 1)\}$. Using the point-wise operations on pairs of real numbers we have that $(q_1 + q_2) \times \mathcal{U}$ yields the set $\{(1, 0), (0, 1)\}$, whereas $(q_1 \times \mathcal{U}) + (q_2 \times \mathcal{U}) = \{(0.5, 0), (0, 0.5)\} + \{(0.5, 0), (0, 0.5)\} = \{(1, 0), (0.5, 0.5), (0, 1)\}$. We will see next that this distributivity property does hold if we restrict to convex sets.

3.3 EQUIVALENT SETS OF UTILITY VALUES

For $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$, we say that $\mathcal{U} \succcurlyeq \mathcal{V}$ if every element of $\mathcal{V}$ is (weakly) dominated by some element of $\mathcal{U}$ (so that $\mathcal{U}$ contains as least as large elements as $\mathcal{V}$), namely if for all $\vec{v} \in \mathcal{V}$ there exists $\vec{u} \in \mathcal{U}$ with $\vec{u} \succcurlyeq \vec{v}$. We also define an equivalence relation $\approx$ between two finite sets $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$ by $\mathcal{U} \approx \mathcal{V}$ if and only if $\mathcal{U} \succcurlyeq \mathcal{V}$ and $\mathcal{V} \succcurlyeq \mathcal{U}$.

Given a set $\mathcal{U} \subseteq \mathbb{R}^p$, we define its *convex closure* $\mathcal{C}(\mathcal{U})$ to consist of every element of the form $\sum_{j=1}^{k}(q_j \times \vec{u}^j)$, where k is an arbitrary natural number, each $\vec{u}^j \in \mathcal{U}$, each $q_j \geq 0$ and $\sum_{j=1}^{k} q_j = 1$, respectively.

Given $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$, we define the equivalence relation $\equiv$ by $\mathcal{U} \equiv \mathcal{V}$ if and only if $\mathcal{C}(\mathcal{U}) \approx \mathcal{C}(\mathcal{V})$. Therefore, two sets of utility vectors are considered equivalent if, for every convex combination of elements of one, there is a convex combination of elements of the other which is at least as good (with respect to the partial order $\succcurlyeq$ on $\mathbb{R}^p$). The following result shows, in particular, that the operations on sets of utility values respects the equivalence relation.

PROPOSITION **1** *Let* $\mathcal{U}, \mathcal{V}, \mathcal{W} \subseteq \mathbb{R}^p$ *be finite sets and let* $q \geq 0$*. The following properties hold: (1)* $\mathcal{U} \equiv \max_{\succcurlyeq}(\mathcal{U})$*; (2) if* $\mathcal{U} \equiv \mathcal{V}$ *then* $q \times \mathcal{U} \equiv q \times \mathcal{V}$*,* $\mathcal{U} + \mathcal{W} \equiv \mathcal{V} + \mathcal{W}$ *and* $\max(\mathcal{U}, \mathcal{W}) \equiv \max(\mathcal{V}, \mathcal{W})$*.*

We can show now that the distributivity, and other properties we require, hold with respect to the $\equiv$ relation between finite sets of partially ordered utility values.

THEOREM **1** *Let* $\succcurlyeq$ *be a partial order on* $\mathbb{R}^p$ *satisfying Independence and Scale-Invariance. Then, for all* $q, q_1, q_2 \geq 0$ *and for all finite sets* $\mathcal{U}, \mathcal{V}, \mathcal{W} \subseteq \mathbb{R}^p$*, we have that:*
(i) $q \times (\mathcal{U} + \mathcal{V}) = q \times \mathcal{U} + q \times \mathcal{V}$*;*
(ii) $(q_1 + q_2) \times \mathcal{U} \equiv (q_1 \times \mathcal{U}) + (q_2 \times \mathcal{U})$*;*
(iii) $q_1 \times (q_2 \times \mathcal{U}) = (q_1 \times q_2) \times \mathcal{U}$*;*
(iv) $\max(q \times \mathcal{U}, q \times \mathcal{V}) = q \times \max(\mathcal{U}, \mathcal{V})$*;*
(v) $\max(\mathcal{U} + \mathcal{W}, \mathcal{V} + \mathcal{W}) \equiv \max(\mathcal{U}, \mathcal{V}) + \mathcal{W}$*.*

This implies that variable elimination can be used to gen-

Algorithm 1: ELIM-MOID

Data: A MOID $\langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U} \rangle$ with $p > 1$ objectives, a legal elimination ordering of the variables $\tau = (Y_1, \ldots, Y_t)$
Result: An optimal policy Δ

```
// partition the functions into buckets
```
1 **for** $l = t$ *downto 1* **do**
2 place in $bucket s[l]$ all remaining components in $\mathbf{P}$ and $\mathbf{U}$ that contain variable Y_l in their scope

```
// top-down step
```
3 **for** $l = t$ *downto 1* **do**
4 let $\Phi^l = \{\phi_1, \ldots, \phi_j\}$ and $\Psi^l = \{\psi_1, \ldots, \psi_k\}$ be the probability and utility components in $bucket s[l]$
5 **if** Y_l *is a chance variable* **then**
6 $\phi^l \leftarrow \sum_{Y_l} \prod_{i=1}^{j} \phi_i$
7 $\psi^l \leftarrow (\phi^l)^{-1} \times \sum'_{Y_l}((\prod_{i=1}^{j} \phi_i) \times (\sum_{j=1}^{k} \psi_j))$
8 **else if** Y_l *is a decision variable* **then**
9 $\phi^l \leftarrow \max_{Y_l} \prod_{i=1}^{j} \phi_i$
10 $\psi^l \leftarrow \max_{Y_l}((\prod_{i=1}^{j} \phi_i) \times (\sum_{j=1}^{k} \psi_j))$
11 place each ϕ^l and ψ^l in the bucket of the highest-index variable in its scope

```
// bottom-up step
```
12 **for** $l = 1$ *to* t **do**
13 **if** Y_l *is a decision variable* **then**
14 $\delta_l \leftarrow \arg\max_{Y_l}((\prod_{i=1}^{j} \phi_i) \times (\sum_{j=1}^{k} \psi_j))$
15 $\Delta \leftarrow \Delta \cup \delta_l$
16 **return** Δ

erate the set of maximal values of expected utility, up to equivalence (see also [5] for more details).

3.4 VARIABLE ELIMINATION

As well as operation $+$ on sets of utilities, we define operation $+'$ on finite sets of utility values by $\mathcal{U} +' \mathcal{V} = \max_{\succcurlyeq}(\mathcal{U} + \mathcal{V})$. Theorem 1 allows us to apply an iterative variable elimination procedure along a legal elimination ordering where chance variables are eliminated by $+'$, decision variables by $\max$, and the probability and (set-valued) utility functions are combined by $\times$ and $+$, respectively. The set of maximal expected utility values is equivalent to $\sum'_{\mathbf{I}_0} \max_{D_1} \cdots \max_{D_m} \sum'_{\mathbf{I}_m} \left(\prod_{i=1}^{n} P_i \times \sum_{j=1}^{r} U_j \right)$.

The variable elimination algorithm, called ELIM-MOID, is described by Algorithm 1. It is based on Dechter's bucket elimination framework [24] and computes the maximal set $\max_{\succcurlyeq}\{EU_\Delta | \text{ policies } \Delta\}$ as well as an optimal policy (the algorithm can be easily instrumented to produce the entire set of optimal policies). Given a legal elimination ordering $\tau = Y_1, \ldots, Y_t$, the input functions are partitioned into a bucket structure, called *buckets*, such that each bucket is associated with a single variable Y_l and contains all input probability and utility functions whose highest variable in their scope is Y_l.

ELIM-MOID processes each bucket, top-down from the last to the first, by a variable elimination procedure that computes new probability (denoted by ϕ) and utility (denoted by ψ) components which are then placed in corresponding lower buckets (lines 3-11). For a chance variable Y_l, the ϕ-message is generated by multiplying all probability components in that bucket and eliminating Y_l by summation. The ψ-message is computed as the average utility in that bucket, normalized by the bucket's compiled ϕ (here Y_l is eliminated by $\sum'$). For a decision variable Y_l, we compute the ϕ and ψ components in a similar manner and eliminate Y_l by maximization. In this case, the product of probability components in the bucket is a constant when viewed as a function of the bucket's decision variable and therefore the compiled ϕ-message is a constant as well [20, 5].

In the second, bottom-up step, the algorithm generates an optimal policy (lines 12-16). The decision buckets are processed in reversed order, from the first variable to the last. Each decision rule is generated by taking the argument of the maximization operator applied over the combination of probability and utility components in the respective bucket, for each combination of the variables in the bucket's scope (i.e., the union of the scopes of all functions in the bucket minus Y_l) while remembering the values assigned to earlier decisions. Ties are broken uniformly at random.

As is usually the case with bucket elimination algorithms, the complexity of ELIM-MOID can be bounded exponentially (time and space) by the width of the ordered induced graph that reflects the execution of the algorithm (i.e., induced width of the legal elimination ordering) [24]. Since the utility values are vectors in $\mathbb{R}^p$, it is not easy to predict the size of the undominated set of expected utility values.

4 APPROXIMATING THE PARETO SET

In this section, we assume without loss of generality a weak Pareto ordering on $\mathbb{R}^p_+$ because the proposed approximation method relies on a log transformation of the solution space as we will see next. The cardinality of the Pareto set $\max_{\geq}\{EU_\Delta : \text{ policies } \Delta\}$ (and also the number of optimal policies) can often get very large. What would then be desirable for the decision maker is an approximation of the Pareto set that *approximately* dominates (or covers) all elements in $\{EU_\Delta : \text{ policies } \Delta\}$ and is of considerably smaller size. This can be achieved by considering the notion of ϵ-covering of the Pareto set which is based on ϵ-*dominance* between utility values [8].

DEFINITION **3 (ϵ-dominance)** *For any finite $\epsilon > 0$, the* ϵ-dominance *relation is defined on positive vectors of $\mathbb{R}^p_+$ by $\vec{u} \geq_\epsilon \vec{v} \Leftrightarrow (1 + \epsilon) \cdot \vec{u} \geq \vec{v}$.*

DEFINITION **4 (ϵ-covering)** *Let $\mathcal{U} \subseteq \mathbb{R}^p_+$ and $\epsilon > 0$. Then a set $\mathcal{U}_\epsilon \subseteq \mathcal{U}$ is called an* ϵ-approximate Pareto set *or an* ϵ-covering*, if any vector $\vec{v} \in \mathcal{U}$ is ϵ-dominated by at least one vector $\vec{u} \in \mathcal{U}_\epsilon$, i.e., $\forall \vec{v} \in \mathcal{U}\ \exists \vec{u} \in \mathcal{U}_\epsilon$ such that $\vec{u} \geq_\epsilon \vec{v}$.*

Algorithm 2: $(\epsilon, \lambda)-$COVERING($\mathcal{U}$)

$\Gamma \leftarrow \emptyset$; $\mathcal{V} \leftarrow \emptyset$;
foreach $\vec{u} \in \mathcal{U}$ **do**
  if $\varphi_\lambda(\vec{u}) \notin \Gamma$ **then**
    remove from Γ all $\varphi_\lambda(\vec{v})$ such that $\varphi_\lambda(\vec{u}) \geq \varphi_\lambda(\vec{v})$;
    $\Gamma \leftarrow \Gamma \cup \{\varphi_\lambda(\vec{u})\}$;
foreach $\varphi_\lambda(\vec{u}) \in \Gamma$ **do** $\mathcal{V} \leftarrow \mathcal{V} \cup \{\vec{u}\}$;
return $\mathcal{V}$;

The set $\mathcal{U}_\epsilon$ is not unique. However, it is possible to compute an ϵ-covering of a finite set $\mathcal{U} \subseteq \mathbb{R}^p_+$ by mapping each vector $\vec{u} \in \mathcal{U}$ onto a hyper-grid using the log transformation $\varphi : \mathbb{R}^p_+ \rightarrow \mathbb{Z}^p_+$, defined by $\varphi(\vec{u}) = (\varphi(u_1), \ldots, \varphi(u_p))$ where $\forall i$, $\varphi(u_i) = \lceil \log u_i / \log(1+\epsilon) \rceil$ [8]. By definition, we have that:

PROPOSITION **2** $\forall \vec{u}, \vec{v} \in \mathbb{R}^p_+$, $\varphi(\vec{u}) \geq \varphi(\vec{v}) \Rightarrow \vec{u} \geq_\epsilon \vec{v}$.

It is easy to see that any cell of the grid represents a different class of vectors having the same image through φ. Based on Proposition 2, any vector belonging to a given cell ϵ-dominates any other vector of that cell. Therefore, for any finite ϵ, we can obtain a valid ϵ-covering of $\mathcal{U}$ by choosing one representative element in each cell and by keeping only undominated cells occupied. For example, if we restrict the vectors in $\mathcal{U}$ to be bounded by: $1 \leq u_i \leq B$ for all $i \in \{1, \ldots, p\}$, then the size of $\mathcal{U}_\epsilon$ is polynomial in $\log B$ and $1/\epsilon$ (see also [8] for more details).

Example 3 *Let $\mathcal{U} = \{\vec{u}, \vec{v}\}$ such that $\vec{u} = (3.1, 2.9)$ and $\vec{v} = (3, 3.05)$. Clearly, $\vec{u} \not\geq \vec{v}$ and $\vec{v} \not\geq \vec{u}$. Set $\epsilon = 0.1$. We have that $\varphi(\vec{u}) = \varphi(\vec{v}) = (12, 12)$, and it is easy to verify that $\vec{u} \geq_\epsilon \vec{v}$ and $\vec{v} \geq_\epsilon \vec{u}$. Therefore, $\mathcal{U}_\epsilon = \{\vec{u}\}$ is a valid ϵ-covering of $\mathcal{U}$, as is $\{\vec{v}\}$.*

We next extend algorithm ELIM-MOID to compute an ϵ-covering of the expected utility set $\{EU_\Delta : \text{ policies } \Delta\}$. However, it is not possible to just replace Pareto dominance with ϵ-dominance at each variable elimination step and still guarantee a valid ϵ-covering because ϵ-dominance is not a transitive relation (e.g., if $\vec{u} \geq_\epsilon \vec{v}$ and $\vec{v} \geq_\epsilon \vec{w}$, we only have that $\vec{u} \geq (1+\epsilon)^2 \cdot \vec{w}$). To overcome this difficulty, we use a finer dominance relation, defined as follows [9].

DEFINITION **5 (**(ϵ, λ)**-dominance)** *For any finite $\epsilon > 0$ and $\lambda \in (0, 1)$, the* (ϵ, λ)-dominance *relation is defined on positive vectors of $\mathbb{R}^p_+$ by $\vec{u} \geq^\lambda_\epsilon \vec{v} \Leftrightarrow (1+\epsilon)^\lambda \cdot \vec{u} \geq \vec{v}$. Given a set $\mathcal{U} \subseteq \mathbb{R}^p_+$, a subset $\mathcal{U}_{(\epsilon,\lambda)} \subseteq \mathcal{U}$ is called an (ϵ, λ)-covering, if $\forall \vec{v} \in \mathcal{U}$ $\exists \vec{u} \in \mathcal{U}_{(\epsilon,\lambda)}$ such that $\vec{u} \geq^\lambda_\epsilon \vec{v}$.*

Algorithm 2 computes an (ϵ, λ)-covering of a finite set $\mathcal{U} \subseteq \mathbb{R}^p_+$ by using the log grid mapping $\varphi_\lambda : \mathbb{R}^p_+ \rightarrow \mathbb{Z}^p_+$ defined by $\varphi_\lambda(\vec{u}) = (\varphi_\lambda(u_1), \ldots, \varphi_\lambda(u_p))$ where $\forall i$, $\varphi_\lambda(u_i) = \lceil \log u_i / \log(1+\epsilon)^\lambda \rceil$. It is easy to see that:

PROPOSITION **3** $\forall \vec{u}, \vec{v} \in \mathbb{R}^p_+$, $\varphi_\lambda(\vec{u}) \geq \varphi_\lambda(\vec{v}) \Rightarrow \vec{u} \geq^\lambda_\epsilon \vec{v}$.

PROPOSITION **4** *Let $\vec{u}, \vec{v}, \vec{w} \in \mathbb{R}^p_+$ and $\lambda, \lambda' \in (0, 1)$. The following properties hold: (i) if $\vec{u} \geq^\lambda_\epsilon \vec{v}$ then $\vec{u} + \vec{w} \geq^\lambda_\epsilon \vec{v} + \vec{w}$, and if $\vec{u} \geq^\lambda_\epsilon \vec{v}$ and $q \geq 0$ then $q \cdot \vec{u} \geq^\lambda_\epsilon q \cdot \vec{v}$; (ii) if $\vec{u} \geq^\lambda_\epsilon \vec{v}$ and $\vec{v} \geq^{\lambda'}_\epsilon \vec{w}$ then $\vec{u} \geq^{\lambda+\lambda'}_\epsilon \vec{w}$.*

We define operations $\max^*$ and $+^*$ on finite sets of utility values by $\max^*(\mathcal{U}, \mathcal{V}) = \max_{\geq^\lambda_\epsilon}(\mathcal{U} \cup \mathcal{V})$ and $\mathcal{U} +^* \mathcal{V} = \max_{\geq^\lambda_\epsilon}(\mathcal{U} + \mathcal{V})$, where $\max_{\geq^\lambda_\epsilon}(\mathcal{U})$ is an (ϵ, λ)-covering of the finite set $\mathcal{U} \subseteq \mathbb{R}^p_+$ (computed using Algorithm 2).

The algorithm called ELIM-MOID$_\epsilon$, which computes an ϵ-covering of the expected utility set, is obtained from Algorithm 1 by replacing $\max$ and $\sum'$ by $\max^*$ and $\sum^*$, respectively. Since the top-down phase of the algorithm consists of t elimination steps, one for each variable, and therefore requires computing t (ϵ, λ_i)-coverings via $\max_{\geq^\lambda_\epsilon}$, $i = 1, \ldots, t$, a sufficient condition to obtain a valid ϵ-covering is to choose the λ_i values summing to 1, specifically $\lambda_i = 1/t$, where t is the number of variables. Thus:

THEOREM **2** *Given a MOID instance $\langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U} \rangle$ with t variables, $p > 1$ objectives and any finite $\epsilon > 0$, algorithm* ELIM-MOID$_\epsilon$ *computes an ϵ-covering.*

The time and space complexity of ELIM-MOID$_\epsilon$ is also bounded exponentially by the induced width of the legal elimination ordering. However, the size of ϵ-covering generated can be in many cases significantly smaller than the corresponding Pareto set, as we will see in Section 6.

5 HANDLING IMPRECISE TRADEOFFS

The Pareto ordering is rather weak (often leading to very large Pareto optimal sets, as mentioned above in Section 4, and illustrated by the experiments in Section 6). Very often the decision maker will be happy to allow some trade-offs between objectives. For example, in a two-objective situation, they may tell us that are happy to gain 3 units of the first objective at the cost of losing one unit of the second, and hence prefer $(3, -1)$ to $(0, 0)$. Such tradeoffs may be elicited using some structured method, or in a more *ad hoc* way. For instance, in Example 1, the decision maker may be asked to imagine a scenario where it was known that the oil content was "wet", and whether they would then have a preference for drilling. To do so would imply a preference of $(50, 12)$ over $(0, 0)$.

We thus consider some set Θ of vector pairs of the form $(\vec{u}, \vec{v})$, where $\vec{u}, \vec{v} \in \mathbb{R}^p$. The idea is that this consists of elicited preferences of the decision maker. We say that binary relation $\succcurlyeq$ on $\mathbb{R}^p$ *extends* Θ if $\vec{u} \succcurlyeq \vec{v}$ for all $(\vec{u}, \vec{v}) \in \Theta$. Similarly, we say that $\succcurlyeq$ extends Pareto if $\vec{u} \geq \vec{v} \Rightarrow \vec{u} \succcurlyeq \vec{v}$.

We consider that the decision maker has a partial order $\succcurlyeq$ over $\mathbb{R}^p$, and that they specify a set of preferences Θ. We assume the Scale-Invariance and Independence properties hold (see Section 2), and, naturally, assume that $\succcurlyeq$ extends

Pareto. The input preferences Θ could contradict the other assumptions we make. We say that Θ *is consistent* if there exists some partial order $\succcurlyeq$ that extends Θ, extends Pareto, and satisfies Scale-Invariance and Independence.

The input preferences Θ (if consistent) give rise to a relation $\succeq_\Theta$ which specifies the deduced preferences. We say that $(\vec{u}, \vec{v})$ *can be deduced from* Θ if $\vec{u} \succcurlyeq \vec{v}$ holds for all partial orders $\succcurlyeq$ that extend Θ, extend Pareto, and satisfy Scale-Invariance and Independence. In this case we write $\vec{u} \succeq_\Theta \vec{v}$. The definition easily implies the following.

PROPOSITION **5** *If Θ is consistent then $\succeq_\Theta$ is a partial order extending Θ and Pareto, and satisfying Scale-Invariance and Independence.*

Proposition 5 shows that this dominance relation $\succeq_\Theta$ satisfies Scale-Invariance and Independence, giving the properties (Theorem 1) we need for the variable elimination algorithm to be correct (up to equivalence).

In Example 1, suppose now we have the additional user preference of $(50, 12)$ over $(0, 0)$, and hence include the pair $((50, 12), (0, 0))$ in Θ. This would then imply that $(11, 12.78)$ is dominated w.r.t. $\succeq_\Theta$ by $(20, 14.2)$.

Theorem 3 below gives a characterization of the partial order $\succeq_\Theta$, which we use as the basis of our implemented algorithm for testing this kind of dominance. Let W be some subset of $\mathbb{R}^p$. Define $\mathbf{C}(W)$, *the convex cone generated by* W, to be the set consisting of all vectors $\vec{u}$ such that there exists $k \geq 0$ and non-negative real scalars $q_1, \ldots, q_k$ and $\vec{w_i} \in W$ with $\vec{u} \geq \sum_{i=1}^{k} q_i \vec{w_i}$, where $\geq$ is the weak Pareto relation (and an empty summation is taken to be equal to 0). $\mathbf{C}(W)$ is the set of vectors that weakly-Pareto dominate some (finite) positive linear combination of elements of W.

THEOREM **3** *Let Θ be a consistent set of pairs of vectors in $\mathbb{R}^p$. Then $\vec{u} \succeq_\Theta \vec{v}$ if and only if $\vec{u} - \vec{v} \in \mathbf{C}(\vec{u_i} - \vec{v_i} : (\vec{u_i}, \vec{v_i}) \in \Theta)$.*

Write finite set of input preferences Θ as $\{(\vec{u_i}, \vec{v_i}) : i = 1, \ldots, k\}$. Theorem 3 shows that, to perform the dominance test $\vec{u} \succeq_\Theta \vec{v}$, it sufficient to check if there exist, for $i = 1, \ldots, k$, non-negative real scalars q_i such that $\vec{u} - \vec{v} \geq \sum_{i=1}^{k} q_i(\vec{u_i} - \vec{v_i})$. This can be determined using a linear programming solver, since it amounts to testing if a finite set of linear inequalities is satisfiable.

Example 4 *Consider $\Theta = \{(-1, 2, -1), (4, -3, 0)\}$ and vectors $\vec{u} = (1, -1, 0)$ and $\vec{v} = (0, -2, 1)$. Then $\vec{u} \succeq_\Theta \vec{v}$ iff $\vec{u} - \vec{v}$ weak Pareto-dominates a non-negative combination of elements of Θ, i.e., $\exists q_1 \geq 0, q_2 \geq 0$ such that $\vec{u} - \vec{v} \geq q_1(-1, 2, -1) + q_2(4, -3, 0)$, which is iff there exists a solution for the linear system defined by: $1 \geq -q_1 + 4q_2$ and $1 \geq 2q_1 - 3q_2$, and $-1 \geq -q_1$. Since this is the case (e.g., $q_1 = 1$; $q_2 = 0.5$) we have $\vec{u} \succeq_\Theta \vec{v}$.*

Alternatively, we can use the fact that the dominance test corresponds to checking whether $\vec{u} - \vec{v}$ is in the convex cone generated by $\{\vec{u_i} - \vec{v_i} : i = 1, \ldots, k\}$ plus the p unit vectors in $\mathbb{R}^p$. We made use of an (incomplete) algorithm [27] for this purpose (which computes the distance of a vector from a cone).

Therefore, the algorithm called ELIM-MOID-TOF that exploits tradeoffs is obtained from Algorithm 1, by replacing the $+'$ and $\max$ operators with $+^\Theta$ and $\max^\Theta$, respectively, where $\max^\Theta(\mathcal{U}, \mathcal{V}) = \max_{\succeq_\Theta}(\mathcal{U} \cup \mathcal{V})$, $\mathcal{U} +^\Theta \mathcal{V} = \max_{\succeq_\Theta}(\mathcal{U} + \mathcal{V})$, and $\max_{\succeq_\Theta}(\mathcal{U})$ is the set of undominated elements of finite set $\mathcal{U} \subseteq \mathbb{R}^p$ with respect to $\succeq_\Theta$.

Instead of eliminating $\succeq_\Theta$-dominated utility values during the computation, one could generate the Pareto optimal set of expected utility values, and only then eliminate $\succeq_\Theta$-dominated values. However, the experimental results in Section 6 (Table 2) indicate that this will typically be much less computationally efficient.

6 EXPERIMENTS

In this section, we evaluate empirically the performance of the proposed variable elimination algorithms on random multi-objective influence diagrams. All experiments were run on a 2.6GHz quad-core processor with 4GB of RAM.

The algorithms considered were implemented in C++ (32-bit) and are denoted by ELIM-MOID (Section 3), ELIM-MOID$_\epsilon$ (Section 4) and ELIM-MOID-TOF (Section 5), respectively. We implemented both methods for performing the $\succeq_\Theta$-dominance, namely the linear programming and the distance from a cone based one, and report only on the former because their performance was comparable overall.

We experimented with a class of random influence diagrams described by the parameters $\langle C, D, k, p, r, a, O \rangle$, where C is the number of chance variables, D is the number of decision variables, k is the maximum domain size, p is the number of parents in the graph for each variable, r is the number of root nodes, a is the arity of the utility functions and O is the number of objectives. The structure of the influence diagram is created by randomly picking $C + D - r$ variables out of $C + D$ and, for each, selecting p parents from their preceding variables, relative to some ordering, whilst ensuring that the decision variables are connected by a directed path. We then added to the graph D utility nodes, each one having a parents picked randomly from the chance and decision variables.

We generated random problems with parameters $k = 2$, $p = 2$, $r = 5$, $a = 3$ and varied $C \in \{15, 25, 35, 45, 55\}$, $D \in \{5, 10\}$ and $O \in \{2, 3, 5\}$, respectively. In each case, 25% of the chance nodes were assigned deterministic CPTs (containing 0 and 1 entries). The remaining CPTs were randomly filled using a uniform distribution. The utility vectors were generated randomly, each objective value being

Table 1: Results with algorithms ELIM-MOID and ELIM-MOID$_\epsilon$ on random influence diagrams. Time limit 20 minutes.

size	w^*	ELIM-MOID					$\epsilon = 0.01$					$\epsilon = 0.1$					$\epsilon = 0.3$				
(C,D,O)		#	time	avg	stdev	med	#	time	avg	stdev	med	#	time	avg	stdev	med	#	time	avg	stdev	med
(15,5,2)	9	16	10.77	3,601	7,422	1,330	20	18.48	2,051	4,811	92	20	0.06	87	150	14	20	0.03	18	24	5
(25,5,2)	11	12	17.28	3,663	6,952	1,623	16	58.47	1,340	2,835	137	20	0.76	157	321	13	20	0.17	24	42	6
(35,5,2)	14	11	60.63	2,104	2,092	2,046	20	141.86	4,554	8,979	344	20	1.49	112	167	23	20	1.12	16	20	5
(45,5,2)	16	5	304.08	7,131	6,086	6,917	18	56.57	1,791	3,183	804	20	24.44	121	199	69	20	2.53	21	27	12
(55,5,2)	18	5	267.74	8,227	11,166	4,266	18	58.91	3,428	6,363	1,270	20	17.59	80	113	23	20	16.60	10	14	5
(15,5,3)	9	13	21.12	3,827	9,993	429	8	8.52	1,247	1,477	973	17	51.31	7,220	12,355	140	19	1.89	470	675	16
(25,5,3)	11	2	163.84	4,638	4,518	4,578	8	103.75	5,167	8,122	2,063	18	77.98	2,641	4,409	876	20	1.43	139	229	62
(35,5,3)	14	0					1	49.56	2,200	0	2,200	13	83.25	6,113	8,614	390	20	37.26	1,326	2,713	200
(45,5,3)	16	1	0.74	907	0	907	3	317.34	30,025	22,643	35,248	15	57.58	5,689	15,495	55	20	76.48	1,256	3,746	20
(55,5,3)	18	0					3	19.95	711	794	220	14	165.69	898	1,918	106	20	85.02	1,434	4,715	43
(15,5,5)	9	7	183.32	19,973	21,437	9,659	3	111.53	7,267	3,789	6,628	7	80.26	2,368	4,541	586	13	10.97	156	235	59
(25,5,5)	11	1	199.13	25,021	0	25,021	0					6	222.08	5,142	5,993	6,499	9	106.95	6,432	17,449	133
(35,5,5)	14	0					0					1	36.04	636	0	636	6	305.37	21	9	27
(45,5,5)	16	0					0					0					7	51.94	6,620	5,898	8,250
(55,5,5)	18	0					0					0					4	77.25	1,556	2,488	3,091
(15,10,2)	12	11	221.24	5,516	6,653	1,946	17	192.55	11,783	27,315	173	20	10.74	1,074	3,122	23	20	6.01	153	470	9
(25,10,2)	17	1	0.62	688	0	688	14	208.05	4,391	12,479	235	19	49.35	186	544	16	20	14.18	42	125	6
(35,10,2)	20	0					2	490.39	3,788	1,601	2,695	15	95.18	266	475	73	16	79.63	68	152	18
(45,10,2)	22	0					1	512.51	4,136	0	4,136	9	196.46	493	647	189	9	125.28	87	111	54
(55,10,2)	26	0					0					1	590.92	20	0	20	1	148.03	7	0	7
(15,10,3)	12	0					0					2	53.32	312	17	164	12	118.71	4,429	9,303	47
(25,10,3)	17	0					0					2	104.96	2,822	2,584	2,703	8	215.39	1,960	2,005	2,678
(35,10,3)	20	0					0					1	15.38	49	0	49	4	272.04	3,496	5,659	6,918
(45,10,3)	22	0					0					2	280.15	956	809	882	2	98.59	87	70	78
(55,10,3)	26	0					0					0					0				
(15,10,5)	12	0					0					0					1	112.00	429	0	429
(25,10,5)	17	0					0					0					1	790.34	584	0	584
(35,10,5)	20	0					0					0					0				
(45,10,5)	22	0					0					0					0				
(55,10,5)	26	0					0					0					0				

drawn uniformly at random between 1 and 30.

We report the average CPU time (in seconds) as well as the average size (together with standard deviation and median) of the maximal expected utility sets generated. In addition, we also record the average induced width (w^*) of the problems obtained using a minfill elimination ordering [24].

Impact of the ϵ-covering Table 1 summarizes the results obtained with algorithms ELIM-MOID and ELIM-MOID$_\epsilon$ with $\epsilon \in \{0.01, 0.1, 0.3\}$ on problems with 5 and 10 decisions. The number shown in column (#) indicates how many instances out of 20 were solved within the time or memory limit. We see that ELIM-MOID can solve only relatively small instances and runs out of time/memory on the larger ones. For example, on problem size $\langle 25, 5, 2\rangle$, ELIM-MOID solved 60% of the instances in about 17 seconds and generated Pareto sets containing about 3,600 vectors on average (with a standard deviation of about 6,900). On the other hand, algorithm ELIM-MOID$_\epsilon$ scales up and solves larger problems while generating significantly smaller ϵ-coverings, especially as ϵ increases. For example, on problem class $(25, 5, 2)$, the average size of the ϵ-covering for $\epsilon = 0.3$ is about 2 orders of magnitude smaller than the corresponding Pareto optimal set. The reason is that as ϵ increases, the corresponding logarithmic grid gets coarser (i.e., fewer cells) and therefore the number of representative vectors needed to cover the optimal Pareto set is smaller.

Figure 3 plots the distribution (mean and standard deviation) of the size of the ϵ-coverings generated for problem class $\langle 35, 5, 5\rangle$, as a function of ϵ. We can see that as ϵ increases the size of the ϵ-covering decreases considerably.

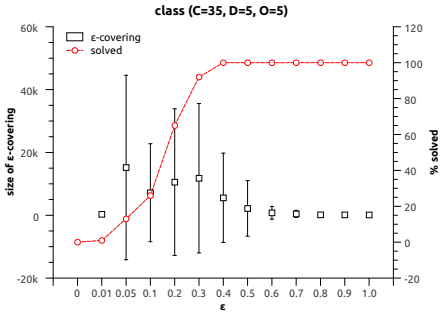


Figure 3: Distribution (mean and stdev) of the ϵ-covering size as a function of the ϵ value. We also plot the number of instances solved out of 100. Time limit 20 minutes.

Impact of imprecise tradeoffs For the purpose of this evaluation, we generated consistent random tradeoffs between the objectives of a given problem instance, as follows. Let (i, j) be a pair of objectives picked randomly out of p objectives. We generate two tradeoffs $a\vec{e}_i - b\vec{e}_j$ and $b\vec{e}_j - ac\vec{e}_i$, where $\vec{e}_i$ and $\vec{e}_j$ are the i-th and j-th unit vectors. Intuitively, one of the tradeoffs indicates how much of objective i one is willing to sacrifice to gain a unit of objective j, and the other is vice versa. In addition, we also generate a 3-way tradeoff between three objectives (i, j, k) picked randomly as well in the form of the tradeoff vector $a\vec{e}_i + b\vec{e}_j - c\vec{e}_k$. Therefore, our random tradeoffs generator is characterized by parameters (K, T, a, b, c), where K is the number of pairs of objectives, T is the number of triplets (and thus a total of $2K + T$ tradeoffs in Θ), and randomly chosen $a, b, c \in [0, 1)$ are used to construct the tradeoff vectors. Notice that parameter c can be used to control the strength of the two-way tradeoffs. Specifically,

Table 2: Results comparing algorithms ELIM-MOID and ELIM-MOID-TOF on random influence diagrams with random tradeoffs. Time limit 20 minutes.

size (C,D,O)	w^*	ELIM-MOID #	time	avg	stdev	med	ELIM-MOID-TOF #	time	avg	stdev	med
$K=1; T=0; a,b,c \in [0.1,1)$											
(15,5,2)	9	9	139.58	2,714	2,864	1,673	94	24.93	98	232	1
(25,5,2)	11	7	18.25	2,344	2,614	269	92	7.12	33	126	1
(35,5,2)	14	2	283.29	9,115	8,934	9,024	83	59.18	147	378	1
(45,5,2)	16	2	397.72	7,596	7,536	7,566	76	23.08	86	302	1
(55,5,2)	18	4	717.68	10,422	5,851	14,896	76	48.02	90	233	2
$K=2; T=1; a,b,c \in [0.1,1)$											
(15,5,3)	9	6	10.69	4,889	4,069	5,830	85	34.62	48	135	2
(25,5,3)	11	2	4.42	4,000	3,938	3,969	70	18.20	41	95	2
(35,5,3)	14	0					50	89.51	119	198	13
(45,5,3)	16	2	242.68	15,431	1,729	8,580	52	28.18	41	73	4
(55,5,3)	18	0					51	94.63	75	154	4
$K=6; T=3; a,b,c \in [0.1,1)$											
(15,5,5)	9	3	65.73	15,062	12,229	14,741	84	19.70	41	104	4
(25,5,5)	11	1	50.35	21,074	0	21,074	74	83.30	97	217	4
(35,5,5)	14	0					59	63.69	101	225	8
(45,5,5)	16	0					61	96.59	107	216	8
(55,5,5)	18	0					41	84.32	51	101	12
$K=1; T=0; a,b,c \in [0.1,1)$											
(15,10,2)	12	5	91.82	6,856	8,280	3,175	78	34.09	60	145	1
(25,10,2)	17	1	808.94	4,964	0	4,964	37	94.08	63	232	1
(35,10,2)	20	0					23	227.78	29	68	1
(45,10,2)	22	0					11	59.97	30	39	13
(55,10,2)	26	0					0				
$K=2; T=1; a,b,c \in [0.1,1)$											
(15,10,3)	12	0					45	157.48	86	191	12
(25,10,3)	17	0					11	204.78	74	83	73
(35,10,3)	20	0					3	303.42	8	8	4
(45,10,3)	22	0					3	158.17	4	4	1
(55,10,3)	26	0					0				
$K=6; T=3; a,b,c \in [0.1,1)$											
(15,10,5)	12	0					40	106.50	27	64	5
(25,10,5)	17	0					21	104.70	67	114	10
(35,10,5)	20	0					5	244.45	72	80	48
(45,10,5)	22	0					0				
(55,10,5)	26	0					0				

if we were to set $c = 1$ then the two objectives i and j would essentially collapse into a single one (we'd have precise rates of exchange between objectives i and j). If $c = 0$ then the second tradeoff for the pair (i, j) is irrelevant.

Table 3: Impact of the quality of the random tradeoffs on bi-objective influence diagrams. Time limit 20 minutes.

size (C,D,O)	w^*	ELIM-MOID #	time	avg	stdev	med	ELIM-MOID-TOF #	time	avg	stdev	med
		$K=0;$					$K=1; c=0$				
(15,5,2)	9	9	139.58	2,714	2,864	1,673	84	48.30	243	542	23
(25,5,2)	11	7	18.25	2,344	2,614	269	61	16.83	79	260	10
(35,5,2)	14	2	283.29	9,115	8,934	9,024	42	94.89	190	344	11
(45,5,2)	16	2	397.72	7,596	7,536	7,566	42	76.83	130	297	5
(55,5,2)	18	4	717.68	10,422	5,851	14,896	41	144.64	268	312	78

Table 2 reports the results obtained with algorithms ELIM-MOID and ELIM-MOID-TOF, respectively. For each problem class $\langle C, D, O\rangle$ we generated 10 random instances, and for each problem instance we generated 10 sets of random tradeoff vectors using the parameters K, T, a, b, c indicated in the header of each horizontal block. As before, the columns labeled by # show how many problems out of 10 (respectively, out of 100) were solved by ELIM-MOID (respectively, ELIM-MOID-TOF). Overall, we notice that the expected utility sets computed by ELIM-MOID-TOF are orders of magnitude smaller than the corresponding Pareto optimal ones generated by ELIM-MOID. We also see that the median size is even smaller, in some cases being actually 1 indicating that the tradeoffs generated were strong enough to make $\succeq_\Theta$ fairly close to being a total order.

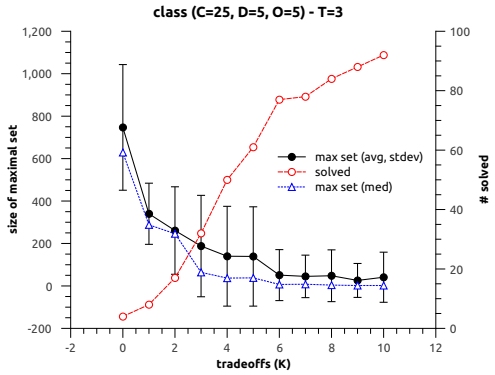


Figure 4: Distribution (mean, stdev, median) of the size of maximal utility sets as a function of pairwise tradeoffs K and fixed 3-way tradeoffs ($T = 3$). We also plot the number of instances solved out of 100. Time limit 20 minutes.

In Table 3 we take a closer look at the impact of the tradeoffs strength for bi-objective problems with 5 decisions. We see that even exploiting a single tradeoff (i.e., using $c = 0$ for two objective case) has a dramatic impact on the size of the maximal expected utility set. For example, on problem class $\langle 55, 5, 2\rangle$, the undominated set of expected utility values computed by ELIM-MOID-TOF contains on average 38 times fewer utility values than the corresponding Pareto optimal set generated by ELIM-MOID.

Figure 4 shows the distribution (mean, standard deviation and median) of the size of the maximal sets generated by ELIM-MOID-TOF on problems from class $\langle 25, 5, 5\rangle$ as a function of the number of pairwise tradeoffs K. As more tradeoffs become available the number of problem instances solved increases because the $\succeq_\Theta$-dominance gets stronger and therefore it reduces the undominated utility sets significantly.

7 CONCLUSION

In this paper, we describe how a variable elimination solution method for influence diagrams is extended to the case of multi-objective utility. A general problem with using the Pareto ordering for multi-objective utility is that the set of maximal expected utility values will often become extremely large. We show how the use of ϵ-coverings can lead to a much more practical computational approach than the exact computation.

We also define a natural way of taking imprecise tradeoffs into account, and give a computational method for checking the resulting dominance condition. Our experimental results indicate that the resulting maximal (multi-objective) values of expected utility can be very much reduced by the adding of (even a small number of) tradeoffs.

References

[1] R. Howard and J. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analyis*, pages 721–762, 1984.

[2] R. Keeney and H. Raiffa. *Decisions with multiple objectives: preferences and value tradeoffs*. Cambridge University Press, 1993.

[3] B. Roy. *Multicriteria methodology for decision analysis*. Kluwer Academic Press, 1996.

[4] M. Ehrgott. *Multicriteria optimization*. Springer, 1999.

[5] N. Wilson and R. Marinescu. An axiomatic framework for influence diagram computation with partially ordered utilities. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 210–220, 2012.

[6] M. Diehl and Y. Haimes. Influence diagrams with multiple objectives and tradeoff analysis. *IEEE Transactions On Systems, Man, and Cybernetics Part A*, 34(3):293–304, 2004.

[7] R. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.

[8] C. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access to web sources. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 86–92, 2000.

[9] J-P. Dubus, C. Gonzales, and P. Perny. Multiobjective optimization using GAI models. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1902–1907, 2009.

[10] L. Zhou, W. Liu, and L. Wang. Influence diagram model with interval-valued utilities. In *IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pages 601–605, 2009.

[11] G. Cooper. A method for using belief networks as influence diagrams. In *Uncertainty in Artificial Intelligence (UAI)*, pages 9–16, 1988.

[12] D. Kikuti and F. Cozman. Influence diagrams with partially ordered preferences. In *Proceedings of 3rd Multidisciplinary Workshop on Advances in Preference Handling*, 2007.

[13] D. Kikuti, F. Cozman, and R. Filho. Sequential decision making with partially ordered preferences. *Artificial Intelligence*, 175(7-8):1346–1365, 2011.

[14] C. de Campos and Q. Ji. Strategy selection in influence diagrams using imprecise probabilities. In *Uncertainty in Artificial Intelligence (UAI)*, pages 121–128, 2008.

[15] M. López-Díaz and L. Rodríguez-Muñiz. Influence diagrams with super value nodes involving imprecise information. *European Journal of Operational Research*, 179(1):203–219, 2007.

[16] D. Maua and C. de Campos. Solving decision problems with limited information. In *Advances in Neural Information Processing Systems (NIPS)*, pages 603–611, 2011.

[17] P. Yu. Cone convexity, cone extreme points, and nondominated solutions in decision problems with multiobjectives. *Journal of Optimization Theory and Applications*, 14(3):319–377, 1974.

[18] M. Wiecek. Advances in cone-based preference modeling for decision making with multiple criteria. *Decision Making in Manufacturing and Services*, 1(1-2):153–173, 2007.

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[20] F. Jensen, V. Jensen, and S. Dittmer. From influence diagrams to junction trees. In *Uncertainty in Artificial Intelligence (UAI)*, pages 367–363, 1994.

[21] H. Raiffa. *Decision analysis*. Addison-Wesley, 1968.

[22] J. Tatman and R. Shachter. Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(1):365–379, 1990.

[23] P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40(1):463–484, 1992.

[24] R. Dechter. A new perspective on algorithms for optimizing policies under uncertainty. In *Artificial Intelligence Planning Systems (AIPS)*, pages 72–81, 2000.

[25] C. Pralet, T. Schiex, and G. Verfaillie. *Sequential decision making problems - representation and solution*. Wiley, 2009.

[26] C. Gonzales, P. Perny, and J-P. Dubus. Decision making with multiple objectives using GAI networks. *Artificial Intelligence*, 175(1):1153–1179, 2011.

[27] Y. Zheng and C. Chew. Distance between a point and a convex cone in n-dimensional space: computation and applications. *IEEE Transactions on Robotics*, 25(6):1397–1412, 2009.

Unsupervised Joint Alignment and Clustering using Bayesian Nonparametrics

Marwan A. Mattar **Allen R. Hanson** **Erik G. Learned-Miller**
Department of Computer Science
University of Massachusetts Amherst, USA
http://vis-www.cs.umass.edu/

Abstract

Joint alignment of a collection of functions is the process of independently transforming the functions so that they appear more similar to each other. Typically, such unsupervised alignment algorithms fail when presented with complex data sets arising from multiple modalities or make restrictive assumptions about the form of the functions or transformations, limiting their generality. We present a transformed Bayesian infinite mixture model that can simultaneously align and cluster a data set. Our model and associated learning scheme offer two key advantages: the optimal number of clusters is determined in a data-driven fashion through the use of a Dirichlet process prior, and it can accommodate any transformation function parameterized by a continuous parameter vector. As a result, it is applicable to a wide range of data types, and transformation functions. We present positive results on synthetic two-dimensional data, on a set of one-dimensional curves, and on various image data sets, showing large improvements over previous work. We discuss several variations of the model and conclude with directions for future work.

1 Introduction

Joint alignment is the process in which data points are transformed to appear more similar to each other, based on a criterion of joint similarity. The purpose of alignment is typically to remove unwanted variability in a data set, by allowing the transformations that reduce that variability. This process is widely applicable in a variety of domains. For example, removing temporal variability in event related potentials allows psychologists to better localize brain responses [32], removing bias in magnetic resonance images [21] provides doctors with cleaner images for their analyses, and removing (affine) spatial variability in images of objects can improve the performance of joint compression [9] and recognition [15] algorithms. Specifically, it has been found that using an aligned version of the Labeled Faces in the Wild [16] data set significantly increases recognition performance [5], even for algorithms that explicitly handle misalignments. Aside from bringing data into correspondence, the process of alignment can be used for other scenarios. For example, if the data are similar up to known transformations, joint alignment can remove this variability and, in the process, recover the underlying latent data [23]. Also, the resulting transformations from alignment have been used to build classifiers using a single training example [21] and learn sprites in videos [17].

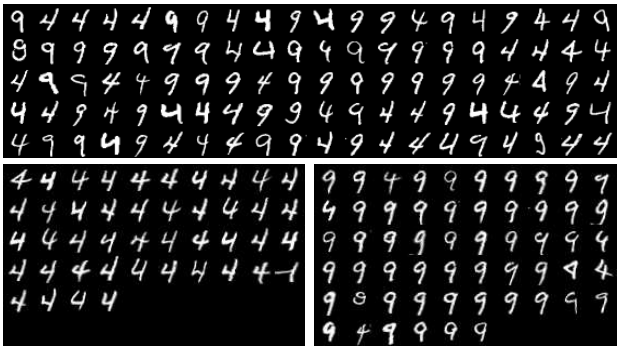


Figure 1: Joint alignment and clustering: given 100 unlabeled images (top), without any other information, our algorithm (§ 3) chooses to represent the data with two clusters, *aligns* the images and *clusters* them as shown (bottom). Our clustering accuracy is 94%, compared to 54% with K-means using two clusters (using the minimum error across 200 random restarts). Our model is not limited to affine transformations or images.

1.1 Previous Work

Typically what distinguishes joint alignment algorithms are the assumptions they make about the data to be aligned

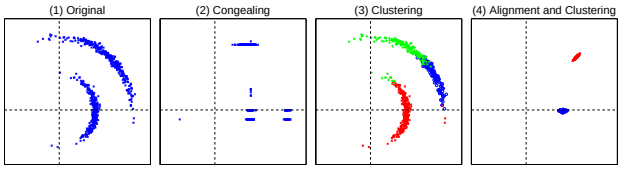


Figure 2: Illustrative example. (1) shows a data set of 2D points. The set of allowable transformations is rotations around the origin. (2) shows the result of the congealing algorithm which transforms points to minimize the sum of the marginal entropies. This independence assumption in the entropy computation causes the points to be squeezed into axis aligned groups. (3) highlights that clustering alone with an infinite mixture model may result in a larger number of clusters. (4) shows the result of the model presented in this paper. It discovers two clusters and aligns the points in each cluster correctly. This result is very close to the ideal one, which would have created tighter clusters.

and the transformations they can incur along with the level of supervision needed. Supervision takes several forms and can range from manually selecting landmarks to be aligned [5] to providing examples of data transformations [28]. In this paper we focus on unsupervised joint alignment which is helpful in scenarios where supervision is not practical or available. Several such algorithms exist.

In the curve domain, the *continuous profile model* [23] uses a variant of the hidden Markov model to locally transform each observation, while a mixture of regression model appended with global scale and translation transformations can simultaneously align and cluster [12]. Mattar *et al*. [25] adapted the *congealing* framework [21] to one dimensional curves. Congealing is an alignment framework that makes few assumptions about the data and allows the use of continuous transformations. It is a gradient-descent optimization procedure that searches for the transformations parameters that maximize the probability of the data under a kernel density estimate. Maximizing the likelihood is achieved by minimizing the entropy of the transformed data. It was initially applied to binary images of digits, but has since also been extended to grayscale images of complex objects [15] and 3D brain volumes [33]. Additionally, several congealing variants [31, 30, 6] have been presented that can improve its performance on binary images of digits and simple grayscale images of faces. Also in the image domain, the transformed mixture of Gaussians [11] and the work of Lui *et al*. [24] are used to align and cluster.

One of the attractive properties of congealing is a clear separation between the transformation operator and optimization procedure. This has allowed congealing to be applied to a wide range of data types and transformation functions [1, 15, 21, 22, 25, 33]. Its main drawback is its inability to handle complex data sets that may contain multiple modes (i.e. images of the digits 1 and 7). While congealing's use of an entropy-based objective function can in theory allow it to align multiple modes, in practice the independence assumption (temporally for curves and spatially for images) can cause it to collapse modes (see Figure 2 for an illustration). Additionally, its method for regularizing parameters to avoid excessive transformations is *ad hoc* and does not prevent it from annihilating the data (shrinking to size zero) in some scenarios.

1.2 Our Approach

The problem we address here is joint alignment of a data set that may contain multiple groups or clusters. Previous nonparametric alignment algorithms (e.g. congealing [21]) typically fail to acknowledge the multi-modality of the data set resulting in poor performance on complex data sets. We address this by simultaneously aligning and clustering [11, 12, 24] the data set. As we will show (and illustrated in Figure 2), solving both alignment and clustering together offers many advantages over clustering the data set first and then aligning the points in each cluster.

To this end, we developed a nonparametric[1] Bayesian joint alignment and clustering model that is a generalization of the standard Bayesian infinite mixture model. Our model possesses many of the favorable characteristics of congealing, while overcoming its drawbacks. More specifically, it:

- Explicitly clusters the data which provides a mechanism for handling complex data sets. Furthermore, the use of a Dirichlet process prior enables learning the number of clusters in a data-driven fashion.
- Can use any generic transformation function parameterized by a vector. This decouples our model from the specific transformations which allows us to plug in different functions for different data types.
- Enables the encoding of prior beliefs regarding the degree of variability in the data set, as well as regularizes the transformation parameters in a principled way by treating them as random variables.

We first present a Bayesian joint alignment model (§ 2) that assumes a unimodal data set (i.e. only one cluster). This model is a special case of our proposed joint alignment and clustering model that we introduce in § 3. We then discuss several variations of our model in § 4 and conclude in § 5 with directions for future work.

1.3 Problem Definition

We are provided with a data set $\mathbf{x} = \{x_i\}_{i=1}^{N}$ of N items and a transformation function, $x_i = \tau(y_i, \rho_i)$ parameter-

[1]Here, we use the term nonparametric to imply that the number of model parameters can grow (a property of the infinite mixtures), and not that the distributions are not parametric.

ized by ρ_i. Our objective is to recover the set of transformation parameters $\{\rho_i\}_{i=1}^N$, such that the aligned data set $\{y_i = \tau(x_i, \rho_i^{-1})\}_{i=1}^N$ is more coherent. In the process, we also learn a clustering assignment $\{z_i\}_{i=1}^N$ of the data points. Here ρ_i^{-1} is defined as the parameter vector generating the inverse of the transformation that would be generated by the parameter vector ρ (i.e. $x_i = \tau(\tau(x_i, \rho_i^{-1}), \rho_i)$).

2 Bayesian Joint Alignment

The Bayesian alignment (BA) model assumes a unimodal data set (in § 3 this assumption is relaxed). Consequently there is a single set of parameters (θ and ρ) that generate the entire data set (see Figure 3). Under this model, every observed data item, x_i, is generated by transforming a canonical data item, y_i, with transformation, ρ_i. More formally, $x_i = \tau(y_i, \rho_i)$, where $y_i \sim F_D(\theta)$ and $\rho_i \sim F_T(\varphi)$. The auxiliary variable y_i is not shown in the graphical model for simplicity. Given the Bayesian setting, the parameters θ and φ are random variables, with their respective prior distributions, $H_D(\lambda)$ and $H_T(\alpha)$.[2]

The model does not assume that there exists a single perfect canonical example that explains all the data, but uses a parametric distribution $F_D(\theta)$ to generate a slightly different canonical example, y_i, for each data item, x_i. This enables it to explain variability in the data set that may not be captured with the transformation function alone. The model treats the transformation function as a black-box operation, making it applicable to a wide range of data types (e.g. curves, images, and 3D MRI scans), as long as an appropriate transformation function is specified.

For both this model and the full joint alignment and clustering model introduced in the next section we use exponential family distributions for $F_D(\theta)$ and $F_T(\varphi)$ and their respective conjugate priors for $H_T(\alpha)$ and $H_D(\lambda)$. This allows us to use Rao-Blackwellized sampling schemes [4] by analytically integrating out the model parameters and caching sufficient statistics for efficient likelihood computations. Furthermore, the hyperparameters now play intuitive roles where they act as a pseudo data set and are easier to set or learn from data.

2.1 Learning

Given a data set $\{x_i\}_{i=1}^N$ we wish to learn the parameters of this model ($\{\rho_i\}_{i=1}^N$, θ, φ). We use a Rao-Blackwellized Gibbs sampler that integrates out the model parameters, θ and φ, and only samples the hidden variables, $\{\rho_i\}_{i=1}^N$. Such samplers typically speed-up convergence. The intuition is that the model parameters are implicitly updated with the sampling of every transformation parameter instead of once per Gibbs iteration. The resulting Gibbs sampler only iterates over the transformation parameters:

$$\begin{aligned}
\forall_{i=1:N}\ \rho_i^{(t)} &\sim p(\rho_i|\mathbf{x}, \boldsymbol{\rho}_{-i}^{(t)}, \alpha, \lambda) \\
&\propto p(\rho_i, x_i|\mathbf{x}_{-i}, \boldsymbol{\rho}_{-i}^{(t)}, \alpha, \lambda) \\
&= p(x_i|\rho_i, \mathbf{x}_{-i}, \boldsymbol{\rho}_{-i}^{(t)}, \lambda) p(\rho_i|\boldsymbol{\rho}_{-i}^{(t)}, \alpha) \\
&= p(y_i|\mathbf{y}_{-i}^{(t)}, \lambda) p(\rho_i|\boldsymbol{\rho}_{-i}^{(t)}, \alpha),
\end{aligned}$$

where $y_i = \tau(x_i, \rho_i^{-1})$. The t superscript in the above equations refers to the Gibbs iteration number.

Sampling ρ_i is complicated by the fact that $p(y_i|\mathbf{y}_{-i}^{(t)}, \lambda)$ depends on the transformation function. Previous alignment research [21, 24], has shown that the gradient of an alignment objective function with respect to the transformations provides a strong indicator for how alignment should proceed. One option would be Hamiltonian Monte Carlo sampling [26] which uses the gradient as a drift factor to influence sampling. However, instead of relying on direct sampling techniques, we use approximations based on the posterior mode [13]. Such an approach is more direct since it is expected that the distribution will be tightly concentrated around the mode. Thus, at each iteration the transformation parameter is updated as follows:

$$\rho_i = \arg\max_{\rho_i} p(y_i|\mathbf{y}_{-i}^{(t)}, \lambda) p(\rho_i|\boldsymbol{\rho}_{-i}^{(t)}, \alpha).$$

Interestingly, the same learning scheme can be derived using the incremental variant [27] of hard-EM.

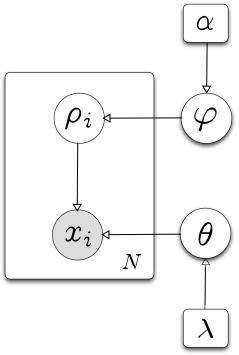


Figure 3: Graphical representation for our proposed Bayesian alignment model (§ 2).

2.2 Model Characteristics

The objective function optimized in our model contains two key terms, a data term, $p(\mathbf{x}|\boldsymbol{\rho}, \theta)$, and a transformation term, $p(\boldsymbol{\rho}|\varphi)$. The latter acts as a regularizer to penalize large transformations and prevent the data from being annihilated. One advantage of our model is that large transformations are penalized in a principled fashion. More specifically, the cost of a transformation, ρ_i is based on the

[2] Here we assume that the hyperparameters α and λ are fixed, but they can be learned or sampled if necessary.

Figure 4: Top row. Means before alignment. Bottom row. Means after alignment with BA. The averages of pixelwise entropies are as follows. Before: 0.3 (top), with congealing: 0.23 (not shown), and with BA: 0.21 (bottom).

learned parameter φ which depends on the transformations of all the other data items, $\boldsymbol{\rho}_{-i}$, and the hyperparameters, α. Learning φ from the data is a more effective means for assigning costs than handpicking them.

The model has several other favorable qualities. It is efficient, can operate on large data sets while maintaining a low memory footprint, allows continuous transformations, regularizes transformations in a principled way, is applicable to a large variety of data types, and its hyperparameters are intuitive to set. Its main drawback is the assumption of a unimodal data set, which we remedy in § 3. We first evaluate this model on digit and curve alignment.

2.3 Experiments

Digits. We selected 50 images of every digit from the MNIST data set and performed alignment on each digit class independently. The mean images before and after alignment are presented in Figure 4. We allowed 7 affine image transformations: scaling, shearing, rotating and translation. $F_D(\theta)$ is the product of independent Bernoulli distributions, one for each pixel location, and $F_T(\varphi)$ is a $7-$D zero mean diagonal Gaussian. For comparison, we also ran the congealing algorithm (see Figure 4).

Curves. We generated 85 curve data sets in a manner similar to curve congealing [25], where we took five original curves from the UCR repository [18] and for each one generated 17 data sets, each containing 50 random variations of the original curve. We used the same transformation function in curve congealing [25], which allows non-linear time warping (4 parameters), non-linear amplitude scaling (8 parameters), linear amplitude scaling (1 parameter), and amplitude translation (1 parameter). $F_D(\theta)$ was set to a diagonal Gaussian distribution (i.e. we treat the raw curves as a random vector), and $F_T(\varphi)$ was a $14-$D zero mean diagonal Gaussian. Again, we compared against the curve congealing algorithm. We computed a standard deviation score by summing the standard deviation at each time step of the final alignment produced by both algorithms. Figure 5 shows a scatter plot of these scores obtained by congealing and BA for all 85 data sets, as well as sample alignment results on two difficult cases. As the figure shows, the curve data sets can be quiet complex.

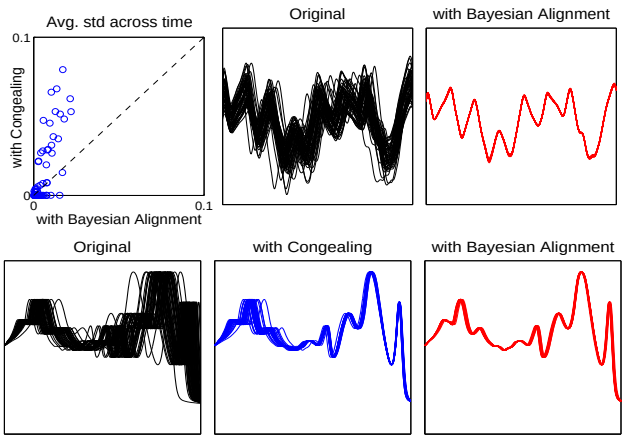


Figure 5: Top row. Left: scatter plot of the standard deviation score (see text) of congealing and the Bayesian alignment algorithm across the 85 synthetic curve data sets. Middle: An example of a difficult data set. Right: The alignment result of the difficult data set. The bottom row shows an example where BA outperformed congealing.

Discussion. On the digits data sets, BA performed at least as well as congealing for every digit class and on average performed better. On the curves data sets, BA does substantially better than congealing in many cases, but in some cases congealing does slightly better. In all the experiments, both congealing and BA converged. BA's advantage is largely due to its explicit regularization of transformations which enables it to perform a maximization at each iteration. Congealing's lack of such regularization requires it to take small steps at each iteration making it more susceptible to local optima. Furthermore, congealing typically requires five times the number of iterations to converge.

3 Clustering with Dirichlet Processes

We now extend the BA model introduced in the previous section to explicitly cluster the data points. This provides a mechanism for handling complex data sets that may contain multiple groups.

The major drawback of the BA model is that a single pair of data and transformation parameters (θ and φ, respectively) generate the entire data set. One natural extension to this generative process is to assume that we have several such parameter pairs (finite but unknown a priori) and each data point samples its parameter pair. By virtue of points sampling the same parameter pair, they are assigned to the same group or cluster. A Dirichlet process (DP) provides precisely this construction and serves as the prior for the data and transformation parameter pairs.

A DP essentially provides a distribution over distributions, or, more formally, a distribution on random probability

measures. It is parameterized by a base measure and a concentration parameter. A draw from a DP generates a finite set of samples from the base measure (the concentration parameter controls the number of samples). A key advantage of DP's is that the number of unique parameters (i.e. clusters) can grow and adapt to each data set depending on its size and characteristics. Under this new probability model, data points are generated in the following way:

1. Sample from the DP, $G \sim DP(\gamma, H_\alpha \times H_\lambda)$. γ is the concentration parameter, and H_α and H_λ are the base measures for $F_T(\varphi)$ and $F_D(\theta)$ respectively.

2. For each data point, x_i, sample a data and transformation parameter pair, $(\theta_i, \varphi_i) \sim G$.

3. Sample a transformation and canonical data item from their distributions, $y_i \sim F_D(\theta_i)$ and $\rho_i \sim F_T(\varphi_i)$.

4. Transform the canonical data item to generate the observed sample, $x_i = \tau(y_i, \rho_i)$.

Figure 6 depicts the generative process as described above (distributional form, right) and in the more traditional graphical representation with the cluster random variable, z, and mixture weights, π, made explicit (left).

Our model can thus be seen as an extension of the standard Bayesian infinite mixture model where we introduced an additional latent variable, ρ_i, for each data point to represent its transformation. Several existing alignment models [11, 12, 21, 23] can be viewed as similar extensions to other standard generative models. Sometimes the transformations are applied to other model parameters instead of data points as in the case of transformed Dirichlet processes (TDP) [29]. TDP is an extension of *hierarchical Dirichlet processes* where global mixture components are transformed before being reused in each group. The challenge in introducing additional latent variables is in designing efficient learning schemes that can accommodate this increase in model complexity.

3.1 Learning

We consider two different learning schemes for this model. The **first** is a blocked, Rao-Blackwellized Gibbs sampler, where we sample both the cluster assignment z_i, and transformation parameters ρ_i, simultaneously:

$$\begin{aligned}(z_i^{(t)}, \rho_i^{(t)}) &\sim p(z_i, \rho_i | \mathbf{z}_{-i}^{(t)}, \boldsymbol{\rho}_{-i}^{(t)}, \mathbf{x}, \gamma, \alpha, \lambda) \\ &\propto p(z_i | \mathbf{z}_{-i}^{(t)}, \gamma) p(\rho_i | \boldsymbol{\rho}_{-i}^{(t)}, \alpha) p(y_i | \mathbf{y}_{-i}^{(t)}, \lambda).\end{aligned}$$

As with the BA model, we approximate $p(\rho_i | \boldsymbol{\rho}_{-i}^{(t)}, \alpha) p(y_i | \mathbf{y}_{-i}^{(t)}, \lambda)$ with a point estimate based on its mode. Consequently this learning scheme is a direct generalization of the one derived for the BA model. Note

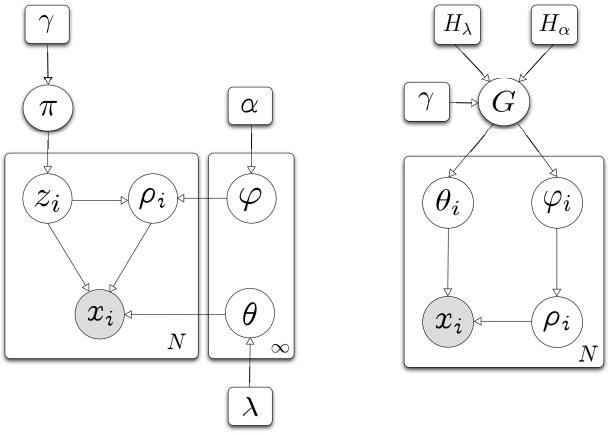


Figure 6: Graphical representation for our proposed nonparametric Bayesian joint alignment and clustering model (left) and its corresponding distributional form (right).

that $p(z_i | \mathbf{z}_{-i}^{(t)}, \gamma)$ is the cluster predictive distribution based on the Chinese restaurant process (CRP) [2].

While this sampler is effective (it produced the positive result in Figure 1) it scales linearly with the number of clusters and computing the most likely transformation for a cluster is an expensive operation. We designed an alternative sampling scheme that does not require the expensive mode computation and whose running time is independent of the number of clusters.

The **second** sampler further integrates out the transformation parameter, and only samples the cluster assignment. We now derive an implementation for this sampler.

$$\begin{aligned}\forall_{i=1:N} \; z_i^{(t)} &\sim p(z_i \,|\, \mathbf{z}_{-i}^{(t)}, \mathbf{x}, \gamma, \alpha, \lambda) \\ &\propto p(z_i, x_i \,|\, \mathbf{z}_{-i}^{(t)}, \mathbf{x}_{-i}, \gamma, \alpha, \lambda) \\ &= p(z_i \,|\, \mathbf{z}_{-i}^{(t)}, \gamma) p(x_i \,|\, \mathbf{z}^{(t)}, \mathbf{x}_{-i}, \alpha, \lambda).\end{aligned}$$

$$\begin{aligned}&p(x_i \,|\, \mathbf{z}, \mathbf{x}_{-i}, \alpha, \lambda) \\ &= \int_{\boldsymbol{\theta}} \int_{\boldsymbol{\varphi}} \int_{\rho_i} p(x_i, \rho_i, \boldsymbol{\theta}, \boldsymbol{\varphi} \,|\, \mathbf{z}, \mathbf{x}_{-i}, \alpha, \lambda) \; d\rho_i \; d\boldsymbol{\varphi} \; d\boldsymbol{\theta} \\ &= \int_{\boldsymbol{\theta}} \int_{\boldsymbol{\varphi}} \left(\int_{\rho_i} p(x_i, \rho_i \,|\, z_i, \boldsymbol{\theta}, \boldsymbol{\varphi}, \alpha, \lambda) \; d\rho_i \right) \cdots \\ &\quad p(\boldsymbol{\theta}, \boldsymbol{\varphi} \,|\, \mathbf{z}_{-i}, \mathbf{x}_{-i}, \alpha, \lambda) \; d\boldsymbol{\varphi} \; d\boldsymbol{\theta} \\ &\overset{(1)}{\approx} \int_{\rho_i} p(x_i, \rho_i \,|\, z_i, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\varphi}}, \alpha, \lambda) \; d\rho_i, \\ &\quad \textit{s.t.} \;\; (\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\varphi}}) = \arg\max_{\boldsymbol{\theta}, \boldsymbol{\varphi}} p(\boldsymbol{\theta}, \boldsymbol{\varphi} \,|\, \mathbf{z}_{-i}, \mathbf{x}_{-i}, \alpha, \lambda) \\ &= \int_{\rho_i} p(\rho_i \,|\, \hat{\boldsymbol{\varphi}}, z_i, \alpha) p(x_i \,|\, \rho_i, z_i, \hat{\boldsymbol{\theta}}, \lambda) \; d\rho_i \\ &= \int_{\rho_i} p(\rho_i \,|\, \hat{\varphi}_{z_i}, \alpha) p(x_i \,|\, \rho_i, \hat{\theta}_{z_i}, \lambda) \; d\rho_i\end{aligned}$$

$$\overset{(2)}{\approx} \quad \frac{\sum_{l=1}^{L} w_i \cdot p(x_i \mid \hat{\rho_i}^l, \hat{\theta}_{z_i}, \lambda)}{\sum_{l=1}^{L} w_i}$$

$$\text{s.t. } \{\hat{\rho_i}^l\}_{l=1}^{L} \sim q(\rho), \qquad w_i = \frac{p(\hat{\rho_i}^l \mid \hat{\varphi}_{z_i}, \alpha)}{q(\hat{\rho_i}^l)}$$

(1) approximates the posterior distribution of the parameters by its mode. The mode is computed using incremental hard-EM. Furthermore, the mode can be computed for each cluster's parameters independently. For every other data point j, perform an EM update:

$$\text{E:} \quad \hat{\rho}_j = \arg\max_{\rho_j} p(\rho_j \mid x_j, \hat{\theta}_{z_j}, \hat{\varphi}_{z_j}) = \arg\max_{\rho_j} p(\rho_j \mid \hat{\varphi}_{z_j}) p(x_j \mid \rho_j, \hat{\theta}_{z_j})$$

$$\text{M:} \quad \hat{\theta}_{z_j} = \arg\max_{\theta} p(\theta, \mid \{x_k, \rho_k \mid z_k = z_j\}, \lambda)$$
$$\hat{\varphi}_{z_j} = \arg\max_{\varphi} p(\varphi \mid \{\rho_k \mid z_k = z_j\}, \alpha)$$

(2) uses importance sampling in order to reduce the number of data transformations that need to be performed. Computing $p(x_i \mid \hat{\rho_i}^l, \hat{\theta}_{z_i}, \lambda)$ requires transforming the data point, which is the most computationally expensive single operation for this sampler. Thus it would be wise to reuse the samples, $\{\hat{\rho_i}^l\}_{l=1}^{L}$, across different clusters. We achieve this through importance sampling, which proceeds by sampling a set of transformation parameters from a proposal distribution, $q(\rho)$ and using those samples for all the clusters by reweighting them differently for each cluster. This is a large computational saving since the number of data transformation operations performed in a single iteration of this sampler is now independent of the number of clusters. Furthermore, the quality of approximation is controlled by the number of samples, L, generated.

To further increase the efficiency of the sampler, we approximate the maximization in the E-step by reusing the samples and selecting the one that maximizes $p(\rho_j \mid x_j, \hat{\theta}_{z_j}, \hat{\varphi}_{z_j})$. This avoids the direct maximization operation in the E-step which can be expensive. While not adopted in this work, further computational gains might be achieved at the expense of memory by storing and reusing samples (i.e. transformed data points) across iterations and reweighting them accordingly.

Thus our sampler iterates over every point in the data set, samples a cluster assignment and then updates $\hat{\theta}$ and $\hat{\varphi}$ for the sampled cluster. It also updates its own transformation parameter, $\hat{\rho}_i$ in the process.

Summary. We presented two samplers for our joint alignment and clustering model. Both samplers work well in practice, but the second is more efficient. For both samplers, every iteration begins by randomly permuting the order of the points and the DP concentration parameter is resampled using auxiliary variable methods [10]. As in the BA model, we cache the sufficient statistics for every cluster which can be updated efficiently as points are reassigned to clusters to allow for efficient likelihood and mode computation.

3.2 Incorporating Labelled Examples

The model presented in the previous section was used without any supervision. Supervision here refers to the ground-truth labels for some of the data points or the correct number of clusters. However, there are many scenarios where this information is available and would be advantageous to incorporate.

It is straightforward to modify the joint alignment and clustering model to accommodate such labelled examples. Lets assume we have positive examples for each cluster as well as a large data set of unlabeled examples. Before attempting to align and cluster the unlabeled examples, we would initialize several clusters and assign the positive examples to their respective clusters. By assigning these examples to their clusters and updating the sufficient statistics accordingly, the cluster parameters have incorporated the positive examples. Depending on the strength of the priors (i.e. the hyperparameters) and the number of positive examples per cluster it may be necessary to add the positive examples several times. The stronger the prior, the more times the positive examples need to be replicated. Note that replicating the positive examples does not increase memory usage since we only store the sufficient statistics for each cluster.

If the labelled portion contains positive examples for all the clusters, then setting the concentration parameter of the DP to 0 would prevent additional, potentially unnecessary, clusters from being created.

3.3 Experiment: Alignment and Clustering of Digits

We evaluated our unsupervised and semi-supervised models on two challenging data sets. The first contains 100 images of the digits "4" and "9", which are the two most similar and confusing digit classes (the performance of KMeans on this data set is close to random guessing). The second contains the 200 images of all 10 digit classes used by Liu *et al*. [24].[3] For the second data set we used the Histogram of Oriented Gradients (HOG) feature representation [7] used by Liu *et al*. to enable a fair comparison.

For both digit data sets we compared several algorithms using the same two metrics reported by Lui *et al*.: *alignment*

[3] Liu *et al*. also evaluated their model on 6 Caltech-256 categories and the CEAS face data set. For both data sets they randomly selected 20 images from each category. We found that the difficulty of a data set varied greatly from one sample to another, so we reached out to the authors. Unfortunately, they were only able to provide us with the digits data set which we do use. The digits data set was the most difficult of the three.

Algorithm	Digits 4 and 9 (Fig 1)		All 10 digits (Fig 7)	
	Alignment	Clustering	Alignment	Clustering
KMeans	$4.18\ (1.57) \pm 0.031$	54.0%	$4.88\ (1.61) \pm 0.033$	62.5%
Infinite mixture model [10]	$3.64\ (1.34) \pm 0.036$	$86.0\%, 4$	$4.87\ (1.64) \pm 0.037$	$69.5\%, 13$
Congealing [21]	$2.11\ (0.93) \pm 0.019$	83.0%	$3.51\ (1.34) \pm 0.029$	70.5%
TIC [11]	–	–	$6.00\ (1.1)$	35.5%
Unsupervised SAC [24]	–	–	$3.80\ (0.9)$	56.5%
Semi-supervised SAC [24]	–	–	not reported	73.7%
Unsupervised JAC [§ 3.1]	$\mathbf{1.44\ (0.69) \pm 0.014}$	$\mathbf{94.0\%}, 2$	$\mathbf{2.38\ (1.12) \pm 0.027}$	$\mathbf{87.0\%}, 12$
Semi-supervised JAC [§ 3.2]	$1.58\ (0.79) \pm 0.016$	$\mathbf{94.0\%}$	$2.71\ (1.25) \pm 0.028$	82.5%

Table 1: Joint alignment and clustering of images. The left subtable refers to the first digit data set comprising 100 images containing the digits "4" and "9" (Figure 1), while the right subtable refers to the second data set comprising 200 images containing all 10 digits (the same data set used by Liu *et al*. [24], Figure 7). The alignment score columns contain three metrics that adhere to the following template: mean (standard deviation) $\pm$ standard error. The number following the clustering accuracy in the "Infinite mixture model" and "Unsupervised JAC" rows is the number of clusters that the model discovered (i.e. chose to represent the data with). On both data sets, our models significantly outperforms previous nonparametric alignment [21], joint alignment and clustering [11, 24], and nonparametric Bayesian clustering [10] models.

score measures the distance between pairs of aligned images assigned to the same cluster (we report the mean and standard deviation of all the distances, and the standard error[4]), and *clustering accuracy* is the Rand index with respect to the correct labels.

Table 1 summarizes the results on the models we evaluated:

- KMeans: we clustered the digits into the correct number of ground-truth classes (2 for the first data set, and 10 for the second) using the best of 200 KMeans runs.
- Infinite mixture model: removing the transformation/alignment component of our model reduces it to a standard Bayesian infinite mixture model. We ran this model to evaluate the advantage of joint alignment and clustering.
- Congealing: we ran congealing on all the images simultaneously and after alignment converged, clustered the aligned images using KMeans (with the correct number of ground-truth clusters). This allows us to evaluate the advantages of simultaneous alignment and clustering over alignment followed by clustering.
- TIC, USAC and SSAC results are listed exactly as reported by Liu *et al*.
- Unsupervised JAC refers to our full nonparametric Bayesian alignment and clustering model (§ 3.1).
- Semi-supervised JAC refers to the semi-supervised variant of our alignment and clustering model (§ 3.2). We used a *single* positive example for each digit and set the DP concentration parameter to 0.

[4]The standard error here is defined as the sample standard deviation divided by the square root of the number of pairs.

Note that the alignment scores for KMeans and the infinite mixture model are not relevant since no alignment takes place in either of these two algorithms. They are only included to offer a reference for the alignment score when the data is not transformed.

As the results show, our models outperform previous work with respect to both alignment and clustering quality. We make three observations about these results:

1. Our unsupervised model outperformed the unsupervised model of Liu *et al*. by 30.5%, and our semi-supervised model outperformed their semi-supervised model by 8.8%. This is in addition to the significant improvement in alignment quality.
2. Our unsupervised model improved upon the standard infinite mixture model in terms of alignment quality, clustering accuracy, and correctness of the discovered number of clusters.
3. The number of clusters discovered by our unsupervised model is quite accurate. For the first data set the model discovered the correct number of clusters (see Figure 1), and for the second it needed two additional clusters (see Figure 7).

These positive results validate our joint alignment and clustering models and associated learning schemes. Furthermore, it provides evidence for the advantage of solving both alignment and clustering problems simultaneously instead of independently.

3.4 Experiment: Alignment and Clustering of Curves

We now present joint alignment and clustering results on a challenging curve data set of ECG heart data [19] that is

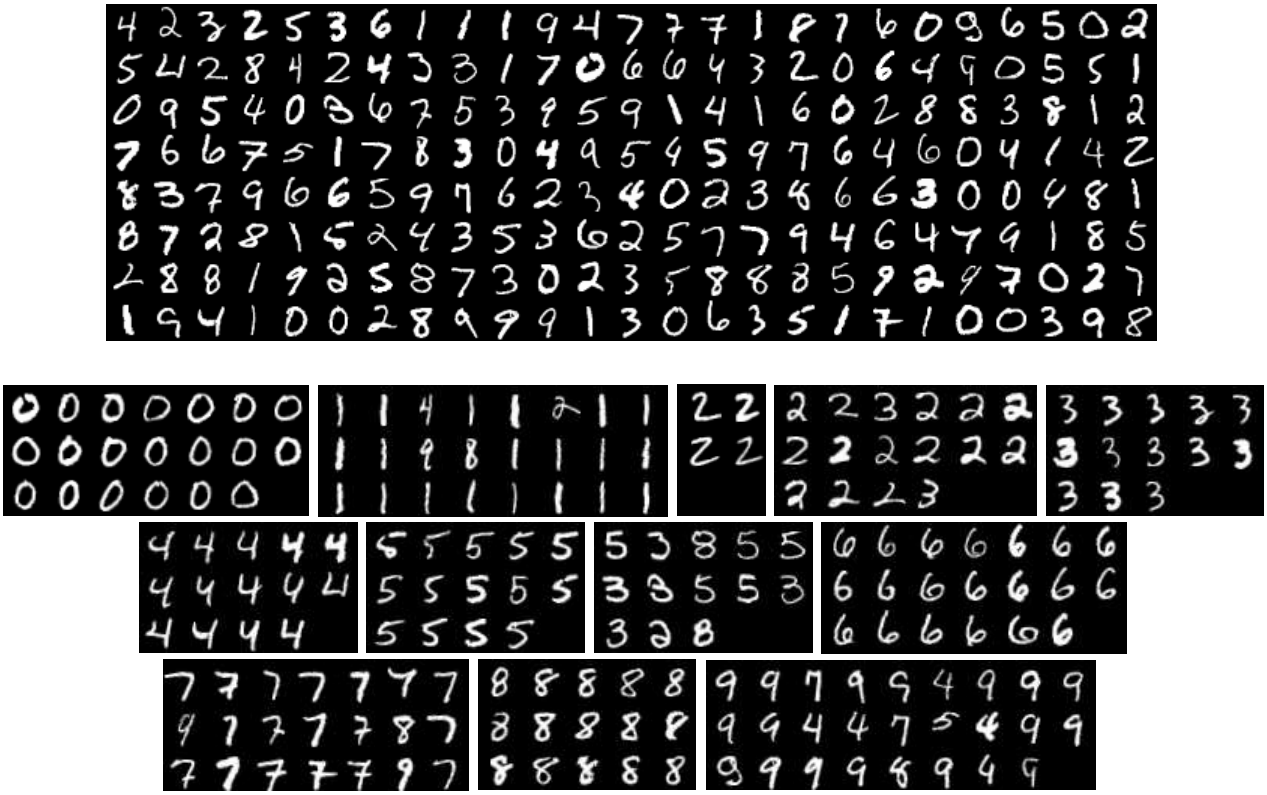


Figure 7: Unsupervised joint alignment and clustering of 200 images of all 10 digits. (Top) All 200 images provided to our model. (Bottom) The 12 clusters discovered and their alignments.

helpful in identifying the heart condition of patients. This data set contains 46 curves. 24 represent a normal heartbeat, and 22 represent an abnormal heartbeat. We ran both congealing and our nonparametric Bayesian joint alignment and clustering model. In both cases we excluded the non-linear scaling in amplitude transformation since the amplitudes of the curves are helpful in classifying whether the curve is normal or abnormal.

Our model discovered 5 clusters in the data set resulting in a clustering accuracy of 84.8%. Inspecting the clusters discovered by our model in Figure 8 highlights the fact that although the data set represents two groups (normal and abnormal), the curves do not naturally fall into two clusters and more are needed to explain the data appropriately. Figure 8 also displays the result of congealing the curves. Clustering the congealed curves into 2 clusters using the best of 200 KMeans runs results in a clustering accuracy of 71.7%. Clustering the congealed curves into 5 clusters in a similar manner results in a clustering accuracy of 76.1%.

The large improvement in clustering accuracy over congealing in addition to a much cleaner alignment result (Figure 8) highlights the importance of explicit clustering when presented with a complex data set. Furthermore it showcases our models ability to perform equally well on both image and curve data sets.

4 Discussions

In this section we discuss the adaptation of our joint alignment and clustering model to both online (when the data arrives at intervals) and distributed (when multiple processors are available) settings. Both of these adaptations are applicable to the unsupervised and semi-supervised settings.

4.1 Online Learning

There are several scenarios where online alignment and clustering may be helpful. Consider for instance a very large data set that cannot fit in memory or the case where the data set is not available up front but arrives over an extended period of time (such as in a tracking application).

An advantage of our model that has not yet been raised is its ability to easily adapt to an online setting where only a portion of the data set is available in the beginning. This is due to our use of conjugate priors and distributions in the exponential family which enable us to efficiently summarize an entire cluster through its sufficient statistics. Consequently, we can align/cluster the initial portion of the data set and save out the sufficient statistics for every cluster after each iteration (for both the data and transformations). Then as new data arrives, we can load in the sufficient statistics and use them to guide the alignment and clustering of the new

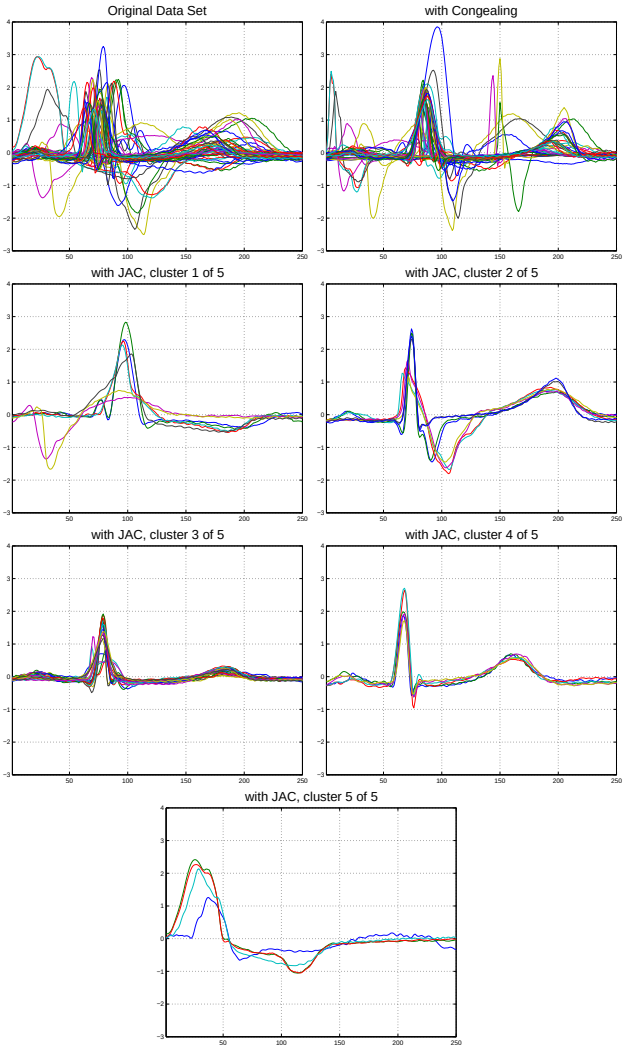


Figure 8: Joint alignment and clustering of ECG heart data. The first row displays the original data set (left) and the result of congealing (right). The last three rows display the 5 clusters discovered by our model.

data in lieu of the original data set which can now be discarded.

Given a sufficiently large initial data set, the alignment of a new data point using the procedure described above would be nearly identical to the result had that data point been included in the original set. This is true since the addition of a single point to an already large data set would have a negligible effect on the sufficient statistics. This process is also applicable to the Bayesian alignment model.

4.2 Distributed Learning

We now describe how to adapt our sampling scheme to a distributed setting using the MapReduce [8] framework. This facilitates scaling our model to large data sets in the presence of many processors. The key difference between the MapReduce implementation and the one described in § 3.1 is that the cluster parameters are updated once per sampling iteration instead of after each point's reassignment (i.e. using a standard sampler instead of a Rao-Blackwellized sampler).

A MapReduce framework involves two key steps, Map and Reduce. For our model the mapper would handle updating the transformation parameter and clustering assignment of a single data point, while the reducer would handle updating the parameters of a single cluster. More specifically, the input to each Map operation would be a data point along with a snapshot of the model parameters (the set of sufficient statistics that summarize the data set). The Map would output the updated cluster assignment and transformation parameter for that data point. The input to the Reduce step would then be all the data points that were assigned to a specific cluster (i.e. we would have a Reduce operation for every cluster created). The Reducer would then update the cluster parameters. Thus each sampling iteration is composed of a Map and Reduce stage.

5 Conclusion

We presented a nonparametric Bayesian joint alignment and clustering model that has been successfully applied to curve and image data sets. The model outperforms congealing and yields impressive gains in clustering accuracy over infinite mixture models. These results highlight the advantage of solving both alignment and clustering tasks simultaneously.

A strength of our model is the separation of the transformation function and sampling scheme, which makes it applicable to a wide range of data types, feature representations, and transformation functions. In this paper we presented results on three data types (2D points, 1D curves, and images), three transformation functions (point rotations, nonlinear curve transformations and affine image transformations), and two feature representations (identity and HOG).

In the future we foresee our model applied to a wide array of problems. Since curves are a natural representation for object boundaries [18], one of our goals is to apply our model to shape matching. We also intend to explore alternative parameter learning schemes based on variational inference [3, 20, 14].

Acknowledgements

Special thanks to Gary Huang for many entertaining and helpful discussions. We would also like to thank Sameer Singh and the anonymous reviewers for helpful feedback. This work was supported in part by the National Science Foundation under CAREER award IIS-0546666 and Grant IIS-0916555.

References

[1] P. Ahammad, C. Harmon, A. Hammonds, S. Sastry, and G. Rubin. Joint nonparametric alignment for analyzing spatial gene expression patterns of Drosophila imaginal discs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[2] D. Blackwell and J. MacQueen. Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, 1:353–355, 1973.

[3] D. M. Blei and M. I. Jordan. Variational inference for Dirichlet process mixtures. *Journal of Bayesian Analysis*, 1(1):121–144, 2006.

[4] G. Cassella and C. P. Robert. Rao-Blackwellization of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[5] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[6] M. Cox, S. Lucey, J. Cohn, and S. Sridharan. Least squares congealing for unsupervised alignment of images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[8] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Symposium on Operating System Design and Implementation*, 2004.

[9] M. Elad, R. Goldenberg, and R. Kimmel. Low bit-rate compression of facial images. *IEEE Transactions on Image Analysis*, 9(16):2379–2383, 2007.

[10] M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, June 1995.

[11] B. J. Frey and N. Jojic. Transformation-invariant clustering and dimensionality reduction using EM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):1–17, Jan. 2003.

[12] S. Gaffney and P. Smyth. Joint probabilistic curve clustering and alignment. In *Neural Information Processing Systems*, 2004.

[13] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. Bayesian Data Analysis. Chapman and Hall, CRC Press, 2003.

[14] R. Gomes, M. Welling, and P. Perona. Incremental learning of nonparametric Bayesian mixture models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[15] G. B. Huang, V. Jain, and E. G. Learned-Miller. Unsupervised joint alignment of complex images. In *IEEE International Conference on Computer Vision*, 2007.

[16] G. B. Huang, M. A. Mattar, T. Berg, and E. G. Learned-Miller. Labeled Faces in the Wild: A database for studying face recognition in unconstrained environments. In *IEEE ECCV Workshop on Faces in Real-Life Images*, 2008.

[17] N. Jojic and B. J. Frey. Learning flexible sprites in video layers. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[18] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. UCR Time Series Repository, Feb. 2008.

[19] S. Kim and P. Smyth. Segmental hidden Markov models with random effects for waveform modeling. *Journal of Machine Learning Research*, 7:945–969, 2006.

[20] K. Kurihara, M. Welling, and Y. W. Teh. Collapsed variational Dirichlet process mixture models. In *International Joint Conference on Artificial Intelligence*, 2007.

[21] E. G. Learned-Miller. Data-driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):236–250, Feb. 2006.

[22] E. G. Learned-Miller and P. Ahammad. Joint MRI bias removal using entropy minimization across images. In *Neural Information Processing Systems*, 2004.

[23] J. Listgarten, R. M. Neal, S. T. Roweis, and A. Emili. Multiple alignment of continuous time series. In *Neural Information Processing Systems*, 2004.

[24] X. Liu, Y. Tong, and F. W. Wheeler. Simultaneous alignment and clustering for an image ensemble. In *IEEE International Conference on Computer Vision*, 2009.

[25] M. A. Mattar, M. G. Ross, and E. G. Learned-Miller. Nonparametric curve alignment. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009.

[26] R. M. Neal. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman and Hall, CRC Press, May 2011.

[27] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368, 1998.

[28] A. Rahimi and B. Recht. Learning to transform time series with a few examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1759–1775, Oct. 2007.

[29] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Describing visual scenes using transformed Dirichlet processes. In *Neural Information Processing Systems*, 2005.

[30] A. Vedaldi, G. Guidi, and S. Soatto. Joint alignment up to (lossy) transformations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[31] A. Vedaldi and S. Soatto. A complexity-distortion approach to joint pattern alignment. In *Neural Information Processing Systems*, 2006.

[32] K. Wang, H. Begleiter, and B. Projesz. Warp-averaging event-related potentials. *Clinical Neurophysiology*, (112):1917–1924, 2001.

[33] L. Zollei, E. G. Learned-Miller, E. Grimson, and W. M. Wells. Efficient population registration of 3D data. In *IEEE ICCV Workshop on Computer Vision for Biomedical Image Applications: Current Techniques and Future Trends*, 2005.

Hokusai — Sketching Streams in Real Time

Sergiy Matusevych
Yahoo! Research
sergiy.matusevych@gmail.com

Alexander J. Smola
Google Inc.
alex@smola.org

Amr Ahmed
Yahoo! Research
amahmed@cs.cmu.edu

Abstract

We describe 北斎 Hokusai, a real time system which is able to capture frequency information for streams of arbitrary sequences of symbols. The algorithm uses the Count-Min sketch as its basis and exploits the fact that sketching is linear. It provides real time statistics of arbitrary events, e.g. streams of queries as a function of time. We use a factorizing approximation to provide point estimates at arbitrary (time, item) combinations. Queries can be answered in constant time.

1 Introduction

Obtaining frequency information of data streams is an important problem in the analysis of sequence data. Much work exists describing how to obtain highly efficient frequency counts of sequences at any given time. For instance, the Count-Min [6] sketch is capable of providing ϵ accuracy with failure probability δ in $O(\log \delta/\epsilon)$ space. Even better guarantees are possible for the space-saver sketch [12], albeit at the expense of having to keep a dictionary of tokens and a considerably more computationally expensive data structure (we need an ordered list rather than just a flat array in memory). The key point is that sketching algorithms can be queried at any time to return the total number of symbols of any given type seen so far.

While highly desirable in its own right it does not solve the following: we would like to know how many symbols were observed at some time in the past. For instance, we may want to know how many queries of "Britney Spears" were carried out, say at noontime last year on Easter Sunday. Clearly we could address this problem by brute force, that is by pre-processing the logfiles. Query languages such as Pig [14] are well suited to generating such summary statistics, albeit unable to retrieve it in *real time*.

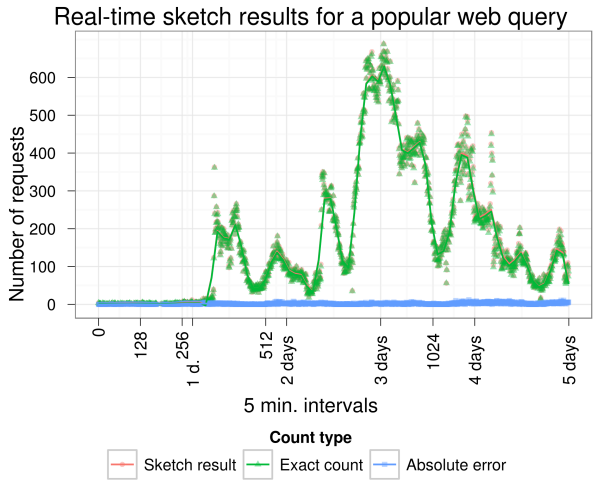


Figure 1: Real-time results for the item aggregation sketch and web query (*"gigi goyette"*) that was popular at the time of our study. Points represent actual results; lines are LOESS smoothed with span 0.1. The plot shows the exact time query started to gain popularity and peaked, as well as daily fluctuations of search traffic. Estimate and real traffic are virtually indistinguishable and the error is minimal.

It is the latter that we focus on in this paper. We present an algorithm capable of retrieving *approximate* count statistics in real time for any given point or interval in time without the need to peruse the original data. We will see that the work required for retrieving such statistics is $O(\log t)$ for lookup and $O(t)$ for decoding (here t is the length of the requested sequence) and moreover that the answers are exact under the assumption of statistical independence. Experiments show that even if this assumption is violated, as it typically is in practice, the statistics still match closely the exact data. We use a number of blocks:

- The Count-Min sketch serves as the basic data aggregator. This is desirable since it has the property of being linear in the data stream. That is, if we sketch time periods T and T', then the sketch

of $T \cup T'$ is given by the sum over the two individual sketches. This makes it easy to aggregate time intervals which are powers of 2.
- We exploit the fact that the Count-Min sketch is linear in the resolution of the hash function. That is, a sketch using a lower resolution hash function is obtained by aggregating adjacent bins.
- We use the fact that for independent random variables the joint distribution can be characterized exactly by retaining only the marginals.
- We use a Markovian approximation for sequences.

Our implementation stores Count-Min sketches of both increasing time intervals in powers of 2, e.g. 1, 2, 4, 8, ... minutes length. Moreover we also store sketches of decreasing bit resolution. Finally, in order to accelerate computation, we store sketches which have both decreasing time and bit resolution. This provides a suitable normalization.

Outline. We begin by giving a brief overview over the Count-Min sketch [6] and how it can be parallelized and made fault tolerant. In Section 3 we describe the data structures used to aggregate the data in a linear fashion and show how reduction in the bit resolution of the hash function allows us to compute both time and keyspace marginals. Using approximate independence assumptions we can recover higher (time, item) resolution of the sketch over time. In Section 4 we apply sketches to $O(1)$ probability estimates in the context of probabilistic graphical models. Section 5 contains experimental results.

2 Count-Min Sketch

Algorithms for sketches of data streams aim at obtaining count statistics for observed items as they keep on arriving in an online fashion. That is, such algorithms typically allow one to assess how many items of a given kind are contained in the data stream. One of the most exciting algorithms for this purpose is the Count-Min sketch of [6], a generalization of the Bloom filter [3] and the count sketch [5], which generates linear data summaries *without* requiring storage of the keys. That is, the maximum amount of memory is allocated for storing count statistics rather than an auxiliary index.

Denote by $\mathcal{X}$ the domain of symbols in the data stream. The Count-Min sketch allocates a matrix of counters $M \in \mathbb{R}^{d \times n}$ initially all set to zero. Moreover, we require d pairwise independent hash functions $h_1, \ldots, h_d : \mathcal{X} \to \{0, \ldots n-1\}$. For each item insertion d counters, as determined by the hash functions and the key, are incremented. Retrieval works by taking the minimum of the associated counts (this mitigates errors due to collisions). Algorithm 1 comes with surprisingly strong guarantees. In particular [6] proves:

Algorithm 1 Count-Min Sketch

insert(x)**:**
for $i = 1$ **to** d **do**
  $M[i, h_i(x)] \leftarrow M[i, h_i(x)] + 1$
end for

query(x)**:**
$c = \min \{M[i, h_i(x)] \text{ for all } 1 \le i \le d\}$
return c

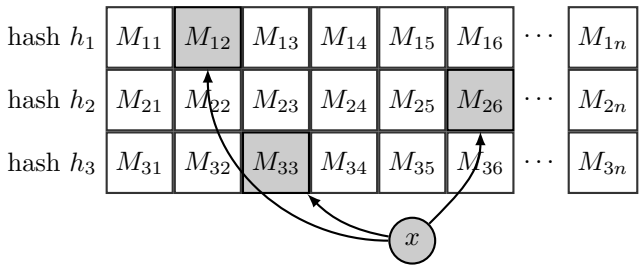


Figure 2: Item x is inserted into each row of the Count-Min sketch (positions 2, 6, and 3 respectively).

Theorem 1 *Assume that* $n = \lceil \frac{e}{\epsilon} \rceil$ *and* $d = \lceil \log \frac{1}{\delta} \rceil$. *Then with probability at least* $1 - \delta$ *the Count-Min sketch returns at any time an estimate* c_x *of the true count* n_x *for any* x *which satisfies*

$$n_x \le c_x \le n_x + \epsilon \sum_{x'} n_{x'}. \tag{1}$$

That is, the guarantee (1) is simply an additive bound on the deviation of counts. Tighter estimates can be obtained from M by using an iterative procedure which goes beyond Algorithm 1. In particular, the Counter Braid iteration [11] provides guarantees of exact reconstruction with high probability provided that the number of stored symbols is sufficiently small and sufficiently well distributed. Note that reconstruction using [11] requires linear time, hence it is not applicable to our scenario of high throughput sketches. Furthermore, whenever the item distribution follows a power law, a tighter bound is available [7]. The following two corollaries are immediate consequences of the fact that the Count-Min sketch is *linear* in the amount of data and in the amount of memory available.

Corollary 2 *Denote by* M_T *and* $M_{T'}$ *the Count-Min sketches for the data stream observed at time intervals* T *and* T'. *Then whenever* $T \cap T' = \emptyset$ *we have that* $M_{T \cup T'} = M_T + M_{T'}$.

This is analogous to Bloom filter hashes of the union of two sets — one simply takes the OR of both hashes. In our case the Boolean semiring is replaced by the semiring of integers (or real numbers) [4].

Corollary 3 *Assume that $n = 2^b$ for $b \in \mathbb{N}$ and let M_b be the summary obtained by the Count-Min sketch. In this case the summary M_{b-1} obtained by a sketch $M_{b-1}[i,j] = M_b[i,j] + M_b[i, j+2^{b-1}]$ using hash functions $h_i^{b-1}(x) := h_i^b(x) \mod 2^{b-1}$.*

This is simply 'folding over' the first and second half of the Count-Min sketch on itself. These two corollaries mean that it is possible to increase the time intervals simply by aggregating sketches over shorter sub-intervals. Likewise, we may reduce the accuracy *after the fact* simply by dropping the most significant bits of h. This will become valuable when compressing the Count-Min sketch by reducing either temporal resolution or accuracy. Obviously as a consequence of this compression we double the error incurred by the sketch. The following chapters present algorithms to counteract this effect.

3 Aggregation

We want to obtain information regarding item frequency over an extended period of time. Assume that we are interested in this data at a resolution of 1 minute (the time scale is arbitrary — we just use 1 minute for the sake of concreteness). Even in the case where M might be relatively small (in the order of 1 MB) we would suffer from memory overflow if we wanted to store 2 years (approximately 2^{20} minutes and hence 1 TB of data) in RAM to allow for fast retrieval.[1] Consequently we need to compress data.

3.1 Time Aggregation

One strategy is to perform binary aggregation on the time axis. That is, rather than retaining a 1 minute resolution for 2 years, we only retain a 2^m minute resolution for the last 2^m minutes. The rationale is that old observations have an exponentially decreasing value and consequently it is sufficient to store them at a concomitantly decreased precision. This is achieved by Algorithm 2. The basic idea is to keep aggregates M^i for time intervals of length $\{1, 1, 2, 4, 8, \ldots, 2^m\}$ available. Note that the sequence contains its own cumulative sum, since $1 + \sum_{i=1}^{n-1} 2^i = 2^n$. To update M^i whenever it covers the time span $[2^{i-1}, 2^i - 1]$ it suffices to sum over the terms covering the smaller segments.

Theorem 4 *At t, the sketch M^j contains statistics for the period $[t - \delta, t - \delta - 2^j]$ where $\delta = t \mod 2^j$.*

[1] We could store this data on disk. Many tools exist for offline parallel analytics which complement our approach. However, we are interested in online instant access.

Algorithm 2 Time Aggregation

```
for all m do do
    Initialize Count-Min sketch M^m = 0
end for
Initialize t = 0 and M̄ = 0
while data arrives do
    Aggregate data into sketch M̄ for unit interval
    t ← t + 1 (increment counter)
    for j = 0 to argmax {l where i mod 2^l = 0} do
        T ← M̄ (back up temporary storage)
        M̄ ← M̄ + M^j (increment cumulative sum)
        M^j ← T (new value for M^j)
    end for
    M̄ ← 0 (reset aggregator)
end while
```

Proof The proof proceeds by induction. At time $t = 0$ this condition clearly holds since all counters are empty. Now assume that it holds for time $t > 0$. Denote by $j^* := \operatorname{argmax}\{l \text{ where } i \mod 2^l = 0\}$ the largest exponent of 2 for which t is divisible by 2^{j^*}. This is the last index which will become aggregated.

- For all M^j with $j > j^*$ the condition holds if we increment $t \leftarrow t+1$ since the intervals are shifted by one time step without crossing any power of 2.
- For all M^j with $j \leq j^*$ note that in the `for` loop we update all such counters by cumulative sums covering a consecutive contiguous sequence of shorter intervals via the cumulative sum counter S. In other words, all these intervals start from $t - \delta = 0$ again. ∎

Lemma 5 *The amortized time required to update the statistics is $O(1)$ per time period regardless of T.*

Proof This follows from the fact in 2^{-i} of all time periods we need to do i work. Hence the total amortized workload is $\sum_{i=1}^{\infty} i \cdot 2^{-i} = 2$. ∎

One of the nice side-effects of this procedure is that an "expensive" aggregation step which can involve up to $\log t$ many additions is always followed by a cheap update requiring only a single update (see Figure 3). This means that the update operations may be carried out as background threads in parallel to new updates, as long as they complete their first step within the update time period (this is trivially the case).

3.2 Item Aggregation

An alternative means of aggregating count statistics over time is to retain the full time resolution while

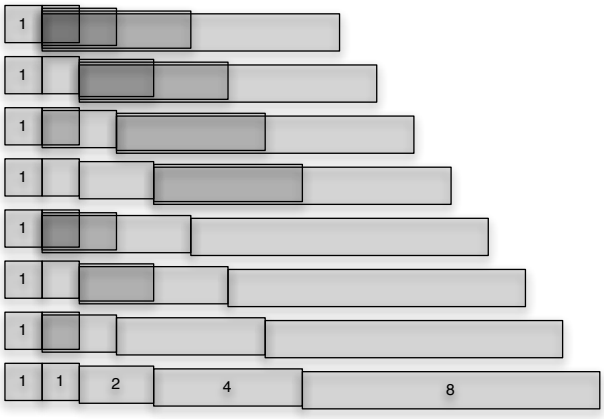


Figure 3: Time aggregation. We always insert into the leftmost aggregation interval. As time proceeds (top to bottom), new intervals are inserted from the left. Whenever an interval of length 2^n is 2^n steps old, it aggregates all data in $[0, 2^n]$ as its new value. The more shaded bins are contained in more than one interval.

sacrificing accuracy. That is, we can shrink the number of bins n over time but we retain the unit time intervals. More specifically, we can halve the resolution each time the Count-Min sketch ages 2^n time steps. One of the advantages of this strategy is that the work required for aggregation is directly proportional to the remaining size of the bins (see Figure 3). We invoke Corollary 3 to halve the amount of space required successively. This yields Algorithm 3.

Algorithm 3 Item Aggregation

$t \leftarrow 0$ (initialize counter)
while data arrives **do**
  Receive aggregated data sketch $\bar{M}$ for unit interval
  $t \leftarrow t + 1$ (increment counter)
  $A^t \leftarrow \bar{M}$
  for $k = 1$ **to** $\lfloor \log_2 t \rfloor$ **do**
    for all i, j **do** (reduce item resolution)
    $A^{t-2^k}[i,j] \leftarrow A^{t-2^k}[i,j] + A^{t-2^k}[i, j + 2^{m-k}]$
    Shrink A^{t-2^k} to length 2^{m-k}
  end for
end while

Note that as before this algorithm requires only $O(1)$ computational cost. Even better than before, this cost is now $O(1)$ for all steps rather than being an amortized (i.e. average) cost: at each time step there is one intervals of size 2^i with $2^i \leq n$ for each i that needs halving. The cost of doing so is bounded by $\sum_{i=0}^{l} 2^i = 2^{l+1} - 1 < 2n$.

Moreover, the amount of storage required per time interval $[t - 2^i, t - 2^{i-1}]$ is constant since for every doubling of the interval size we halve the amount of storage required per sketch. This means that we can retain full temporal resolution, albeit at the expense of increasingly lose bounds on the item counts. In the extreme case where $2^m = n$ we end up with a sketch which tells us simply how many tokens we received at a given point in time but no information whatsoever regarding the type of the tokens.

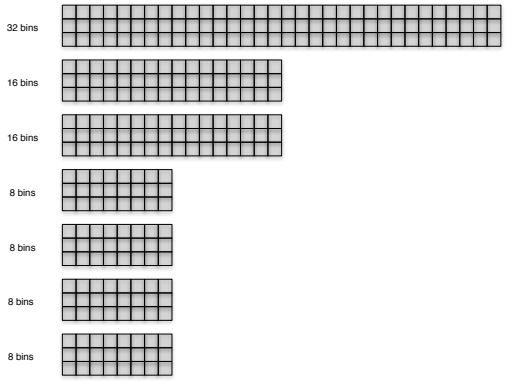


Figure 4: Item aggregation. Every 2^n with $n \in \mathbb{N}$ we halve the amount of bits used in the sketch. This allows us to sketch T time steps in $O(\log T)$ space. Obviously having fewer storage bins means that we have less space available to store fine-grained information regarding the observed keys. However, for heavy hitters this is sufficient.

Note that by default the absolute error also doubles at every aggregation step (the scaling is slightly different for power law distributions). This means that after several iterations the accuracy of the sketch becomes very poor for low-frequency items while still providing good quality for the heavy hitters. We will exploit this as follows — for low frequency items we will use an interpolation estimate whereas for high frequency items we will use the aggregate sketch directly.

3.3 Resolution Extrapolation

We can take advantage of the following observation to improve our estimates: at some point in time we would like to obtain to-the-minute resolution of observations for arbitrary objects. In the extreme case we may have to-the-minute counts for *all* events and simultaneously highly specific counts for a *long* period of time. The basic idea in extrapolating counts is to use the fact that a joint distribution $p(x, t)$ over time and events can be recovered using its marginals $p(x)$ and $p(t)$ whenever x and t are independent of each other. In terms of counts this means that we estimate

$$\hat{n}_{xt} = \frac{n_x \cdot n_t}{n} \text{ where } n = \sum_t n_t = \sum_x n_x. \quad (2)$$

The quantities n_x and n_t are available (as upper bounds) via the Count-Min sketches M^i and A^j respectively. Finally, n *could* be obtained at runtime by carrying out the summation over t or x respectively. However, this is too costly as it requires summing over $O(T)$ bins at time T. Instead, we compute a third set of aggregate statistics simultaneously to Algorithms 2 and 3 which performs both time and item aggregation.

Algorithm 4 Item and Time Aggregation

```
t ← 0 (initialize counter)
while data arrives do
  Wait until item and time aggregation complete
  t ← t + 1 (increment counter)
  S = M^2 (we only start combining at time 2)
  for j = 1 to argmax{l where i mod 2^l = 0} do
    S[i, j] ← S[i, j] + S[i, j + 2^{m−j}]
    Shrink S to length 2^{m−j}
    T ← S (back up temporary storage)
    S ← S + B^j (increment cumulative sum)
    B^j ← T (new value for B^j)
  end for
end while
```

While individual estimates on n_t and n_x would allow for upper bounds via Theorem 1, and likewise a lower bound on n, we only obtain an approximation (we take a ratio between two upper bounds): we use (2) for each hash function separately and perform the `min` operation subsequently. At time T the query for $n(x,t)$ requires:

$$\hat{n}(x,t) := \min_i \frac{M^{\jmath^*}[i, h_i(x)] A^t[i, h_i^{m-\jmath^*}(x)]}{B^{\jmath^*}[i, h_i^{m-\jmath^*}(x)]} \quad (3)$$

$$\text{where } \jmath^* := \lfloor \log_2(T-t) \rfloor$$

We compute $\jmath^*$ as the indicator for the most recent interval containing information pertaining t after we have seen T instants of data. The use of $h^{m-\jmath^*}$ is needed to restrict the key range to the available array. As a consequence we obtain *estimates* of counts at full time and token resolution, albeit at decreasing accuracy as we look further into the past.

Since (3) is only an approximation of the true counts we use it only whenever the Count-Min estimate with reduced bit resolution is not accurate enough. Given that the accuracy decreases with $e2^{-b}$ where 2^b is the number of bins used in the sketch, we have the following variant of the sketching algorithm for queries: use the simplified item aggregate whenever the error is small enough. Otherwise switch to interpolation. Since we know that the absolute error of the Count-Min sketch is well controlled, we know that this strategy is self consistent for frequent items. See Algorithm 5 for details.

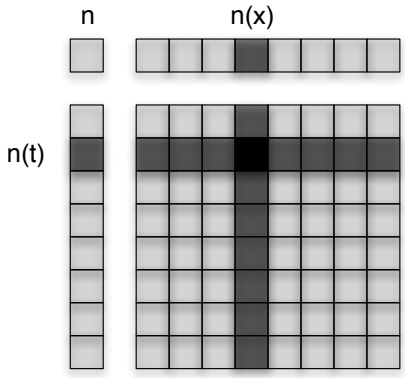


Figure 5: Resolution interpolation. We approximate $n(x,t)$ by $\frac{n(t)n(x)}{n}$ assuming independence, i.e. by approximating $p(a,b) \approx p(a) \cdot p(b)$. This is analogous to density estimation via copulas [13].

Algorithm 5 Improved Interpolating Sketch

```
query(x, t):
Obtain ñ(x, t) := min_i A^t[i, h_i^{m−j*(x)}]
if ñ(x, t) > et/2^b then
  return ñ(x, t) (heavy hitter)
else
  Compute n̂(x, t) using (3).
  return n̂(x, t) (interpolate)
end if
```

4 Beyond Sketches

Often we want to estimate the likelihood of a sequence of symbols, e.g. *probabilistic graphical models*. Unfortunately it is undesirable to insert long sequences directly into the Count-Min sketch. The reason for this is that, as stated in Theorem 1, the error in approximating the counts is an *absolute* error rather than being relative to the key frequency. While this is not a problem for relatively frequently occurring terms (provided that we know that they are frequent), the error may not be acceptable for low probability (and hence infrequently occurring) keys. We resort to tools from undirected graphical models to obtain $O(1)$ estimates for item frequencies of more complex objects.

Cliques For sequences we use a Markovian approximation. That is, we model e.g. trigrams in terms of a product of unigrams or in terms of a chain of bigrams. For concreteness consider modeling the string 'abc':

$$p(abc) \approx p(a) \cdot p(b) \cdot p(c) \quad \text{Unigrams} \quad (4)$$

$$p(abc) \approx p(a,b)p(c|b) = \frac{p(a,b)p(b,c)}{p(b)} \quad \text{Bigrams} \quad (5)$$

Clearly, whenever (4) is exact, it is included in (5) as a special case, since the bigram model subsumes the

unigram representation as a special case. In general we may use the Markov model (4) and (5) as a frequency estimate for items that are too rare to track.

Since we only have access to the item frequencies rather than the true probabilities of occurrence it is advisable to use hierarchical smoothing. While it would be most desirable to use a Dirichlet-process style smoothing along the lines of [16], the latter is too costly to apply in real time as it requires considerably more than $O(1)$ computation. Instead we may resort to a simplified approach like backoff smoothing:

$$\hat{p}(a) = \frac{n_a + n_0}{n + Ln_0} \text{ and } \hat{p}(ab) = \frac{n_{ab} + n_1 \hat{p}(a)\hat{p}(b)}{n + n_1}. \quad (6)$$

Here n_0 and n_1 are parameters which describe the amount of backoff smoothing we employ to estimate the joint as a product of the marginals. More generally, we may use Good-Turing or Kneser-Ney [10] smoothing. Eq. (5) can be extended as follows:

Theorem 6 *Assume that x has an independence structure that can be expressed as an undirected graph $G(V, E)$ in the sense of an undirected graphical model (here V denotes the vertices associated with the coordinates of x and E denotes the edges in the sense of an undirected graphical model). Moreover denote by T a junction tree containing $G(V, E)$ as subgraph with a set of cliques $\mathcal{C}$ and a set of separator sets $\mathcal{S}$, then it suffices if we obtain counts for x_C and x_S with $C \in \mathcal{C}$ and $S \in \mathcal{S}$ to generate frequency estimates via*

$$\hat{p}(x) = n^{|\mathcal{S}|-|\mathcal{C}|} \prod_{C \in \mathcal{C}} n_{x_C} \prod_{S \in \mathcal{S}} n_{x_S}^{-1} \quad (7)$$

Proof This follows immediately from the Hammersley-Clifford Theorem [2], the fact that a junction tree contains the maximum cliques of an undirected graph, and the fact that empirical frequencies converge to their probabilities. ∎

In this view (4) and (5) are simple applications of a Markov chain. It also provides us with a recipe for selecting and generating extrapolation formulae for more complex structures.

Maximum Entropy Estimation A natural choice for estimating joint probabilities from marginals would be to perform a (smoothed) maximum entropy estimate which is then guaranteed to be consistent. However, this is too costly, as it requires at least $O(m)$ operations, where m is the size of the support of the distribution. Such an estimate will lead to

$$p(x|\theta) = \exp\left(\sum_{i=1}^{d} \theta[i, h_i(x)] - g(\theta)\right) \quad (8)$$

This follows directly, e.g. from [8, 1]. The advantage is that after significant computation to obtain θ we can provide more accurate frequency estimates at the same cost as the original Count-Min sketch. Note that the likelihood of the data stream is given by

$$\prod_t p(x_t|\theta) = \exp\left(\operatorname{tr} M^\top \theta - n g(\theta)\right). \quad (9)$$

Unfortunately computing $g(\theta)$ is very costly — it requires that we have access to the support of the distribution, which would require an auxiliary data structure in its own right, thus obviating the advantages afforded by sketches. This is why we do not pursue this avenue further in the present paper.

5 Experiments

5.1 Setup

Code We implemented the Count-Min sketch algorithms 2, 3 and 4 in the client-server setting, using C++ and ICE (http://www.zeroc.com) middleware. The experiments were run on quad core 2GHz Linux x86 servers with 16GB RAM and Gigabit network connection, and used sketches with 4 hash functions, 2^{23} bins, and 2^{11} aggregation intervals (amounting to 7 days in 5 minute intervals). For trigram interpolation, we load Wikipedia into a single 12GB sketch with 3 hash functions, 2^{30} bins, and no time aggregation.

Data We use two datasets for our research. One is a proprietary dataset containing a subset of search queries of five days in May 2011. The second dataset contains the entire textual dump of the English version of Wikipedia as per January 4, 2012 (we removed the XML markup as a preprocessing step). In all cases full statistics of the data were computed using Hadoop. These counts serve as the gold standard for any sketches and allow us to obtain a proper evaluation of the estimated frequency counts.

Our choice of data (see Figure 6) was guided by both the need to build algorithms applicable in an industrial context and by the need to provide reproducible experimental descriptions. Furthermore, both datasets are quite different in terms of the size of their long tail. The smaller query dataset has a very significant component of infrequent terms whereas Wikipedia, largely prose, displays a much lighter tail with a support of only 4.5 Million unique terms.

5.2 Performance

In our experiments all three types of sketches were able to consistently handle over 50k inserts per second. For read requests we measured on average 22k requests per

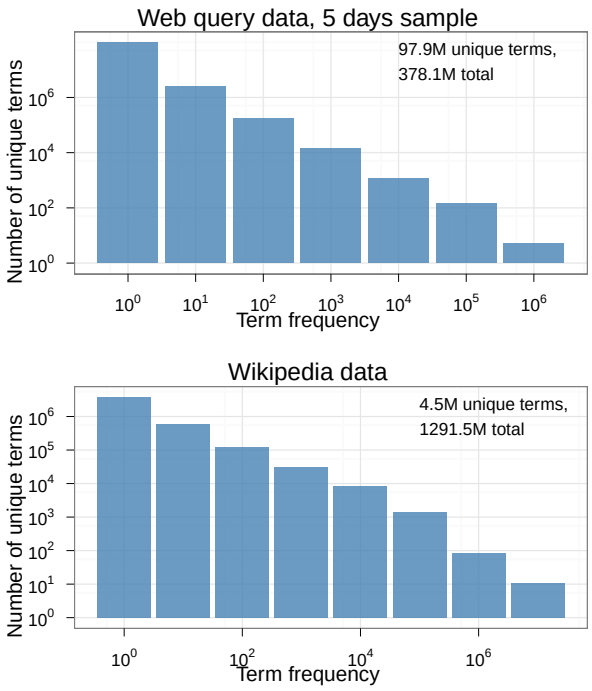


Figure 6: (Top) web query data from 5 days in May 2011; (Bottom) English text in Wikipedia as per January 4, 2012. While both datasets exhibit power-law behavior they differ quite a bit in size and support.

second for time aggregation sketch, and 8,5k requests per second for item aggregation and resolution interpolation. Performance is proportional to the network bandwidth: inserts are one-way, asynchronous, and use very short messages, whereas queries require two-way communication and response messages are larger.

5.3 Accuracy

The first set of experiments is to evaluate the accuracy of the sketches under temporal aggregation. Since the Wikipedia dataset constitutes a snapshot in time we only study the accuracy for the query dataset. The Wikipedia dataset is only used to study interpolation. The experimental protocol is as follows:

Gold Standard For all experiments we computed the true counts using a batch algorithm on Hadoop. It serves as reference for our estimates.
Time aggregation as described in Section 3.1.
Key aggregation as described in Section 3.2.
Interpolation as described in Section 3.3.
Absolute deviation is given by the absolute amount that the sketch deviates from the true counts. In the case of item aggregation sketching estimates are always an overestimate. In all other cases this is not necessarily true. We compute it using $\sum_x |\hat{n}_x - n_x|$.
Relative deviation is given by the ratio between the deviation and the estimate, i.e. $\sum_x \frac{|\hat{n}_x - n_x|}{\hat{n}_x}$. Note that this quantity may exceed 1 since we aggregate over all keys.

We are not aware of any sketching algorithm addressing the problem of providing aggregates for arbitrary time intervals. Hence we compare our results to the exact offline aggregation results. Furthermore, we compare to a naive aggregation per time slice and also to a piecewise constant model. One would expect the latter to perform well, given that we are dealing with time-series data which is unlikely to change dramatically in between intervals.

As can be seen in Figure 7, the interpolation algorithm is well competitive relative to algorithms that perform constant interpolation or that reduce item counts. As expected the error increases with the amount of time past. This is the case since we compress data every 2^n time steps by a factor of 2. The fact that the error of the 'item aggregation' variant is considerably higher than of the 'time aggregation' algorithm suggests that the item frequency does not vary strongly over time for most items, when compared to the variation between items. Hence the relative frequency of occurrence is generally a better estimate. That said, by using interpolation we combine the best of both worlds and obtain overall good estimates.

To obtain a more detailed estimate we stratify accuracy per item frequency interval, i.e. we stratify by the number of occurrences of the item, as depicted in Figure 8. Not very surprisingly for frequent items, reducing the bit resolution of the sketch is less damaging, as follows directly from Theorem 1, hence for heavy hitters interpolation is on par with item aggregation.

5.4 Multigram Interpolation

We show that the factorizing approximation of Section 4 can be used to obtain good estimates of multigrams in both textual documents and query streams. To show that this works for interpolating terms, we approximate trigrams by sketching unigrams and bigrams with a regular Count-Min sketch. We do that in three different ways:

Unigram approximation by consuming unigrams and using (4) to estimate trigram probabilities;
Bigram approximation by consuming bigrams and unigrams and using (5);
Trigram sketching by consuming and querying sketch for trigrams directly.

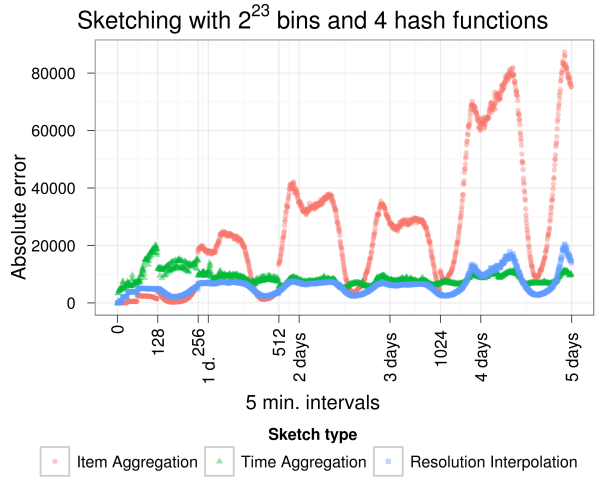


Figure 7: Absolute accuracy $\hat{n} - n$ of three different aggregation algorithms.

	Absolute error	Relative error
Unigram approximation	24977905.97	0.265740
Bigram approximation	1222160.44	0.013002
Trigram sketching	8352974.60	0.088867

Table 1: Absolute and relative deviation of trigram approximation models using Wikipedia data.

Table 1 compares the resulting estimates with exact trigram counts. Note that due to the lower number of collisions, our bigram approximation has less than 15% the error of direct aggregation. This is quite significant since it means that in order to store higher order terms direct sketching may not be quite so desirable (possibly with the exception of storing the heaviest hitters directly). It offers a very fast and cheap means of retrieving estimates relative to exact storage.

6 Discussion

Summary In this paper we presented an algorithm for aggregating statistics on data streams which *retains* temporal information of the items. At its heart we used the Count-Min sketch to obtain a simple yet effective compressed representation of the data. By using time and item aggregation we showed how it is possible to obtain sketches for a more extensive set of query types. Moreover, we showed how the theory of graphical models can be employed to obtain estimates of structured set of covariates.

Our system allows real-time data access in constant time without any need for expensive preprocessing. This makes it an attractive alternative to batch processing systems which, while accurate, are unable to respond to requests without latency and which are therefore less well suited to online data analysis. Experiments demonstrated the excellent performance of the proposed algorithms.

An alternative of our work is to use empirical Laplace transforms to aggregate data. That is, many sketches are amenable to weighted inserts over time. This allows us to obtain sketches of the Laplace-transform counts of a sequence. Decoding then occurs at the client issuing a frequency query to the server. Details of this are the subject of future work.

Extension to Delayed Updates In some cases data may arrive with some delay. Here the Count-Min sketch excels due to the fact that it is a *linear* statistic of the data: We can always insert data later into the appropriate (aged) records at a later stage.

A second (and much more significant) advantage is that the additive property of Corollary 2 also applies to sets. Denote by $S(\mathcal{X})$ the data sketch structure given by Algorithms 2, 3, and 4. In this case $S(\mathcal{X} \cup \mathcal{X}') = S(\mathcal{X}) + S(\mathcal{X}')$. Hence we may use MapReduce to carry out sketches on subsets of the data first and then aggregate the terms between mappers. Since everything is linear once the data has been inserted this exhibits perfect scaling properties.

Note that the issue might arise that different subsets of the data span slightly different time ranges. In this case it is important to ensure that all sketches are synchronized to the same time intervals since otherwise aliasing might occur (e.g. a summary of 1 week's data might start on different days on different machines).

Parallelization Note that the sketches discussed in the present paper are well amenable to parallelization. In fact, we may use consistent hashing [9] to distribute hash functions and keys over a range of workstations. Details of this strategy are described in [15]. In a nutshell the idea is to send keys to several machines, each of which compute only a single row of the matrix M, each of them using a different hash function. The combination of both ideas leads to a powerful enterprise framework for sketching data streams in real time.

Acknowledgments This work was supported by the Australian Research Council. We thank George Varghese for inspiring discussions.

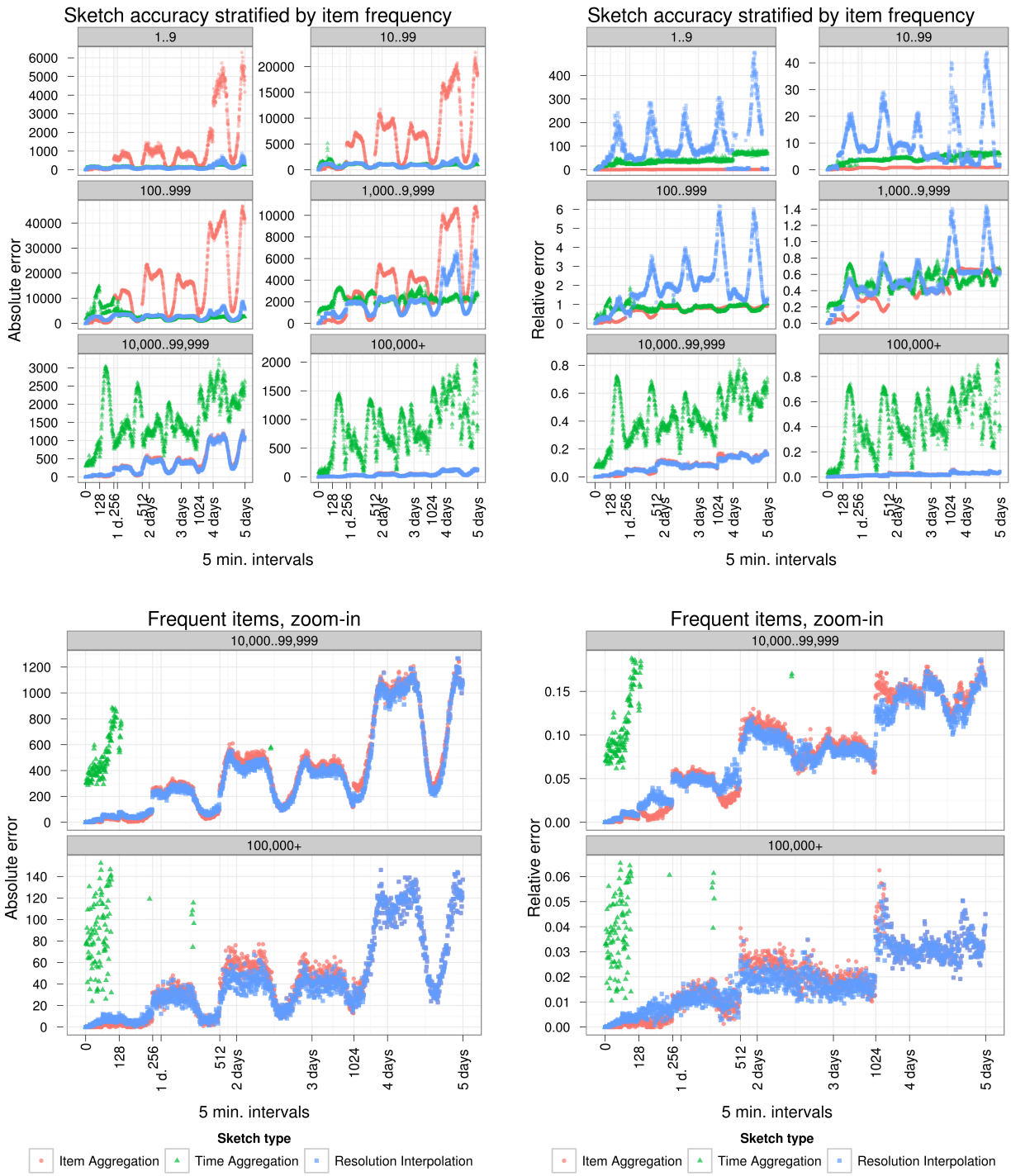


Figure 8: Absolute $\hat{n} - n$ and relative $\frac{\hat{n}-n}{\hat{n}}$ accuracy, stratified by item frequency interval. Left column: absolute error. Right column: relative deviation. Top row: error over time, stratified by frequency of occurrence of items. Bottom row: error for the heaviest hitters.

References

[1] Y. Altun and A. J. Smola. Unifying divergence minimization and statistical inference via convex duality. In H.U. Simon and G. Lugosi, editors, *Proc. Annual Conf. Computational Learning Theory*, LNCS, pages 139–153. Springer, 2006.

[2] J. Besag. Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society. Series B*, 36(2):192–236, 1974.

[3] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[4] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, volume 1. A. K. Peters, Ltd., 2004.

[5] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

[6] G. Cormode and M. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *LATIN: Latin American Symposium on Theoretical Informatics*, 2004.

[7] G. Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *SDM*, 2005.

[8] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, New York, 1991.

[9] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Symposium on the Theory of Computing STOC*, pages 654–663, New York, May 1997. Association for Computing Machinery.

[10] R. Kneser and H. Ney. Improved backing-off for M-gram language modeling. In *Proc. ICASSP '95*, pages 181–184, Detroit, MI, May 1995.

[11] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani. Counter braids: a novel counter architecture for per-flow measurement. In Z. Liu, V. Misra, and P.J. Shenoy, editors, *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 121–132. ACM, 2008.

[12] A. Metwally, D. Agrawal, and A. El Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Trans. Database Systems*, 31(3):1095–1133, 2006.

[13] R. B. Nelsen. *An introduction to copulas*. Springer, 2006.

[14] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: a not-so-foreign language for data processing. In Jason Tsong-Li Wang, editor, *ACM SIGMOD Conference on Management of Data, Vancouver, BC, Canada*, pages 1099–1110. ACM, 2008.

[15] A. J. Smola, S. Matusevich, and A. Ahmed. Divide and prosper — fault tolerant scalable sketches. In *Conference on Very Large Databases*, 2012. submitted.

[16] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(576):1566–1581, 2006.

The Complexity of Approximately Solving Influence Diagrams

Denis D. Mauá
IDSIA
Switzerland, CH 6928
denis@idsia.ch

Cassio P. de Campos
IDSIA
Switzerland, CH 6928
cassio@idsia.ch

Marco Zaffalon
IDSIA
Switzerland, CH 6928
zaffalon@idsia.ch

Abstract

Influence diagrams allow for intuitive and yet precise description of complex situations involving decision making under uncertainty. Unfortunately, most of the problems described by influence diagrams are hard to solve. In this paper we discuss the complexity of approximately solving influence diagrams. We do not assume no-forgetting or regularity, which makes the class of problems we address very broad. Remarkably, we show that when both the treewidth and the cardinality of the variables are bounded the problem admits a fully polynomial-time approximation scheme.

1 INTRODUCTION

Influence diagrams are well-known graphical models of decision making under uncertainty which allow for compact and intuitive representation of complex situations, and relatively fast inference [7–9, 17].

Most of the algorithms for inference with influence diagrams require a total ordering over the decisions and unlimited memory. The first assumption is called *regularity* and is graphically equivalent to the existence of a directed path comprising all decision variables in the diagram. The second assumption, called *no-forgetting*, states that decisions are made in light of all previously disclosed information. Graphically, it implies that the parent set of a decision node contains all previous decision nodes and their parents. These conditions allow an optimal solution to be obtained by dynamic programming [19], but impose an inherent exponential worst-case complexity, since the output might include a decision function over exponentially many values (e.g., a table prescribing a value for the last decision for each one of the exponentially many assignments of the other decision variables).

There are two common approaches to avoid the exponential complexity. The first one is to insert arcs entering decision variables in a way that makes them insensitive to the previous decisions, thus decreasing the size needed to represent its corresponding solution [8, 14]. This, however, implies observing quantities that were initially deemed unobservable, which might be undesirable or even unfeasible. The second approach is to relax no-forgetting and work with a limited memory, that is, to assume that not all previous decisions and observations are known and considered when making a decision [10, 13, 20]. Graphically, it corresponds to dropping arcs until the maximum size of a decision function becomes manageable. The latter approach, which gives rise to limited memory influence diagrams [10], is additionally justified by situations where decisions have to be taken independently of one another, as in team decision analysis [1, 6]; for such cases, some of the no-forgetting arcs are inherently absent.

Removing arcs from the original graph can make the problem harder. This essentially occurs because the problem might no longer be solvable by standard dynamic programming approaches. Zhang, Qi, and Poole [21] and more recently Lauritzen and Nilsson [10] determined sufficient conditions under which even influence diagrams that violate no-forgetting can be solved exactly by dynamic programming. Any diagram of bounded treewidth meeting these conditions can be thus solved efficiently. As de Campos and Ji [4] showed, however, even structurally very simple cases can fail to meet these conditions and be hard to solve. In fact, we have recently shown that even singly connected diagrams of treewidth equal to two, with decision variables having no parents and variables taking on a bounded number of states are NP-hard to solve, and that the problem is inapproximable if the cardinality of the variables is unbounded [12].

In this paper, we resume our investigation and analyze the impact of the number of value nodes and

variable cardinality on the theoretical complexity of solving (limited memory) influence diagrams. We do not assume no-forgetting or regularity. We show that if the treewidth is bounded, the number of value variables do not affect the asymptotic complexity. Most importantly, we show that if the cardinalities of the variables are also bounded, the problem admits a fully polynomial-time approximation scheme.

2 INFLUENCE DIAGRAMS

To help introduce the notation and illustrate concepts, consider the following example of a decision problem where an agent first needs to select one among the actions $d_1^{(1)}, \ldots, d_1^{(i)}$, which causes one of $c_1^{(1)}, \ldots, c_1^{(j)}$ outcomes to obtain with probability $P(c_1|d_1)$, and generates a reward of $U(c_1)$. Then, without observing the value c_1, a second agent needs to select one among the actions $d_2^{(1)}, \ldots, d_2^{(k)}$, which generates one of the outcomes $c_2^{(1)}, \ldots, c_2^{(\ell)}$ with probability $P(c_2|c_1, d_2)$ and a reward of $U(c_2)$. The overall utility $U(d_1, c_1, d_2, c_2)$ of the agents' actions their corresponding outcomes is given by the sum of the intermediate rewards, that is, $U(d_1, c_1, d_2, c_2) = U(c_1) + U(c_2)$.

In the language of influence diagrams, the quantities and events of interest are represented by three distinct types of variables. *Chance variables* represent events on which the decision maker has no control, such as outcomes of tests or consequences of actions; *decision variables* represent the alternatives a decision maker has at each decision step; finally, *value variables* are used to represent immediate rewards. In the example, we can represent the decision of the first agent, its outcome, and the corresponding reward by a decision variable D_1, a chance variable C_1 and a value variable V_1, respectively. Similarly, we can represent the second decision, its outcome and the corresponding reward by a decision variable D_2, a chance variable C_2 and a value variable V_2. Regarding the notation, we denote variables by capital letters and identify them with the set of values they can assume; generic and particular values are denoted in lower case. For instance, a generic action of the first agent is represented as $d_1 \in D_1 = \{d_1^{(1)}, \ldots, d_1^{(i)}\}$, while the corresponding set of possible rewards (which depend on the outcome c_1) is given by $V_1 = \{U(c_1^{(1)}), \ldots, U(c_1^{(j)})\}$.

The functional dependencies between the variables in the model can be compactly represented by a directed acyclic graph (DAG) whose nodes are in a one-to-one correspondence to the (decision, chance and value) variables of the problem. For simplicity, we identify nodes with their associated variables. Hence, we can talk about the parents $\mathtt{Pa}(X)$ of a variable X, its children $\mathtt{Ch}(X)$, or yet its family $\mathtt{Fa}(X) = \mathtt{Pa}(X) \cup \{X\}$ of X. The arcs entering decision variables represent the set of variables whose values will be known by the time the corresponding decision is made, either because they are observed at this time (e.g., an outcome of a previous decision or an action taken by a different agent) or remembered (e.g., a previous decision or observation). The problem in the example can be depicted as a DAG in Figure 1 (as usual, chance, decision and value variables are represented by ovals, squares and diamonds, respectively). The existence of the dashed arc connecting D_1 to D_2 depends on whether the decision of the first agent is known by the second agent at the time the latter acts. It is present if the second agent knows (and takes into account) the decision of the first agent, and it is absent otherwise.

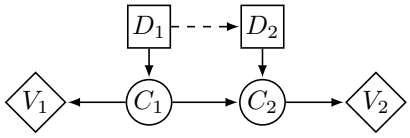


Figure 1: Influence diagram for the problem in the example.

An influence diagram is a tuple $(\mathcal{C}, \mathcal{D}, \mathcal{V}, \mathcal{G}, \mathcal{P}, \mathcal{U})$, where $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{V}$ are the sets of chance, decision and value variables, respectively, $\mathcal{G}$ is a DAG over $\mathcal{C} \cup \mathcal{D} \cup \mathcal{V}$, $\mathcal{P}$ is a set containing one (conditional) probability distribution $P(C|\mathtt{Pa}(C))$ per chance variable C in $\mathcal{C}$, and $\mathcal{U}$ is a set containing one reward function $U(\mathtt{Pa}(V))$ per value variable V. Thus, an influence diagram specifies both a *joint probability distribution* $P(\mathcal{C}|\mathcal{D}) = \prod_{C \in \mathcal{C}} P(C|\mathtt{Pa}(C))$ of the outcomes conditional on the decisions, and a *utility* $U(\mathcal{C}, \mathcal{D}) = \sum_{V \in \mathcal{V}} U(\mathtt{Pa}(V))$ over outcomes and decisions.

2.1 The Strategy Selection Problem

For any decision variable $D \in \mathcal{D}$, a *policy* is a conditional probability distribution $P(D|\mathtt{Pa}(D))$ prescribing a (possibly randomized) action for each state of the parents. We say that a policy $P(D|\mathtt{Pa}(D))$ is *pure* if for every assignment to the parents it assigns all the mass to a single action $d \in D$. A *strategy* is a set $\mathcal{S} = \{P(D|\mathtt{Pa}(D)) : D \in \mathcal{D}\}$ containing a policy for each decision variable. A strategy $\mathcal{S}$ induces a(n unconditional) joint probability distribution over decisions and outcomes by

$$P_{\mathcal{S}}(\mathcal{C}, \mathcal{D}) = P(\mathcal{C}|\mathcal{D}) \prod_{D \in \mathcal{D}} P(D|\mathtt{Pa}(D)) \,,$$

and has an associated expected utility given by

$$\mathrm{E}_{\mathcal{S}} = \sum_{\mathcal{C}, \mathcal{D}} P_{\mathcal{S}}(\mathcal{C}, \mathcal{D}) U(\mathcal{C}, \mathcal{D}) \,.$$

The primary use of an influence diagram is the *strategy selection problem*, which consists in finding a strategy $\mathcal{S}^*$ that maximizes the expected utility, that is, to find $\mathcal{S}^*$ such that $\mathrm{E}_{\mathcal{S}^*} \geq \mathrm{E}_{\mathcal{S}}$ for all $\mathcal{S}$. The value $\mathrm{E}_{\mathcal{S}^*}$ is called the *maximum expected utility* and is denoted shortly by MEU. Most algorithms for finding optimal strategies have complexity at least exponential in the treewidth of the influence diagram, which is a measure of its resemblance to a tree and is better formalized by the notion of tree decomposition.

A *tree decomposition* of an influence diagram is a tree-shaped graph where each node is associated to a subset of the chance and decision variables in the diagram. The decomposition satisfies the *family preserving* and the *running intersection* properties, namely, that the family of each decision and chance variable, as well as the parent set of each value variable, is contained in at least one set associated to a node of the tree, and that the graph obtained by dropping nodes that do not contain any given chance or decision variable is still a tree. The treewidth of a tree decomposition is the maximum number of variables associated to a node minus one. The treewidth of an influence diagram is the minimum treewidth of a tree decomposition of it.

For a fixed integer k, Bodlaender [2] showed that for any diagram one can in linear time either obtain a tree decomposition of treewidth at most k or know that it does not exist. Hence, for any diagram of bounded treewidth we can obtain in linear time an optimal tree decomposition, that is, a tree decomposition whose treewidth equals the treewidth of the diagram. Moreover, any tree decomposition can be turned into a binary tree decomposition (i.e., one in which each node has at most three neighbors) of same treewidth in linear time [18]. Given a tree decomposition $\mathcal{T}$ with m nodes, we denote by $\mathcal{X}_1, \ldots, \mathcal{X}_m$ the sets of variables associated to nodes $1, \ldots, m$, respectively. Thus, $\mathcal{X}_1 \cup \cdots \cup \mathcal{X}_m = \mathcal{C} \cup \mathcal{D}$.

Given an influence diagram, we can evaluate the expected utility of any strategy $\mathcal{S}$ in time and space at most exponential in its treewidth. Hence, if a diagram has bounded treewidth, we can obtain $\mathrm{E}_{\mathcal{S}}$ for any strategy $\mathcal{S}$ in polynomial time [9, Chapter 23]. Obtaining the optimal strategy in this way is however unfeasible for any moderately large problem, as the number of strategies is too high. In fact, de Campos and Ji [4] showed that even in diagrams of bounded treewidth the strategy selection problem is NP-hard. We have recently [12] strengthened their result by showing that the problem is already NP-hard in singly connected diagrams of treewidth equal to two and variables assuming at most three values. The result uses a reduction from the partition problem to a influence diagram whose underlying graph is a tree and contains a single value variable. One might then wonder whether allowing more than one value variable affects the difficulty of the problem. The answer, as the following result shows, is no.

Theorem 1. *For any influence diagram $\mathcal{I}$, there is an influence diagram $\mathcal{I}'$ containing a single value variable such that any strategy $\mathcal{S}$ for $\mathcal{I}$ is also a strategy for $\mathcal{I}'$ and obtains the same expected utility. Moreover, the treewidth of $\mathcal{I}'$ is at most the treewidth of $\mathcal{I}$ plus three.*

Proof. Let $\mathcal{I} = (\mathcal{C}, \mathcal{D}, \mathcal{V}, \mathcal{G}, \mathcal{P}, \mathcal{U})$ be an arbitrary influence diagram and consider, without loss of generality, a binary tree decomposition $\mathcal{T}$ of $\mathcal{I}$ such that for every value variable V_i in $\mathcal{V}$ there is a leaf node in the tree whose associated set of variables is $\mathtt{Pa}(V_i)$.[1] Assume additionally that the tree is rooted at a node r and that the leaf nodes $\ell_1, \ell_2, \ldots, \ell_q$ associated to the sets $\mathtt{Pa}(V_1), \ldots, \mathtt{Pa}(V_q)$, respectively, where $q = |\mathcal{V}|$ denotes the number of value variables, are ordered in such a way that they agree with an *in-order tree traversal* of the tree decomposition, that is, in a depth-first tree traversal of $\mathcal{T}$ rooted at r, the node ℓ_1 precedes ℓ_2, which precedes ℓ_3, and so on. Let $\overline{U}$ and $\underline{U}$ be upper and lower bounds, respectively, on the reward functions in $\mathcal{U}$, with $\overline{U} > \underline{U}$.

Now consider a diagram $\mathcal{I}' = (\mathcal{C}', \mathcal{D}, \{V\}, \mathcal{G}', \mathcal{P}', \{U\})$, which contains a single value variable V instead of the q value variables of $\mathcal{I}$ and an augmented set of chance variables $\mathcal{C}' = \mathcal{C} \cup \{W_1, \ldots, W_q, O_1, \ldots, O_q\}$, where $W_1, \ldots, W_q, O_1, \ldots O_q$ are binary variables. Furthermore, the subgraph of $\mathcal{G}'$ obtained by considering only nodes in $\mathcal{C}$ and $\mathcal{D}$ is identical to $\mathcal{G}$ with the chance variables $W_1, \ldots, W_q$ replacing the value variables $V_1, \ldots, V_q$, respectively. Also, the variables $O_1, \ldots, O_q$ are arranged in a chain such that W_1 is the parent of O_1, W_2 and O_1 are the parents of O_2 and so forth, as in Figure 2(b). Each variable W_i, for $i = 1, \ldots, q$, is associated to a probability distribution $P(W_i|\mathtt{Pa}(V_i))$ such that $P(w_i^{(1)}|\mathtt{Pa}(V_i)) = (U(\mathtt{Pa}(V_i)) - \underline{U})/(\overline{U} - \underline{U})$. Each variable O_i, $i = 1, \ldots, q$, is associated to a probability distribution $P(O_i|O_{i-1}, W_i)$ (we assume $O_0 = \emptyset$) such that $P(o_i^{(1)}|o_{i-1}^{(1)}, w_i^{(1)}) = 1$ and $P(o_i^{(1)}|o_{i-1}^{(1)}, w_i^{(2)}) = (i-1)/i$ and $P(o_i^{(1)}|o_{i-1}^{(2)}, w_i^{(1)}) = 1/i$ and $P(o_i^{(1)}|o_{i-1}^{(2)}, w_i^{(2)}) = 0$ ($P(o_1^{(1)}|w_1^{(1)}) = 1$ and $P(o_1^{(1)}|w_1^{(2)}) = 0$). Finally, the value node V, with O_q as sole parent, is associated to a utility function $U(O_q)$

[1] Any binary tree decomposition can be transformed to meet this requirement by repeatedly selecting a node i associated to a superset of the parents of a value variable V_i not meeting the requirement, and then adding two nodes j and k such that the children of i become children of j, and k is a child of i; the node j is associated to the set of variables associated to i, while k is associated to $\mathtt{Pa}(V_i)$. Note that the treewidth is unaltered by these operations.

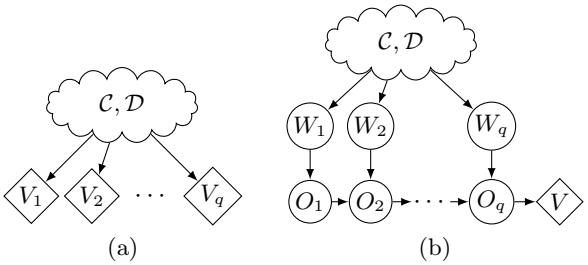


Figure 2: (a) Influence diagram with multiple value variables. (b) Its equivalent influence diagram containing a single variable as used in the proof of Theorem 1.

such that $U(o_q^{(1)}) = q\overline{U}$ and $U(o_q^{(2)}) = q\underline{U}$.

To show that the treewidth of $\mathcal{I}'$ is not greater than the treewidth of $\mathcal{I}$ by more than three, we build a tree decomposition $\mathcal{T}'$ for $\mathcal{I}'$, based on the tree decomposition $\mathcal{T}$, as follows. First, we take each node ℓ_i of $\mathcal{T}$ and include W_i, O_i and O_{i-1} in $\mathcal{X}_i$ (O_{i-1} is included for $i > 1$), which is enough to cover their families and to satisfy the family preserving property. Note that at this stage $\mathcal{T}'$ does not satisfy the running intersection property, because O_{i-1} appears in both ℓ_{i-1} and ℓ_i but not necessarily in every (set associated to a) node in between. Finally, we walk around the tree $\mathcal{T}'$ in a Euler tour tree traversal where each edge is visited exactly twice, and we include O_{i-1} in each node of $\mathcal{T}'$ that appears between ℓ_{i-1} and ℓ_i during the walk.[2] By doing so we guarantee that the running intersection property is also satisfied. Since the Euler tour tree traversal visits each leaf once and each internal node at most three times, the procedure inserts three new variables in the sets associated to $\ell_1, \ldots, \ell_q$ and at most three variables O_i in the sets associated to non-leaf nodes, and therefore does not increase the treewidth of the decomposition by more than three.

It remains to show that the diagrams are equivalent with respect to the expected utility of strategies. Let $\mathcal{S}$ be a strategy for $\mathcal{I}$. Then $\mathcal{S}$ is also a strategy for $\mathcal{I}'$ (because the two diagrams share the same decision variables and graph over $\mathcal{C},\mathcal{D}$). First, we need to show that for $i = 1, \ldots, q$ it follows that

$$P(o_i^{(1)}|\mathcal{C},\mathcal{D}) = \frac{1}{i}\left(\sum_{j=1}^{i} P(w_j^{(1)}|\mathcal{C},\mathcal{D})\right),$$

which we do by induction in i. The basis ($i = 1$) follows trivially, because $P(o_1^{(1)}|w_1^{(1)}) = 1$ by definition, so according to the graph structure we have that $P(o_1^{(1)}|\mathcal{C},\mathcal{D}) = P(w_1^{(1)}|\mathcal{C},\mathcal{D})$. Now assume by hypothesis of the induction that the above result is valid for every $1 \le i \le k < q$. Then $P(o_{k+1}^{(1)}|\mathcal{C},\mathcal{D})$

$$\begin{aligned}
&= P(o_{k+1}^{(1)}|o_k^{(1)}, w_{k+1}^{(1)})P(o_k^{(1)}|\mathcal{C},\mathcal{D})P(w_{k+1}^{(1)}|\mathcal{C},\mathcal{D}) \\
&\quad + P(o_{k+1}^{(1)}|o_k^{(1)}, w_{k+1}^{(2)})P(o_k^{(1)}|\mathcal{C},\mathcal{D})P(w_{k+1}^{(2)}|\mathcal{C},\mathcal{D}) \\
&\quad + P(o_{k+1}^{(1)}|o_k^{(2)}, w_{k+1}^{(1)})P(o_k^{(2)}|\mathcal{C},\mathcal{D})P(w_{k+1}^{(1)}|\mathcal{C},\mathcal{D}) \\
&\quad + P(o_{k+1}^{(1)}|o_k^{(2)}, w_{k+1}^{(2)})P(o_k^{(2)}|\mathcal{C},\mathcal{D})P(w_{k+1}^{(2)}|\mathcal{C},\mathcal{D}) \\
&= \left(\frac{1}{k+1} + \frac{k}{k+1}\right) P(o_k^{(1)}|\mathcal{C},\mathcal{D})P(w_{k+1}^{(1)}|\mathcal{C},\mathcal{D}) + \\
&\quad \left(\frac{k}{k+1}\right) P(o_k^{(1)}|\mathcal{C},\mathcal{D})P(w_{k+1}^{(2)}|\mathcal{C},\mathcal{D}) + \\
&\quad \left(\frac{1}{k+1}\right) P(o_k^{(2)}|\mathcal{C},\mathcal{D})P(w_{k+1}^{(1)}|\mathcal{C},\mathcal{D}) \\
&= \left(\frac{k}{k+1}\right) P(o_k^{(1)}|\mathcal{C},\mathcal{D}) + \frac{1}{k+1} P(w_{k+1}^{(1)}|\mathcal{C},\mathcal{D}) \\
&= \frac{k}{(k+1)}\left(\frac{1}{k}\sum_{j=1}^{k} P(w_j^{(1)}|\mathcal{C},\mathcal{D})\right) + \frac{1}{k+1} P(w_{k+1}^{(1)}|\mathcal{C},\mathcal{D}) \\
&= \frac{1}{k+1}\left(\sum_{j=1}^{k+1} P(w_j^{(1)}|\mathcal{C},\mathcal{D})\right),
\end{aligned}$$

which shows that the result holds also for $i = k + 1$.

We can now show that for any strategy $\mathcal{S}$ the associated expected utility $\mathrm{E}'_{\mathcal{S}}$ in $\mathcal{I}'$ equals the associated expected utility $\mathrm{E}_{\mathcal{S}}$ in $\mathcal{I}$. Let $\mathcal{C}'' = \mathcal{C}' \setminus \{O_q\}$. By definition, $\mathrm{E}'_{\mathcal{S}} = \sum_{\mathcal{C}',\mathcal{D}} P_{\mathcal{S}}(\mathcal{C}',\mathcal{D})U(\mathcal{C}',\mathcal{D}) = \sum_{\mathcal{C}',\mathcal{D}} P_{\mathcal{S}}(\mathcal{C}',\mathcal{D})U(O_q)$, which is equal to

$$\begin{aligned}
&\sum_{\mathcal{C}'',\mathcal{D}} P_{\mathcal{S}}(\mathcal{C}'',\mathcal{D})\left(q\overline{U}P(o_q^{(1)}|\mathcal{C}'',\mathcal{D}) + q\underline{U}P(o_q^{(2)}|\mathcal{C}'',\mathcal{D})\right) \\
&= q\underline{U} + \sum_{\mathcal{C}'',\mathcal{D}} P_{\mathcal{S}}(\mathcal{C}'',\mathcal{D})q(\overline{U} - \underline{U})P(o_q^{(1)}|\mathcal{C}'',\mathcal{D}) \\
&= q\underline{U} + q(\overline{U} - \underline{U})\sum_{\mathcal{C},\mathcal{D}} P_{\mathcal{S}}(\mathcal{C},\mathcal{D})P(o_q^{(1)}|\mathcal{C},\mathcal{D}) \\
&= q\underline{U} + q(\overline{U} - \underline{U})\sum_{\mathcal{C},\mathcal{D}} P_{\mathcal{S}}(\mathcal{C},\mathcal{D})\frac{1}{q}\sum_{i=1}^{q} P(w_i^{(1)}|\mathcal{C},\mathcal{D}) \\
&= q\underline{U} + q\frac{\overline{U} - \underline{U}}{q}\sum_{\mathcal{C},\mathcal{D}} P_{\mathcal{S}}(\mathcal{C},\mathcal{D})\sum_{i=1}^{q} P(w_i^{(1)}|\mathtt{Pa}(V_i)) \\
&= q\underline{U} + (\overline{U} - \underline{U})\sum_{\mathcal{C},\mathcal{D}} P_{\mathcal{S}}(\mathcal{C},\mathcal{D})\sum_{j=1}^{q}\frac{U(\mathtt{Pa}(V_i)) - \underline{U}}{\overline{U} - \underline{U}} \\
&= q\underline{U} + \frac{\overline{U} - \underline{U}}{\overline{U} - \underline{U}}\sum_{\mathcal{C},\mathcal{D}} P_{\mathcal{S}}(\mathcal{C},\mathcal{D})\sum_{i=1}^{q}[U(\mathtt{Pa}(V_i)) - \underline{U}] \\
&= q\underline{U} - q\underline{U} + \sum_{\mathcal{C},\mathcal{D}} P_{\mathcal{S}}(\mathcal{C},\mathcal{D})\sum_{i=1}^{q} U(\mathtt{Pa}(V_i)),
\end{aligned}$$

[2] An Euler tour tree traversal of $\mathcal{T}$ rooted at r is a list of $2m-1$ symbols produced by calling $ET(r)$, where $ET(i)$ is a recursive function that takes a node i with left children j and right children k (if they exist) and prints out i, calls $ET(j)$ (if j exists), prints out i again, calls $ET(k)$ (if k exists), and then prints out i once more.

which is equal to $\sum_{\mathcal{C},\mathcal{D}} P_{\mathcal{S}}(\mathcal{C},\mathcal{D})U(\mathcal{C},\mathcal{D}) = \mathrm{E}_{\mathcal{S}}$. □

Cooper [3] showed that in any influence diagram containing a single value variable the (single) utility function can be re-normalized to take values on the interval $[0, 1]$ without affecting the optimality of the strategies. Together with the above result, this allows us, without loss of generality, to focus exclusively on influence diagrams containing a single value variable taking values on the interval $[0, 1]$.

Under the usual assumptions of complexity theory, when a problem is NP-hard to solve the best available options are (i) trying to devise an algorithm that runs efficiently on many instances but has exponential worst-case complexity, or (ii) trying to develop an approximation algorithm that for all instances provides in polynomial time a solution that is provably within a certain range of the optimal solution. We have investigated option (i) in a recent work [11, 12], and showed empirically that a sophisticate variable elimination algorithm can, in spite of its exponential worst-case complexity, solve a large number of problems in feasible time. In the remainder, we study the theoretical feasibility of option (ii). Given $0 < \epsilon < 1$, we say that a procedure is an *ϵ-approximation* algorithm for the strategy selection problem if for any influence diagram it finds a strategy $\mathcal{S}$ such that $\mathrm{MEU} \leq (1+\epsilon)\,\mathrm{E}_{\mathcal{S}}$. The algorithm is said to be a *fully polynomial-time approximation scheme* if it runs in time polynomial in the input and in $1/\epsilon$ for any given ϵ. The following result implies that without assuming a bounded number of states per variable approximating the strategy selection problem in polynomial time by virtually any reasonable factor is NP-hard, and option (ii) is most likely unfeasible.

Theorem 2 ([12]). *Given a singly connected influence diagram of bounded treewidth, for any $1 < 1 + \epsilon < 2^{\theta}$, there is no polynomial-time ϵ-approximation algorithm unless P=NP, where θ is the number of numerical parameters (i.e., probabilities and rewards) needed to specify the problem.*

The result is obtained by a reduction from the SAT problem, similarly to the proof of inapproximability of MAP in Bayesian networks given by Park and Darwiche [16]. According to the above result, algorithms like SPU [10] or mini-bucket elimination [5], which find a strategy in polynomial time, cannot guarantee that the worst-case ratio between the expected utility of that strategy and the maximum expected utility is greater than $2^{-\theta}$, even if the input is constrained to influence diagrams of bounded treewidth. Since the factor $2^{-\theta}$ quickly approaches zero as the size of the problem increases, efficient algorithms eventually produce very poor solutions. As we show in the rest of the paper, this is no longer true if we assume that both the treewidth of the diagram and the cardinality of the variables are bounded. In this case, we show constructively the existence of a fully polynomial-time approximation scheme.

3 FINDING PROVABLY GOOD STRATEGIES

Our first step in showing the existence of a fully polynomial-time approximation scheme for the strategy selection problem is to devise an algorithm that obtains provably good strategies, that is, an ϵ-approximation scheme which returns, for any $\epsilon > 0$, a strategy $\mathcal{S}$ satisfying $\mathrm{MEU} \leq (1+\epsilon)\,\mathrm{E}_{\mathcal{S}}$. The algorithm (scheme) we devise is a generalization of the MPU algorithm [11, 12] to the case of (provably good) approximate inference in influence diagrams, and it consists in propagating sets of probability potentials over a tree decomposition. The idea behind the propagation of sets of potentials is that the expected utility of each strategy can be computed by propagating a (single) potential over the same tree decomposition, and so the propagated sets account for the simultaneous evaluation of many different strategies. To be efficient, some of these computations are halted early on, so that not every strategy has its expected value computed. The difficulty is in guaranteeing that within the evaluated strategies (i.e., those whose expected values were actually computed) there is at least one which achieves the desired approximation factor $1 + \epsilon$.

Let α be a real value greater than one, and $\mathcal{K}$ be a set of nonnegative functions (called potentials) over a set of variables $\mathcal{X}$. We say that $\mathcal{K}' \subseteq \mathcal{K}$ is a *α-covering* of $\mathcal{K}$ if for every potential $P(\mathcal{X})$ in $\mathcal{K}$ there is $Q(\mathcal{X})$ in $\mathcal{K}'$ such that $P(\mathcal{X}) \leq \alpha Q(\mathcal{X})$. We also define the potential $1(\mathcal{X})$ which returns one for every assignment to the variables in $\mathcal{X}$.

For any decision variable D, let $\mathcal{P}_D$ denote the set of all pure policies for D. Hence, $\mathcal{P}_D$ is a set containing $|D|^{\gamma}$ distributions $P(D|\mathtt{Pa}(D))$, where $|D|$ is the number of values D can assume and $\gamma = \prod_{X \in \mathtt{Pa}(D)} |X|$ is the number of assignments to its parents (i.e., the number of different combinations of values the parent variables can jointly assume).

The procedure in Algorithm 1 takes an influence diagram (which we assume contains a single value variable taking values on $[0, 1]$), a tree decomposition of the diagram, and a nonnegative value ϵ, and computes an expected utility $E_{\mathcal{S}}$ induced by some strategy $\mathcal{S}$ such that $\mathrm{MEU} \leq (1 + \epsilon)E_{\mathcal{S}}$, that is, it is an ϵ-approximation for computing the maximum expected utility. To keep things simple for now, let us assume

Algorithm 1 Finding Provably Good Strategies

Require: A positive number ϵ, an influence diagram $\mathcal{I}$ and a tree decomposition $\mathcal{T}$ with m nodes
Ensure: The value E satisfies MEU $\leq (1+\epsilon)\,\mathrm{E}$
1: // INITIALIZATION
2: let $\alpha = 1 + \epsilon/(2m)$
3: let $\mathcal{K}_1, \ldots, \mathcal{K}_m$ be singletons containing the potentials $1(\mathcal{X}_1), \ldots, 1(\mathcal{X}_m)$, respectively
4: **for each** chance variable C **do**
5: find a node i in $\mathcal{T}$ such that $\mathtt{Fa}(C) \subseteq \mathcal{X}_i$
6: set $\mathcal{K}_i \leftarrow \textsc{Combine}(\mathcal{K}_i, \{P(C|\mathtt{Pa}(C))\})$
7: **end for**
8: **for each** decision variable D **do**
9: find a node i in $\mathcal{T}$ such that $\mathtt{Fa}(D) \subseteq \mathcal{X}_i$
10: set $\mathcal{K}_i \leftarrow \textsc{Combine}(\mathcal{K}_i, \mathcal{P}_D)$
11: **end for**
12: find a node i in $\mathcal{T}$ such that $\mathtt{Pa}(V) \subseteq \mathcal{X}_i$
13: set $\mathcal{K}_i \leftarrow \textsc{Combine}(\mathcal{K}_i, \{U(\mathtt{Pa}(V))\})$
14: // PROPAGATION
15: select a node r as the root of $\mathcal{T}$
16: label all nodes as inactive
17: **while** there is an inactive node i **do**
18: select an inactive node i whose children are all active
19: set $\mathcal{A}_i \leftarrow \textsc{Combine}(\mathcal{K}_i, \mathcal{C}_j : j \in \mathtt{Ch}(i))$
20: set $\mathcal{B}_i \leftarrow \textsc{SumOut}(\mathcal{A}_i, \mathcal{X}_i \setminus \mathcal{X}_{\mathtt{Pa}(i)})$
21: set $\mathcal{C}_i \leftarrow \textsc{Covering}(\mathcal{B}_i, \alpha)$
22: label i as active
23: **end while**
24: let $\mathrm{E} = \max\{\mu : \mu \in \mathcal{C}_r\}$

this is our goal. We will see later on how to modify the algorithm to also provide the strategy $\mathcal{S}$, not only its expected value, and thus devise an ϵ-approximation algorithm for the strategy selection problem.

The algorithm is similar in spirit to the computation of marginal queries in Bayesian networks by junction trees [8], but differs radically in that it stores and propagates sets of probability potentials, instead of storing and propagating single probability potentials. Like junction-tree algorithms for Bayesian networks, the algorithm contains an initialization step, where the sets of probability potentials are assigned to nodes of the tree decomposition, and a propagation step, where messages are sent from the leaves towards the root node. The propagation finishes when the root receives a message from every child, and obtains a set of values $\mathcal{C}_r$ (since $\mathtt{Pa}(r) = \emptyset$, the set $\mathcal{B}_r$ marginalizes out all variables from each potential $P(\mathcal{X}_r)$ in $\mathcal{B}_r$, hence producing real numbers).

The operations Combine and SumOut are analogous to the multiplication and marginalization of probability potentials but operate over sets, that is, the former returns the set of all potentials obtained by pairwise multiplication of the potentials in the sets given as arguments, whereas the latter returns the set obtained by summing out the variables in the second argument from every potential in the first argument. More formally, given a list of sets $\mathcal{K}_1, \ldots, \mathcal{K}_n$ containing potentials over the sets of variables $\mathcal{Y}_1, \ldots, \mathcal{Y}_n$, respectively, we define the Combine operation as $\textsc{Combine}(\mathcal{K}_1, \ldots, \mathcal{K}_n) = \{P(\mathcal{Y}_1) \cdots P(\mathcal{Y}_n) : P(\mathcal{Y}_i) \in \mathcal{K}_i, i = 1, \ldots, n\}$; given a set $\mathcal{K}$ of potentials over the set of variables $\mathcal{Y}$ and a subset $\mathcal{Z} \subseteq \mathcal{Y}$, the SumOut operation is given by $\textsc{SumOut}(\mathcal{K}, \mathcal{Z}) = \{\sum_{\mathcal{Z}} P(\mathcal{Y}) : P(\mathcal{Y}) \in \mathcal{K}\}$. Finally, the Covering operation returns an α-covering for the argument, where $\alpha = 1 + \epsilon/(2m)$ is set according to the approximation factor ϵ and the number of nodes in the tree m. For the moment, let us ignore how α-coverings are actually obtained. We shall get back to this point later on.

The algorithm begins by assigning each probability distribution associated to a chance variable to a node of the tree decomposition. If two or more such functions are assigned to the same node i, the result is a singleton $\mathcal{K}_i$ containing their multiplication. The algorithm then assigns policies to nodes in much a similar way, except that the sets associated to nodes which have been chosen in the second loop are no longer singletons. Finally, the utility function is associated to a node of the tree. For example, if the tree decomposition contained only a single node with all decision and chance variables associated to it, the result of the initialization step would be a single set $\mathcal{K}_1$ containing all joint probability distributions $P_{\mathcal{S}}(\mathcal{C}, \mathcal{D})$ induced by strategies $\mathcal{S}$. On the other hand, if for each (decision, chance and value) variable X there were exactly one node i in the tree such that $\mathtt{Fa}(X) \subseteq \mathcal{X}_i$, then after the initialization step the set $\mathcal{K}_i$, for $i = 1, \ldots, m$, would be equal to the singleton $\{P(X|\mathtt{Pa}(X))\}$, if the node i were associated to the family of a chance variable X, to the singleton $\{U(\mathtt{Pa}(X))\}$, if i were the node associated to the parents of the value variable V, or to the set of policies $\mathcal{P}_X$, if i was associated to the family of a decision variable X.

The propagation step starts by rooting the tree decomposition $\mathcal{T}$ at an arbitrary node r. This allows us to organize the neighbors of a node in the tree as the parent (i.e., the one closer to the root) and the children (the ones farther from the root), and defines a direction for the propagation of messages, which consists in selecting a node i satisfying the condition (which initially only leaves do) and computing the corresponding sets of potentials $\mathcal{A}_i$, $\mathcal{B}_i$ and $\mathcal{C}_i$. The first two are set analogous to the potentials produced during a junction-tree propagation in a Bayesian network. The message set $\mathcal{C}_i$ is obtained by removing elements from

$\mathcal{B}_i$ in a way that satisfies the α-covering condition (we will get back to this). Once all nodes have been processed (including the root), the algorithm produces a solution E by seeking the highest value in the set $\mathcal{C}_r$. To show that this value is indeed the expected utility of a feasible strategy $\mathcal{S}$ such that MEU $\leq (1+\epsilon)\,\mathrm{E}_{\mathcal{S}}$, we need to introduce some additional notation.

For any node i of the tree decomposition, let $\mathcal{T}[i]$ be the subtree rooted at i, that is, the graph obtained by removing all nodes which are not descendants of i or i itself. Let also $\mathcal{X} = \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_m = \mathcal{C} \cup \mathcal{D}$ denote all variables in the tree. We denote by $\mathcal{X}[i]$ the set of all variables associated to the nodes in $\mathcal{T}[i]$, that is, $\mathcal{X}[i] = \bigcup_{j \in \mathcal{T}[i]} \mathcal{X}_j$ (hence $\mathcal{X}_i \subseteq \mathcal{X}[i]$). We define a function $\sigma : \mathcal{X} \to \{1, \ldots, m\}$ that returns for each variable X the node i to which its associated (probability, policy or utility) potential was assigned in the initialization step. Given a node i, the function $U(\mathtt{Pa}(V); i)$ returns the utility function $U(\mathtt{Pa}(V))$ if $\sigma(V) = i$ and one otherwise. Finally, let $\mathcal{Y}_i = \mathcal{X}_i \cap \mathcal{X}_{\mathtt{Pa}(i)}$ be the separator set of i and $\mathtt{Pa}(i)$.

Lemma 3. *The value* E *computed by the procedure in Algorithm 1 is the expected utility* $\mathrm{E}_{\mathcal{S}}$ *associated to some valid strategy* $\mathcal{S}$*.*

Proof. First we show by induction in the nodes from the leaves toward the root that for any node i any potential $P(\mathcal{Y}_i)$ in $\mathcal{B}_i$ or $\mathcal{C}_i$ satisfies

$$P(\mathcal{Y}_i) = \sum_{\mathcal{X}[i] \setminus \mathcal{X}_{\mathtt{Pa}(i)}} \prod_{C : \sigma(C) \in \mathcal{T}[i]} P(C|\mathtt{Pa}(C)) \prod_{D : \sigma(D) \in \mathcal{T}[i]} P(D|\mathtt{Pa}(D)) \prod_{j \in \mathcal{T}[i]} U(\mathtt{Pa}(V); j)$$

for some partial strategy $\{P(D|\mathtt{Pa}(D)) : D \in \mathcal{D}, \sigma(D) \in \mathcal{T}[i]\}$. Consider a leaf node i. Then the induction hypothesis is trivially satisfied by applying the definitions of the Combine and SumOut operations, and noting that $\mathcal{T}[i]$ contains only the node i. Now consider an internal node i and assume that for every child j of i the induction hypothesis holds. Then for any $P(\mathcal{Y}_i)$ in $\mathcal{B}_i$ or in $\mathcal{C}_i$ it follows that $P(\mathcal{Y}_i)$

$$= \sum_{\mathcal{X}_i \setminus \mathcal{X}_{\mathtt{Pa}(i)}} \prod_{C : \sigma(C) = i} P(C|\mathtt{Pa}(C)) \prod_{D : \sigma(D) = i} P(D|\mathtt{Pa}(D)) \; U(\mathtt{Pa}(V); i) \prod_{j \in \mathtt{Ch}(i)} P(\mathcal{Y}_j)$$

$$= \sum_{\mathcal{X}[i] \setminus \mathcal{X}_{\mathtt{Pa}(i)}} \prod_{C : \sigma(C) \in \mathcal{T}[i]} P(C|\mathtt{Pa}(C)) \prod_{D : \sigma(D) \in \mathcal{T}[i]} P(D|\mathtt{Pa}(D)) \prod_{j \in \mathcal{T}[i]} U(\mathtt{Pa}(V); j)\,,$$

which satisfies the induction hypothesis. The result of the lemma is thus obtained by applying the induction result to the root node r. For any $\mu \in \mathcal{C}_r$ we have that

$$\mu = \sum_{\mathcal{X}} \prod_{C} P(C|\mathtt{Pa}(C)) \prod_{D} P(D|\mathtt{Pa}(D)) U(\mathtt{Pa}(V))$$

for some strategy $S = \{P(D|\mathtt{Pa}(D))\}$. □

Theorem 4. *The value* E *satisfies* MEU $\leq (1+\epsilon)\,\mathrm{E}$.

Proof. Let $S^* = \{P^*(D|\mathtt{Pa}(D)) : D \in \mathcal{D}\}$ denote an optimal strategy. We first show by induction from the leaves toward the root that for any node i there is a $P(\mathcal{Y}_i) \in \mathcal{C}_i$ such that $P^*(\mathcal{Y}_i) \leq \alpha^{s_i} P(\mathcal{Y}_i)$, where

$$P^*(\mathcal{Y}_i) = \sum_{\mathcal{X}[i] \setminus \mathcal{X}_{\mathtt{Pa}(i)}} \prod_{C : \sigma(C) \in \mathcal{T}[i]} P(C|\mathtt{Pa}(C)) \prod_{D : \sigma(D) \in \mathcal{T}[i]} P^*(D|\mathtt{Pa}(D)) \prod_{j \in \mathcal{T}[i]} U(\mathtt{Pa}(V); j)\,,$$

and s_i is the number of nodes in $\mathcal{T}[i]$. Consider a leaf node i. Then the induction hypothesis holds since $P^*(\mathcal{Y}_i) \in \mathcal{B}_i$ by design and by definition of α-covering there is $P(\mathcal{Y}_i) \in \mathcal{C}_i$ such that $P^*(\mathcal{Y}_i) \leq \alpha P(\mathcal{Y}_i)$. Assume the induction holds for all children j of a node i. By design, there is $P(\mathcal{Y}_i) \in \mathcal{B}_i$ such that

$$P(\mathcal{Y}_i) = \sum_{\mathcal{X}_i \setminus \mathcal{X}_{\mathtt{Pa}(i)}} \prod_{C : \sigma(C) = i} P(C|\mathtt{Pa}(C)) \prod_{D : \sigma(D) = i} P^*(D|\mathtt{Pa}(D)) U(\mathtt{Pa}(V); i) \prod_{j \in \mathtt{Ch}(i)} P(\mathcal{Y}_j)\,,$$

and by using the inductive hypothesis it follows that $P^*(\mathcal{Y}_i) \leq \alpha^{\sum_{j \in \mathtt{Ch}(i)} s_j} P(\mathcal{Y}_i)$. And since $\mathcal{C}_i$ is an α-covering of $\mathcal{B}_i$, there is $Q(\mathcal{Y}_i) \in \mathcal{C}_i$ such that $P(\mathcal{Y}_i) \leq \alpha Q(\mathcal{Y}_i)$, and thus $P^*(\mathcal{Y}_i) \leq \alpha^{1+\sum_{j \in \mathtt{Ch}(i)} s_j} Q(\mathcal{Y}_i) = \alpha^{s_i} Q(\mathcal{Y}_i)$. Since by Lemma 3, every $\mu \in \mathcal{C}_r$ corresponds to the expected utility of some strategy $\mathcal{S}$, it follows from the induction result on r that there is $\mathrm{E}_{\mathcal{S}} \in \mathcal{C}_r$ such that MEU $\leq \alpha^m\,\mathrm{E}_{\mathcal{S}}$, and since E maximizes over $\mu \in \mathcal{C}_r$, we have that MEU $\leq \alpha^m\,\mathrm{E} = (1 + \epsilon/(2m))^m\,\mathrm{E}$. Finally, it follows from the inequality $(1+2x) \geq (1+x/k)^k$, valid for every positive real $x \leq 1$ and positive integer k, that MEU $\leq (1+\epsilon)\,\mathrm{E}$. □

Recall from Section 2 that for any influence diagram of bounded treewidth we can obtain a binary tree decomposition of minimum treewidth in linear time. Assume, without loss of generality, that such a minimum treewidth binary tree decomposition is given as input to the approximation algorithm, and let ω be the treewidth of the influence diagram given as input and κ be the maximum number of values a decision or chance variable can assume. The complexity of the algorithm is bounded by the complexity of computing a potential $P(\mathcal{Y}_i)$ in a set $\mathcal{B}_i$ times the cardinality of the

largest set $\mathcal{B}_i$ plus the complexity of the initialization step and the complexity of the COVERING operation. Computing a $P(\mathcal{Y}_i)$ requires $2\prod_{X\in\mathcal{X}_i}|X|$ multiplications and $\prod_{X\in\mathcal{X}_i\setminus X_{\mathtt{Pa}(i)}}|X|$ additions, and hence takes $O(\kappa^\omega)$ time, which is polynomial in the cardinality of the (chance and decision) variables (since ω is assumed to be bounded by a constant). The complexity of the initialization step is bounded by the complexity of combining the sets of policies $\mathcal{P}_D$ with the sets $\mathcal{K}_i$ for each decision variable. Each decision variable has at most κ^{κ^ω} policies, which makes this step exponential in κ. Although it is possible to transform the diagram so that the complexity of the initialization step becomes polynomially bounded in κ [12, Prop. 7], we refrain from doing so here because it would make the algorithm more complicated, and it would not change the worst-case running time, which we know from Theorem 2 that cannot be polynomial in κ.

Regarding the cardinality of the sets $\mathcal{C}_i$ and the complexity of the COVERING operations, in principle, we would like to be able to implement COVERING in way that it takes polynomial time in its argument and provides an α-covering that is as small as possible. The former requirement is easily met by sequentially inspecting an element $P(\mathcal{Y}_i)$ in $\mathcal{B}_i$ and inserting it into $\mathcal{C}_i$ only if there is no other $Q(\mathcal{Y}_i)$ in $\mathcal{B}_i$ such that $P(\mathcal{Y}_i)\leq\alpha Q(\mathcal{Y}_i)$. This procedure takes time polynomial in the cardinality of $\mathcal{B}_i$ and in $|\mathcal{Y}_i|$, but does not guarantee that the size of the obtained set is bounded, and in the worst case we might simply output $\mathcal{C}_i=\mathcal{B}_i$ at every node i, which would cause $\mathcal{C}_r$ to contain as many as $\prod_{i=1}^m|\mathcal{K}|_i$ or $O(\kappa^{\kappa^{m\omega}})$ elements. As we show in the next section, there is a better way of finding α-coverings (in polynomial time) that guarantees that the cardinality of the sets $\mathcal{C}_i$ remains bounded in the (size of the influence diagram given as) input (measured in bits and using a reasonable encoding) if the cardinality of any variable is bounded.

Before introducing the fully polynomial-time approximation scheme, there is a minor detail we have postponed in the discussion, which is how to modify the algorithm to provide not only the expected value but also a strategy that obtains that value. This can be easily implemented by associating to any potential generated during the algorithm the policies that were used either directly or indirectly to produce it. More specifically, let δ be a dictionary, which is initialized with $\delta[1(\mathcal{X}_i)]\leftarrow\{\}$ for $i=1,\ldots,m$, $\delta[P(C|\mathtt{Pa}(C))]\leftarrow\{\}$ for every C in $\mathcal{C}$, $\delta[U(\mathtt{Pa}(V))]\leftarrow\{\}$, and $\delta[P(D|\mathtt{Pa}(D))]\leftarrow\{P(D|\mathtt{Pa}(D))\}$ for every D in $\mathcal{D}$ and $P(D|\mathtt{Pa}(D))$ in $\mathcal{P}_D$. We redefine the operations COMBINE and SUMOUT to update δ as the computations are done as follows. Let $\mathcal{K}$ be the output of COMBINE$(\mathcal{K}_1,\ldots,\mathcal{K}_n)$. Then for every $P(\mathcal{Y}_1)\cdots P(\mathcal{Y}_n)\in\mathcal{K}$, where $P(\mathcal{Y}_1)\in\mathcal{K}_1,\ldots,P(\mathcal{Y}_n)\in\mathcal{K}_n$, we assign $\delta[P(\mathcal{Y}_1)\cdots P(\mathcal{Y}_n)]\leftarrow\delta[P(\mathcal{Y}_1)]\cup\cdots\cup\delta[P(\mathcal{Y}_n)]$. Likewise, let $\mathcal{K}$ be the output of SUMOUT$(\mathcal{K}',\mathcal{Z})$. Then $\delta[\sum_{\mathcal{Z}}P(\mathcal{Y})]\leftarrow\delta[P(\mathcal{Y})]$ for every $\sum_{\mathcal{Z}}P(\mathcal{Y})$ in $\mathcal{K}$, where $P(\mathcal{Y})\in\mathcal{K}'$. The COVERING operation does not need modification since it returns a subset of its argument. Finally, we obtain a strategy $\mathcal{S}$ such that $\mathrm{E}_{\mathcal{S}}=\mathrm{E}$ from $\delta[\mathrm{E}]$. Note that these additional operations do not change the asymptotical running time complexity, and can be efficiently implemented using pointers to the original functions, incurring a very small increase in the space complexity.

4 A FULLY POLYNOMIAL-TIME APPROXIMATION SCHEME

When the maximum cardinality of a variable κ is assumed bounded, the complexity of computing each potential $P(\mathcal{Y}_i)$ in the procedure in Algorithm 1 as well as the cardinalities of the sets $\mathcal{K}_i$ are bounded by a constant. Hence, the only difficulty one needs to overcome in order to devise a fully polynomial-time approximation scheme out of that procedure is to guarantee that the operation COVERING returns sets $\mathcal{C}_i$ whose cardinality is polynomially bounded by the input size, where the latter is defined as the number of bits needed to encode all numerical parameters (i.e., probabilities and utilities) as well as the variables and the underlying graph of the diagram. For definiteness, we assume the numerical parameters are specified as rational numbers in a reasonable way. The procedure in Algorithm 2 partitions the space $\mathcal{Y}$ over which the potentials in the input set $\mathcal{K}$ are specified in hyperrectangles such that the ratio of any two potentials falling in the same rectangle is at most α. Thus, we can provide an α-covering of $\mathcal{K}$ by letting $\mathcal{K}'$ be a set obtained by selecting exactly one element from each non-empty rectangle.[3] Remarkably, we show that the number of elements in $\mathcal{K}'$ is a polynomial function of the size of the influence diagram (in bits).

The next result, which is inspired by a similar result by Papadimitriou and Yannakakis [15], relates the cardinality of the output of the COVERING procedure in Algorithm 2 with the size in bits of the input set $\mathcal{K}$.

Lemma 5. *Let $\mathcal{K}$ be a set of potentials $P(\mathcal{Y})$ whose range is contained in $[0,1]$. Then the set $\mathcal{K}'$ obtained by the procedure in Algorithm 2 is an α-covering of $\mathcal{K}$ with at most $(1-\lfloor\log_\alpha t\rfloor)^\eta$ elements, where t is the smallest (strictly) positive number in the range of a potential in $\mathcal{K}$ and $\eta=\prod_{X\in\mathcal{Y}}|X|$ is the number of assignments to $\mathcal{Y}$.*

[3]The notation $\lfloor\log_\alpha P(\mathcal{Y})\rfloor$ denotes a function $F(\mathcal{Y})$ such that $F(\boldsymbol{y})=\lfloor\log_\alpha P(\boldsymbol{y})\rfloor$ if $P(\boldsymbol{y})\neq 0$ and $F(\boldsymbol{y})=0$ otherwise, where $\boldsymbol{y}$ is an assignment to $\mathcal{Y}$.

Algorithm 2 Finding A Small α-Covering

Require: A set $\mathcal{K}$ of potentials over a set of variables $\mathcal{Y}$, a value $\alpha > 1$
Ensure: $\mathcal{K}'$ is an α-covering of $\mathcal{K}$
1: let $\mathcal{K}'$ and $\mathcal{L}$ initially be empty sets
2: **for each** $P(\mathcal{Y})$ in $\mathcal{K}$ **do**
3: **if** $\lfloor \log_\alpha P(\mathcal{Y}) \rfloor$ is not in $\mathcal{L}$ **then**
4: insert $\lfloor \log_\alpha P(\mathcal{Y}) \rfloor$ into $\mathcal{L}$
5: insert $P(\mathcal{Y})$ into $\mathcal{K}'$
6: **end if**
7: **end for**

Proof. To see that $\mathcal{K}'$ is an α-covering of $\mathcal{K}$, note that by design if a potential $P(\mathcal{Y}) \in \mathcal{K}$ is not in $\mathcal{K}'$ then the latter contains a potential $Q(\mathcal{Y}) \in \mathcal{K}$ such that $\lfloor \log_\alpha P(\mathcal{Y}) \rfloor = \lfloor \log_\alpha Q(\mathcal{Y}) \rfloor$, which implies $P(\mathcal{Y}) \leq \alpha Q(\mathcal{Y})$.

Regarding the cardinality of $\mathcal{K}'$, first note that there are $-\lfloor \log_\alpha t \rfloor$ (distinct) integers between t and one (the minus sign is because $t \leq 1$). Now consider a potential $P(\mathcal{Y}) \in \mathcal{K}$. By assumption, the range of $P(\mathcal{Y})$ is contained in $[0, 1]$, and for each assignment $\boldsymbol{y}$ to $\mathcal{Y}$ we have that either $P(\boldsymbol{y}) \geq t$ or $P(\boldsymbol{y}) = 0$. Hence, for each $\boldsymbol{y}$ there are only $(1 - \lfloor \log_\alpha t \rfloor)$ distinct values the number $\lfloor \log_\alpha P(\boldsymbol{y}) \rfloor$ can assume, and therefore only $(1 - \lfloor \log_\alpha t \rfloor)^\eta$ possibilities for $\lfloor \log_\alpha P(\mathcal{Y}) \rfloor$. □

The following result is an immediate consequence of the above result which shows that the cardinality of any set $\mathcal{C}_i$ is polylogarithmic in the smallest positive number being specified by a potential in $\mathcal{B}_i$.

Corollary 6. *For $i = 1, \ldots, m$, the set $\mathcal{C}_i$ contains $O([1 - \lfloor \log_\alpha t_i \rfloor]^{\kappa^\omega})$ elements, where t_i is the smallest (strictly) positive number in a potential in $\mathcal{B}_i$.*

We can now state the main result of this paper.

Theorem 7. *There is a fully polynomial-time approximation scheme for influence diagrams of bounded treewidth and bounded variable cardinality.*

Proof. Assume, without loss of generality, that the influence diagram given as input contains only one value variable taking values in $[0, 1]$. Let t denote the smallest (strictly) positive numerical parameter in the specification (i.e., the smallest nonzero probability or utility specified by the diagram), and let b denote the size of the diagram (in bits). Since the input probabilities and utilities are (by assumption) rational numbers, each positive input number is not smaller than 2^{-b} (otherwise we would need more than b bits to encode it). Any potential $P(\mathcal{Y})$ in a set $\mathcal{B}_i$ is obtained by multiplying and marginalizing the functions specified by the diagram, and hence any value $P(\boldsymbol{y})$ in $P(\mathcal{Y})$ is a polynomial in the numerical parameters in the input. Moreover, since each variable in the network is associated to a function over at most κ^{κ^ω} numbers, the polynomial has degree at most $O(n\kappa^{\kappa^\omega}) \leq O(n)$, where n is the number of variables. In particular, the smallest positive value t_i in a potential in $\mathcal{B}_i$, for $i = 1, \ldots, m$, is also a polynomial in the numerical parameters of the input of degree $O(n)$, and since these are either zero or some number greater than or equal to 2^{-b}, it follows that $t_i \geq 2^{-bO(n)}$. Thus, we have from Corollary 6 that the cardinality of any set $\mathcal{C}_i$ is $O([1 - \lfloor \log_\alpha 2^{-bO(n)} \rfloor]^{\kappa^\omega}) \leq O([bn/\ln(\alpha)]^{\kappa^\omega})$. But since $\alpha = 1 + \epsilon/2m$, we have from the inequality $\ln(1 + x) \geq x/(1 + x)$ valid for all $x > 0$ that

$$O\left(\left[\frac{bn}{\ln(\alpha)}\right]^{\kappa^\omega}\right) \leq O\left(\left[bn\frac{1+\frac{\epsilon}{2n}}{\frac{\epsilon}{2n}}\right]^{\kappa^\omega}\right) \leq O\left(\left[\frac{bn^2}{\epsilon}\right]^{\kappa^\omega}\right).$$

Hence, for $i = 1, \ldots, m$ the number of elements in any $\mathcal{C}_i$ is polynomial in b, n and $1/\epsilon$. □

4.1 CONCLUSION

Influence diagrams provide a very expressive language to describe decision problems under uncertainty, especially if no-forgetting and regularity are not required. Finding an optimal strategy for such problems is NP-hard even in diagrams of bounded treewidth and very simple structure (e.g., a tree), which makes approximation algorithms an interesting alternative. Here again the problem shows itself to be hard. Without assuming that variables have bounded cardinality, there is no polynomial-time approximation algorithm unless P=NP. As we show here, neither restricting the diagrams to a single value variable makes the problem easier to solve or approximate.

In this paper, we give some hope in light of so many negative results by showing that when the diagram has bounded treewidth and the variables take on a bounded number of values, there is a fully polynomial-time approximation algorithm for the (optimal) strategy selection problem. Although our proof is constructive, the algorithm we provide is not expected to be practical for any reasonably large problem due to the huge constants hidden in the asymptotic analysis. Nevertheless, the existence of such an scheme shall motivate researchers to investigate more efficient approximation algorithms to solve influence diagrams of low treewidth and low variable cardinality.

Acknowledgements

This work was partially supported by the Swiss National Science Foundation grants no. 200020_134759/1 and 200020_137680/1, and Hasler Foundation grant no. 10030.

References

[1] C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21 (3):293–320, 2010.

[2] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[3] G. F. Cooper. A method for using belief networks as influence diagrams. *Fourth Workshop on Uncertainty in Artificial Intelligence*, 1988.

[4] C. P. de Campos and Q. Ji. Strategy selection in influence diagrams using imprecise probabilities. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, pages 121–128, 2008.

[5] R. Dechter. An anytime approximation for optimizing policies under uncertainty. In *Workshop of Decision Theoretic Planning, AIPS*, 2000.

[6] A. Detwarasiti and R. D. Shachter. Influence diagrams for team decision analysis. *Decision Analysis*, 2(4):207–228, 2005.

[7] R. A. Howard and J. E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762. Strategic Decisions Group, 1984.

[8] F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*. Information Science and Statistics. Springer, 2nd edition, 2007.

[9] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[10] S. L. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1235–1251, 2001.

[11] D. D. Mauá and Cassio P. de Campos. Solving decision problems with limited information. In *Advances in Neural Information Processing Systems 24*, pages 603–611. 2011.

[12] D. D. Mauá, C. P. de Campos, and M. Zaffalon. Solving limited memory influence diagrams. *Journal of Artificial Intelligence Research*, 44:97–140, 2012.

[13] N. Meuleau, K.-E. Kim, L. P. Kaelbling, and A. R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 417–426, 1999.

[14] D. Nilsson and M. Höhle. Computing bounds on expected utilities for optimal policies based on limited information. Research Report 94, Dina, 2001.

[15] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 86–92. IEEE Computer Society, 2000.

[16] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

[17] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[18] P. P. Shenoy. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of Approximate Reasoning*, 17 (2–3):239–263, 1997.

[19] J. A. Tatman and R. D. Shachter. Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2): 365–379, 1990.

[20] X. Wu, A. Kumar, and S. Zilberstein. Influence diagrams with memory states: Representation and algorithms. In *Proceedings of the 2nd International Conference on Algorithmic Decision Theory*, pages 306–319, 2011.

[21] N. L. Zhang, R. Qi, and D. Poole. A computational theory of decision networks. *International Journal of Approximate Reasoning*, 11(2):83–158, 1994.

Learning STRIPS Operators from Noisy and Incomplete Observations

Kira Mourão
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
kmourao@inf.ed.ac.uk

Luke Zettlemoyer
Computer Science & Engineering
University of Washington
Seattle, WA98195
lsz@cs.washington.edu

Ronald P. A. Petrick
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
rpetrick@inf.ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
steedman@inf.ed.ac.uk

Abstract

Agents learning to act autonomously in real-world domains must acquire a model of the dynamics of the domain in which they operate. Learning domain dynamics can be challenging, especially where an agent only has partial access to the world state, and/or noisy external sensors. Even in standard STRIPS domains, existing approaches cannot learn from noisy, incomplete observations typical of real-world domains. We propose a method which learns STRIPS action models in such domains, by decomposing the problem into first learning a transition function between states in the form of a set of classifiers, and then deriving explicit STRIPS rules from the classifiers' parameters. We evaluate our approach on simulated standard planning domains from the International Planning Competition, and show that it learns useful domain descriptions from noisy, incomplete observations.

1 INTRODUCTION

Developing agents with the ability to act autonomously in the world is a major goal of artificial intelligence. One important aspect of this development is the acquisition of domain models to support planning and decision-making: to operate effectively in the world, an agent must be able to accurately predict when its actions will succeed, and what effects its actions will have. Only when a reliable action model is acquired can the agent usefully combine sequences of actions into plans, in order to achieve wider goals. However, learning domain dynamics can be a challenging problem: agents' observations may be noisy, or incomplete; actions may be non-deterministic; the world may be noisy or contain many irrelevant objects and relations.

In this paper we consider the problem of acquiring explicit domain models from the raw experiences of an agent exploring the world, where the agent's observations are incomplete, and observations and actions are subject to noise. The domains we consider are relational STRIPS (Fikes and Nilsson, 1971) domains, although our approach has the potential to be extended to more expressive domains. Given the autonomous learning setting, we assume only a weak domain model where the agent knows how to identify objects, has acquired predicates to describe object attributes and relations, and knows what types of actions it may perform, but not the appropriate contexts for the actions, or their effects. Experience in the world is then developed through observing changes to object attributes and relations when motor-babbling with primitive actions.

Other approaches to learning STRIPS operators do not handle both noisy and incomplete observations (see Section 3). We develop a two-stage approach to the problem which decouples the requirement to tolerate noisy, incomplete observations from the requirement to learn compact STRIPS operators. In the first stage we learn action models by constructing a set of kernel classifiers which tolerate noise and partial observability, but whose action models are implicit in the learnt parameters of the classifiers, similar to the work of Mourão *et al.* (2009, 2010). However, we additionally use the method to learn preconditions as well as effects, as suggested but not explored in earlier work. Also, we evaluate additional kernels for the learning problem and select a better performing alternative. The initial action model learnt in this first stage acts as a noise-free, fully observable source of observations from which to extract explicit action rules. In the second stage we devise a novel method to derive explicit STRIPS operators from the model implicit in the kernel classifiers. In experiments the resulting rules perform as well as the original classifiers, while providing a compact representation of the action models suitable for use in automated planning systems.

2 THE LEARNING PROBLEM

A *domain* is a tuple $\mathcal{D} = \langle \mathcal{O}, \mathcal{P}, \mathcal{A} \rangle$, where $\mathcal{O}$ is a finite set of world objects, $\mathcal{P}$ is a finite set of predicate (relation) symbols, and $\mathcal{A}$ is a finite set of actions. Each predicate and action also has an associated arity. A *fluent expression* is a statement of the form $p(c_1, c_2, ..., c_n)$, where $p \in \mathcal{P}$,

n is the arity of p, and each $c_i \in \mathcal{O}$. A *state* is any set of fluent expressions, and $\mathcal{S}$ is the set of all possible states. Since state observations may be incomplete we assume an open world where unobserved fluents are considered to be unknown. For any state $s \in \mathcal{S}$, a fluent expression ϕ is true at s iff $\phi \in s$. The negation of a fluent expression, $\neg\phi$, is true at s (also, ϕ is false at s) iff $\neg\phi \in s$. If $\phi \in s$ then $\neg\phi \notin s$. Any (legal) fluent expression not in s is unobserved.

Each action $a \in \mathcal{A}$ is defined by a set of *preconditions*, Pre_a, and a set of *effects*, Eff_a. In STRIPS domains Pre_a can be any set of fluent expressions and Eff_a can be any set of fluent expressions and negated fluent expressions. Action preconditions and effects can also be parameterised. An action with all of its parameters replaced with objects from $\mathcal{O}$ is said to be an *action instance*. Objects mentioned in the preconditions or the effects must be listed in the action parameters (the *STRIPS scope assumption* (SSA)).

The task of the learning mechanism is to learn the preconditions and effects Pre_a and Eff_a for each $a \in \mathcal{A}$, from data generated by an agent performing a sequence of randomly selected actions in the world and observing the resulting states. The sequence of states and action instances is denoted by $s_0, a_1, s_1, ..., a_n, s_n$ where $s_i \in \mathcal{S}$ and a_i is an instance of some $a \in \mathcal{A}$. Our data consists of *observations* of the sequence of states and action instances $s'_0, a_1, s'_1, ..., a_n, s'_n$, where state observations may be noisy (some $\phi \in s_i$ may be observed as $\neg\phi \in s'_i$) or incomplete (some $\phi \in s_i$ are not in s'_i). Action failures are allowed: the agent may attempt to perform actions whose preconditions are unsatisfied. In these cases the world state does not change, but the observed state may still be noisy or incomplete. To make accurate predictions in domains where action failures are permitted, the learning mechanism must learn both preconditions and effects of actions.

E.g. in BlocksWorld, where an agent stacks and unstacks blocks, consider a state with blocks `B1` and `B2` on the table:

```
(AND armempty (ontable B1) (ontable B2) (clear B1)
 (clear B2) (NOT (on B1 B2)) (NOT (on B2 B1))
 (NOT (holding B1)) (NOT (holding B2))).
```

A corresponding noisy, incomplete observation is:

```
(AND armempty (ontable B2) (holding B1) (clear B1)
 (NOT (on B1 B2)) (NOT (holding B2))).
```

It is noisy as `(holding B1)` is incorrect, and incomplete as `(ontable B1)`, `(clear B2)` and `(NOT(on B2 B1))` are missing. A sequence of noisy, incomplete observations of states and actions could be as follows:

```
s'_0: (AND armempty (ontable B2) (holding B1)
       (clear B1) (NOT (on B1 B2)) (NOT (holding B2)))
a_1: (pickup B1)
s'_1: (AND (NOT (clear B1)) (holding B1) (NOT (on B1 B2))
       (NOT (ontable B2)) (NOT (on B2 B1)))
a_2: (stack B1 B2)                                    (1)
s'_2: (AND armempty (clear B1) (clear B2)
       (ontable B2) (NOT (holding B1)))
a_3: (stack B2 B1)
s'_3: (AND armempty (clear B1) (NOT (clear B2))
       (NOT (on B2 B1)) (NOT (on B2))).
```

Taking a sequence of such inputs, we learn action descriptions for each action in the domain. For example, the `stack` action, which moves a block from the gripper on to another block, would be represented as:

```
(:action stack
  :parameters (?ob ?underob)
  :precondition (and (clear ?underob) (holding ?ob))
  :effect (and (arm-empty) (clear ?ob) (on ?ob ?underob)
          (not (clear ?underob)) (not (holding ?ob)))).
```

3 RELATED WORK

Previous work fulfils some but not all of the requirements for learning models in the setting we consider. In autonomous robotics, various techniques exist to learn preconditions and effects of actions in noisy, partially observable worlds (Doğar *et al.*, 2007; Metta and Fitzpatrick, 2003; Modayil and Kuipers, 2008). However, none of these approaches learn relational models. Conversely, much previous work on learning relational action models relies on the provision of prior knowledge of the action model. Strategies include seeding initial models with approximate planning operators (Gil, 1994), using known successful plans (Wang, 1995), excluding action failures (Amir and Chang, 2008), or the presence of a teacher (Benson, 1996). Such knowledge is unlikely to be available to an autonomous agent learning the dynamics of its domain. Additionally, only a few approaches are capable of learning under either partial observability (Amir and Chang, 2008; Yang *et al.*, 2007; Zhuo *et al.*, 2010), noise in any form (Pasula *et al.*, 2007; Rodrigues *et al.*, 2010), or both (Halbritter and Geibel, 2007; Mourão *et al.*, 2010). Approaches which handle both do so by generating implicit action models which must be used as a black-box to make predictions of state changes, and do not generate symbolic action representations.

We extract rules from classifiers based on the intuition that more discriminative features will contribute more to the classifier's objective function. This is similar to the insight underlying feature selection methods of the type which rank each feature according to the sensitivity of the classifier's objective function to the removal of the feature (Kohavi and John, 1997; Guyon *et al.*, 2002). Our approach differs in that features are selected separately for individual examples, rather than once across the entire training set, and we define a stopping criterion which identifies when the set of selected features can no longer form a rule.

Our work also has links with earlier work in version spaces (Mitchell, 1982) and the associated greedy search, which underlie many other approaches to rule learning (e.g. Amir and Chang, 2008; Pasula *et al.*, 2007). Our rule search benefits from extra information to guide the search, in the form of the weights associated with each hypothesis by the previously trained statistical classifiers, which can be highly robust to noise and incomplete observations.

4 LEARNING IMPLICIT MODELS

The basis of our approach is the division of the learning problem into two parts: initially a classification method is used to learn to predict effects of actions, then STRIPS rules are derived from the resulting action representations. We define the *implicit action model* to be the model of the domain implicit in the learnt parameters of the classifiers, and the *explicit action model* to be the domain model described by STRIPS rules. In learning an implicit action model, our approach follows earlier work (Croonenborghs *et al.*, 2007; Halbritter and Geibel, 2007; Mourão *et al.*, 2009, 2010) which encodes the learning problem in terms of the inputs and outputs of a set of classifiers. However, none of these methods generate explicit action models.

Following the approach of Mourão *et al.* (2009, 2010), we encode the state descriptions with a fixed dimension vector representation by only considering objects which are mentioned in the action parameter list. By the SSA this is sufficient to learn STRIPS rules. For an action instance with arguments $o_1, o_2, ..., o_n$ we therefore restrict the state description to all possible fluents whose arguments are in $\{o_1, o_2, ..., o_n\}$. Also we schematise descriptions by replacing the i-th action parameter with the label arg_i whenever it appears in any fluent. Thus all state descriptions are now written in terms of the action parameters and not in terms of specific objects. The SSA fixes a small number of objects to consider for an action, as well as their roles, which allows relational state descriptions to be encoded in a vector, as each possible fluent in a state maps to exactly one possible fluent in any other state. We encode state descriptions as vectors where each bit in the vector corresponds to each possible fluent which could exist in the schematised state description. The value of a bit is 1 if the fluent is true, -1 if false, and a wildcard value * if unobserved.

In our previous BlocksWorld example (1), the state descriptions for s'_0 and s'_1 in the context of the `pickup(B1)` action a_1 would include only `B1`, and are schematised to form prior and successor states s_{prior} = `(AND armempty(holding arg1)(clear arg1))` and s_{succ} = `(AND (holding arg1) (NOT(clear arg1)))` respectively, to give $\langle s_{prior}, \texttt{pickup(arg1)}, s_{succ}\rangle$ as a training example. Conversely, the descriptions for s'_1 and s'_2 in the context of the `stack(B1 B2)` action a_2 contain both `B1` and `B2` as both are action parameters, giving s_{prior} = `(AND (holding arg1) (NOT(on arg1 arg2)) (NOT(on arg2 arg1)) (NOT(clear arg1)) (NOT(ontable arg2)))` and s_{succ} = `(AND armempty (clear arg1) (clear arg2) (ontable arg2) (NOT(holding arg1)))`.

Encoding the state descriptions as vectors, where the bits correspond to `armempty` followed by `clear`, `ontable`, `holding` and `on` with first argument `arg1`, and then the same predicates with first argument `arg2`, we obtain $v^{prior} = \langle 1, 1, *, 1, *, *, *, *, * \rangle$ and $v^{succ} = \langle *, -1, *, 1, *, *, *, *, * \rangle$ for a_1 and $v^{prior} = \langle *, -1, *, 1, -1, *, -1, *, -1 \rangle$ and $v^{succ} = \langle 1, 1, *, -1, *, 1, 1, *, * \rangle$ for a_2.

Given the vectorised state descriptions, a changes vector v^{diff} is derived for each training example, where the i-th bit v_i^{diff} is defined as follows:

$$v_i^{diff} = \begin{cases} 0, \text{ if } v_i^{prior} = v_i^{succ} \text{ and } v_i^{succ}, v_i^{prior} \in \{-1, 1\} \\ 1, \text{ if } v_i^{prior} \neq v_i^{succ} \text{ and } v_i^{succ}, v_i^{prior} \in \{-1, 1\} \\ *, \text{ otherwise.} \end{cases}$$

A set of classifiers now learns the implicit action model. Each classifier $C_{a,i}$ corresponds to a particular action a in the domain and predicts the i-th bit of the changes vector. Thus if $\langle s_{prior}, a, s_{succ}\rangle$ is a training example for $C_{a,i}$ then the input to the classifier is v^{prior} with target value v_i^{diff}.

Voted perceptron classifiers equipped with a kernel function have previously been applied to the problem of learning action models (Mourão *et al.*, 2009, 2010) as they are computationally efficient and known to tolerate noise (Khardon and Wachman, 2007). We take a similar approach. Mourão *et al.* used voted perceptrons combined with a DNF kernel, $K(x, y) = 2^{same(x,y)}$, where $same(x, y)$ is the number of bits with equal values in x and y (Sadohara, 2001; Khardon and Servedio, 2005). In the $same(x, y)$ calculation, bits with unobserved values are excluded. The features of the DNF kernel are all possible conjunctions of fluents, seemingly ideal for learning action preconditions which are arbitrary conjunctions of fluents. However, DNF is not PAC-learnable by a perceptron using the DNF kernel, as examples exist on which it can make exponentially many mistakes (Khardon *et al.*, 2005). We therefore consider as an alternative the k-DNF kernel, whose features are all possible conjunctions of fluents of length $\leq k$ for some fixed k: $K(x, y) = \sum_{l=0}^{k} \binom{same(x,y)}{l}$ (Khardon and Servedio, 2005). Preconditions with more than k fluents are still possible since the voted perceptron supports hypotheses which are conjunctions of features.

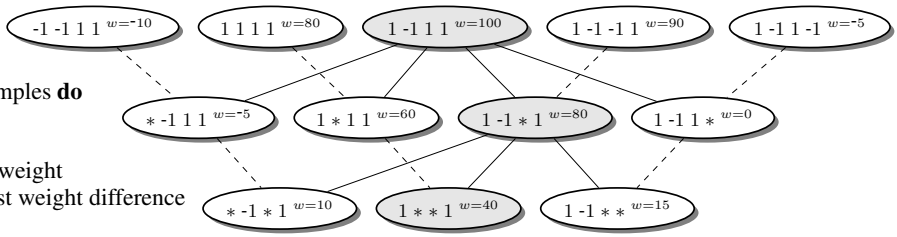
5 RULE EXTRACTION

Once the classifiers have been trained, the first step in deriving explicit action rules is to extract individual *per-effect* rules to predict each fluent in isolation. We will look at how to combine the rules to extract postconditions in Section 6.

The rule extraction process takes as input a classifier $C_{a,i}$ and returns a set of preconditions (as vectors) which predict that the fluent corresponding to bit i will change if action a is performed. For example, in the BlocksWorld domain a set of preconditions extracted from the classifiers for `stack` is shown in Figure 2.

for $v \in SV^+$ **do**
  $child := v$
  while $child$ only covers +ve training examples **do**
    $parent := child$
    for each valued bit in $parent$ **do**
      flip bit to its negation and calculate weight
    $child :=$ child whose parents have least weight difference
  $rule_v := parent$

(a) Rule Extraction Algorithm



(b) Example of Rule Extraction Process

Figure 1: Each node in (b) contains a vector corresponding to a possible precondition, and the weight w assigned to the vector by the voted perceptron model. Each level of the lattice contains vectors with one fewer feature than the level above. Lines join parent and children nodes: solid lines link the candidate parent rule at one level with its children in the level below, and dashed lines link children to their alternative parent. Shaded nodes are the preconditions selected at each iteration through the lattice. The positive support vector "seed" is the vector $\langle 1$ -1 1 $1\rangle$ with weight 100. Following the rule extraction algorithm in (a), the child whose parents have the least weight difference, the vector $\langle 1$ -1 * $1\rangle$, is chosen as the next candidate rule. The process ends with the rule $\langle 1$ * * $1\rangle$ as both children have a negative counterexample in the training data (not shown).

```
(armempty) changes when:
[8]  (AND (NOT(armempty)) (NOT(ontable arg1)))

(clear arg1) changes when:
[14] (AND (NOT(clear arg1)) (holding arg1)
      (NOT(on arg1 arg2)))
[12] (AND (NOT(clear arg1)) (NOT(ontable arg1))
      (NOT(on arg2 arg1)))
[8]  (AND (clear arg1) (ontable arg1) (clear arg2)
      (NOT(on arg1 arg2)) (NOT(on arg2 arg1)))

(ontable arg1) changes when:
[6]  (AND (NOT(ontable arg1)) (NOT(on arg1 arg2)))
[4]  (AND (NOT(armempty)) (ontable arg1)
      (NOT(clear arg2)))

(holding arg1) changes when:
[15] (AND (holding arg1))

(on arg1 arg2) changes when:
[3]  (AND (NOT(on arg1 arg2)))

(clear arg2) changes when:
[12] (AND (clear arg2) (ontable arg2)
      (NOT(holding arg2)))
[6]  (AND (NOT(armempty)) (NOT(clear arg1)) (clear arg2)
      (holding arg1) (NOT(on arg1 arg2)))
[2]  (AND (NOT(clear arg1)) (clear arg2)
      (NOT(ontable arg2)))
```

Figure 2: Per-effect rules generated for the BlocksWorld `stack` action from 1000 examples in a world with 5% noise and 25% observability. Weights are shown in square brackets. Fluents in bold are neither in, nor implied by, the true action specification. Many of these fluents will later be excluded by the rule combination process (Section 6).

Rules are extracted from a voted perceptron with kernel K and support vectors $SV = SV^+ \cup SV^-$, where SV^+ (SV^-) is the set of support vectors whose *predicted* values are 1 (-1). The positive support vectors are each instances of some rule learnt by the perceptron, and so are used to "seed" the search for rules. The extraction process aims to identify and remove all irrelevant bits in each support vector, using the voted perceptron's prediction calculation to determine which bits to remove.

The *weight* of any possible state description vector $\mathbf{x}$ is defined to be the value calculated by the voted perceptron's prediction calculation before thresholding (Freund and Schapire, 1999):

$$weight_e(\mathbf{x}) = \sum_{i=1}^{n} c_i \; sign \sum_{j=1}^{i} y_j \alpha_j K(\mathbf{x}_j, \mathbf{x})$$

where each x_i is one of the n support vectors, y_i is the corresponding target value, c_i and α_i are the parameters learnt by the classifier, and e is the effect predicted by the classifier. The predicted value for $\mathbf{x}$ is 1 if $weight_e(\mathbf{x}) > 0$ and -1 otherwise. A *child* of vector $\mathbf{x}$ is any distinct vector obtained by replacing a single bit of $\mathbf{x}$ with the value *. Similarly, a *parent* of $\mathbf{x}$ is any vector obtained by replacing a *-valued bit with the value 1 or -1.

The basic intuition behind the rule extraction process is that more discriminative features will contribute more to the weight of an example. Thus the rule extraction process operates by taking each positive support vector and repeatedly deleting the feature which contributes least to the weight until some stopping criterion is satisfied. This leaves the most discriminative features underlying the example, which can be used to form a precondition. An example of the process of extracting rules is shown in Figure 1(b), and an outline of the algorithm in Figure 1(a), as follows. Take each positive support vector v in turn, and aim to find a conjunction $rule_v$ which covers v and does not cover any negative training examples, but where every child of $rule_v$ covers at least one negative example. Construct $rule_v$ by a greedy algorithm which first takes v as a candidate rule and then repeatedly creates a new candidate rule by choosing one bit to set to the $*$ value. The bit is chosen by considering the difference in weights between the current candidate $\mathbf{x} = \langle x_1, ..., x_i, ..., x_n \rangle$ and

each $\mathbf{x}_{\neg i} = \langle x_1, ..., \neg x_i, ..., x_n \rangle$, finding

$$\underset{x_i \in \{x_1,...,x_i,...,x_n\}}{\operatorname{argmin}} (weight_e(\mathbf{x}) - weight_e(\mathbf{x}_{\neg i})).$$

Removing the resulting x_i removes the least discriminative bit in the current candidate rule. At each step the new candidate rule is tested against the training examples. If it classifies a negative training example as positive, then the rule is too general and $rule_v$ is set to the previous candidate rule, otherwise the process repeats. The result is a set of rules for each action, predicting when a particular output bit changes. There may be many rules, up to one per positive support vector, each consisting of a set of preconditions which, if satisfied, predict the output bit will change.

6 RULE COMBINATION

The rule extraction process described above produces a set of rules for an action, such as for the BlocksWorld `stack` action shown in Figure 2. However, in STRIPS we expect a single rule for each action, consisting of a set of preconditions and a set of effects such as in the definition of the BlocksWorld `stack` action given in Section 2.

The rule combination process therefore builds a single STRIPS-like rule for each action, taking as input the set of rules produced by the rule extraction process: $\{(v_1, e_1), (v_2, e_2), \ldots, (v_r, e_r)\}$ where v_i is the vector representing the i-th set of preconditions, and e_i is the bit which changes when v_i holds. Rule combination generates a rule (v_{rule}, e_{rule}) where the j-th element of v_{rule}, $v_{rule,j} \subseteq \bigcup_i v_{i,j}$ and $e_{rule} \subseteq \bigcup_i e_i$. Given the definitions in Section 4, we can directly convert the single state vector v_{rule} and the set of effects e_{rule} into a precondition and effect in STRIPS format.[1]

Without noise or partial observability, the combination process is a straightforward conjunction of all preconditions and all effects in the set of rules for an action, i.e., $\forall j \; v_{rule,j} = \bigcup_i v_{i,j}$ and $e_{rule} = \bigcup_i e_i$. However, when learning from noisy examples, unwanted additional fluents can be introduced to the per-effect rules via noisy support vectors. Similarly, incomplete training examples can mean some necessary fluents are missing from individual per-effect rules. In this section we describe an approach to identify and eliminate fluents introduced by noise while adding in fluents omitted due to partial observability.

To support the process of choosing between different potential rules which may contain noisy fluents, or omit necessary fluents, we introduce two filtering functions. *AcceptPrecons* takes an existing precondition and effects (v_{rule}, e_{rule}) for an action a, and assesses whether a new precondition v_{new} predicts the effects of a at least as well as the current precondition. *AcceptEffects* takes an existing precondition and effects for an action a, and assesses whether the precondition predicts a new effect e_{new} of a at least as well as it predicts the current effects. We describe *AcceptPrecons* and *AcceptEffects* in detail in Section 6.2.

[1] Since the effects in e_{rule} are changes it may be necessary to identify from what value the change is made, by referring to the rule from which the effect bit originated.

$R := \{(v_1, e_1), \ldots, (v_r, e_r)\}$
$rule := (v_1, \varnothing)$
$locks = \varnothing$
while $R \neq \varnothing$ **do**
  $next :=$ highest weighted rule in R
  $R := R \setminus \{next\}$
  $v_{candidate} = CombinePrecons(rule, next, locks)$
  if $v_{candidate} \neq v_{rule}$ **then**
    $v_{candidate} = SimplifyPrecons(rule, next, v_{candidate})$
    if $AcceptPrecons(rule, v_{candidate})$ **then**
      $v_{rule} := v_{candidate}$
    if $AcceptEffect(rule, e_{next})$ **then**
      $e_{rule} := e_{rule} \cup e_{next}$
    $e_{rule} = SimplifyEffects(rule)$

Figure 3: Outline Rule Combination Algorithm

6.1 RULE COMBINATION OVERVIEW

With the filtering functions in place, we now describe how the rule combination process generates STRIPS rules by combining and refining the per-effect rules. Figure 3 gives an outline of the algorithm, described below.

For each action the process derives a rule (v_{rule}, e_{rule}) from the set of rules $R = \{(v_1, e_1), \ldots, (v_r, e_r)\}$ produced by rule extraction, ordered so that $weight_{e_i}(v_i) \geq weight_{e_j}(v_j)$ if $i < j$. The process first initialises v_{rule} to the highest weighted precondition in R and sets $e_{rule} = \varnothing$. The rule is then refined by combining it with each of the remaining per-effect rules in turn, in order of highest weight.

Each time (in *CombinePrecons*) the process combines the current precondition v_{rule} with the precondition from the next per-effect rule v_i, which we will name v_{next}, into a candidate precondition $v_{candidate}$. This includes resolving any conflicts between v_{rule} and v_{next}. We now have a candidate precondition $v_{candidate}$ which is a merge of v_{rule} and v_{next}. The process refines $v_{candidate}$ further by testing it against a set of alternatives in *SimplifyPrecons* and setting $v_{candidate}$ to the best result. Now $v_{candidate}$ is tested against the original v_{rule}, using *AcceptPrecons*. If $v_{candidate}$ is accepted, v_{rule} is updated to $v_{candidate}$. Similarly, e_{next} is tested against the original set of effects e_{rule}, using *AcceptEffects*. If e_{next} is accepted, e_{rule} is updated to $e_{rule} \cup e_{next}$. Finally, the process refines e_{rule} by testing it against a set of alternatives in *SimplifyEffects* and setting e_{rule} to the best result. In the next section we describe each of the subprocedures in detail.

6.2 ALGORITHM DETAILS

CombinePrecons: In attempting to combine (v_{rule}, e_{rule}) with (v_{next}, e_{next}), the first check is whether e_{next} contradicts any effect in e_{rule}. Effects conflict if both rules predict change to a fluent, but the rules have different values for the fluent in their preconditions. (For example, the two per-effect rules for `(ontable arg1)` in Figure 2 conflict.) If there is a conflict, the new rule is rejected, as we assume only one rule per action, and the higher weighted baseline rule is more likely to be correct.

Second, *CombinePrecons* combines the preconditions on every bit which is not locked (listed in $locks$) and does not conflict, i.e., $\forall i$ $v_{rule,i} = v_{next,i}$ or $v_{rule,i} = *$:

$$v_{candidate,i} = \begin{cases} v_{rule,i}, & \text{if } v_{rule,i} = v_{next,i} \text{ or } i \in locks \\ v_{rule,i}, & \text{if } v_{next,i} = * \\ v_{next,i}, & \text{if } v_{rule,i} = * \text{ and } i \notin locks. \end{cases}$$

For conflicts, where $v_{rule,i} \neq v_{next,i}$ and $v_{rule,i}, v_{next,i} \in \{1, -1\}$, *CombinePrecons* decides which value each conflict bit should take in $v_{candidate}$, as follows.

For each conflicting fluent, there are three possible values the fluent could take in the preconditions of the true rule: $*$ (unobserved), 1 (true) or -1 (false). The weight $weight_e$ (for each effect e in e_{rule}) of each variant is calculated (with the values of other conflicting fluents set to $*$). The preferred variant is where the value is $*$, indicating a non-discriminative feature, and giving the simplest precondition. However, a variant is only acceptable if the weight of the resulting precondition is positive for all effects in e_{rule}, since then the new precondition still predicts the same effects as the current precondition v_{rule}. If accepted, the fluent is locked at the $*$ value, to prevent later, possibly noisy rules, from resetting it. Locked fluents are recorded in the $locks$ variable (see Figure 3). If the $*$-variant is unacceptable, then the (1)-valued or (-1)-valued cases are considered, provided they have positive weights on all the effects. If both variants are acceptable, whichever has the highest average weight over all the effects is selected. If neither variant is acceptable then the conflict is unresolved for this fluent. As long as the conflicts on every fluent are resolved, the rule combination process can continue with the new candidate precondition. If not, the current rule is rejected (and *CombinePrecons* returns v_{rule}).

SimplifyPrecons: Once *CombinePrecons* has generated a candidate precondition $v_{candidate}$, *SimplifyPrecons* considers alternative, less specific preconditions. It creates a set of alternatives $v_{candidate\backslash i}$ for each bit i in $v_{candidate}$ which differs from v_{rule}. $v_{candidate\backslash i} = v_{candidate}$ except at bit i where $v_{candidate\backslash i} = *$. Whenever the filtering function *AcceptPrecons* rates $v_{candidate}$ as worse than any $v_{candidate\backslash i}$, the associated fluent $v_{candidate,i}$ is set to $*$.

SimplifyEffects: In light of the new preconditions, *SimplifyEffects* tests if any of the effects should be removed from the new v_{rule} . For instance, more specific preconditions may lower the incidence of some effects (as seen in the training data) to the extent that *AcceptEffects* rejects them. Each effect is tested against all the other effects by *AcceptEffects* and, if rejected, removed from e_{rule}.

AcceptPrecons: In the precondition filtering function, before even making a comparison between preconditions, the new precondition v_{new} must be checked to ensure that it is consistent with the classifiers and supported by the training data. For this we require a notion of coverage of the training set. Coverage is defined to account for partial observability, so that precondition v_{new} covers example x at effect e (denoted $covers_e(v_{new}, x)$) if none of the fluents in the example state contradict the fluents in the rule preconditions, and e is in both the example state changes and the rule effects. Now v_{new} can form a rule precondition if:

1. v_{new} is consistent with the classifiers: for each $e \in e_{rule}$, v_{new} should be classified by the classifier $C_{a,e}$ as predicting change, that is,
   $\forall e \in e_{rule}\ weight_e(v_{new}) > 0$; and
2. v_{new} is supported by the training data: for each $e \in e_{rule}$, v_{new} should cover at least one training example where e changed, that is,
   $\forall e \in e_{rule}\ |\{x : covers_e(v_{new}, x)\}| > 0$.

Both should be considered, as weight alone may permit rules which do not cover any training examples, while coverage alone may allow negatively weighted rules.

Additionally, *AcceptPrecons* uses differences in precision and recall to identify and reject any new precondition which performs significantly worse than the existing precondition. It rejects preconditions where either the precision or recall on the training set drops substantially for any $e \in e_{rule}$. Since precision and recall is a trade-off, the comparison is made using the F-score[2] for precondition pre at effect e: $F_{pre,e}$. Ideally we want the new precondition to improve on (or at least not worsen) the F-score, but we must introduce some tolerance to account for the effects of noise.

For instance, suppose $v_{new} = \langle 1, 1, *, * \rangle$ is more general than $v_{rule} = \langle 1, 1, 0, * \rangle$, and that v_{new} is in fact the true rule. The F-score for v_{rule} is calculated on the subset of training examples which v_{rule} covers, while the F-score for v_{new} also includes training examples which $\langle 1, 1, 1, * \rangle$ covers. If the training set happens to have a higher proportion of training examples with a noisy outcome covered by $\langle 1, 1, 1, * \rangle$ than $\langle 1, 1, 0, * \rangle$ then the F-score for v_{new} can be lower than for v_{rule}. To account for such effects of noise, we allow the new F-score $F_{v_{new},e}$ to drop to some fraction ϵ_p of the F-score for the existing precondition $F_{v_{rule},e}$, for any effect $e \in e_{rule}$. For new F-scores below this value, the new precondition is rejected.

[2]F-score is the harmonic mean of precision and recall (true positives/predicted changes and true positives/actual changes, respectively) (Van Rijsbergen, 1979).

AcceptEffect: The effects filtering function similarly compares F-scores. Given (v_{rule}, e_{rule}) it compares how well v_{rule} predicts a new effect e_{new} relative to how well it predicts each $e \in e_{rule}$: specifically it compares $F_{v_{rule},e_{new}}$ to $F_{v_{rule},e}$ for each $e \in e_{rule}$. This identifies effects which are inconsistent with the other effects in terms of precision and recall. In particular, effects which occur in far fewer examples than other effects are identified in this way: these are likely to be caused by noise, or could be conditional effects. An effect is rejected by the function if its F-score is less than some fraction ϵ_e times the F-score on any other effect of the same rule.

In our evaluation, ϵ_p and ϵ_e were set to 0.95 and 0.5 respectively, and not varied across the domains. The values were selected empirically via experiments on a holdout dataset from one experimental domain (ZenoTravel).

6.3 RULE COMBINATION EXAMPLE

We now consider an example of one iteration of the rule combination process. Working with the BlocksWorld domain `stack` action, suppose the current rule is $(\langle *, -1, -1, 1, -1, 1, 1, *, * \rangle, \{1, 3, 5\})$ corresponding to the precondition `(AND (NOT(clear arg1)) (NOT(ontable arg1)) (holding arg1) (NOT(on arg1 arg2)) (clear arg2) (ontable arg2))` and effects {`(clear arg1) (holding arg1) (clear arg2)`}. We try to combine this with the new rule $(\langle *, *, -1, *, -1, *, -1, *, 1 \rangle, \{2\})$ corresponding to the precondition `(AND (NOT(ontable arg1)) (NOT(on arg1 arg2)) (NOT(ontable arg2)) (on arg2 arg1))` and effects {`(ontable arg1)`}.

CombinePrecons finds no conflicts in the effects, and generates the candidate precondition $\langle *, -1, -1, 1, -1, 1, ?, *, 1 \rangle$ where $?$ denotes a conflicting fluent. To resolve the conflict the weights of the vectors $\langle *, -1, -1, 1, -1, 1, *, *, 1 \rangle$, $\langle *, -1, -1, 1, -1, 1, 1, *, 1 \rangle$ and $\langle *, -1, -1, 1, -1, 1, -1, *, 1 \rangle$ are calculated for each of the $C_{stack,(clear\ arg_1)}$, $C_{stack,(holding\ arg_1)}$ and $C_{stack,(clear\ arg_2)}$ classifiers. If $\langle *, -1, -1, 1, -1, 1, *, *, 1 \rangle$ is accepted as $v_{candidate}$, *SimplifyPrecons* would consider the alternative precondition $\langle *, -1, -1, 1, -1, 1, *, *, * \rangle$ as the last bit in $v_{candidate}$ was different in v_{rule}.

Assuming $\langle *, -1, -1, 1, -1, 1, *, *, * \rangle$ is accepted as $v_{candidate}$ by *SimplifyPrecons*, it is compared to the original v_{rule} (the only difference now is that the bit corresponding to `(ontable arg2)` is unset in $v_{candidate}$). $v_{candidate}$ has higher weight and so *AcceptPrecons* accepts $v_{candidate}$ and $v_{rule} := v_{candidate}$. Conversely the new effect is rejected by *AcceptEffects* so $e_{rule} = \{1, 3, 5\}$ as before. Finally, *SimplifyEffects* uses *AcceptEffects* to test the relative prediction performance of the new v_{rule} on each effect, with no changes. The new rule is $(\langle *, -1, -1, 1, -1, 1, *, *, * \rangle, \{1, 3, 5\})$.

7 EXPERIMENTS

We tested our approach on several simulated domains taken from the International Planning Competition (IPC) at `http://ipc.icaps-conference.org/`. The domains differ in terms of the number and arity of actions and predicates, and the number and hierarchy of types. The main domain characteristics are detailed in Table 1.

Sequences of random actions and resulting states were generated from the PDDL domain descriptions and used as training and testing data. All data was generated using the Random Action Generator 0.5 available at `http://magma.cs.uiuc.edu/filter/`, modified to also generate action failures. Table 2 shows the numbers of objects used in training and testing data for each domain.

Ten different randomly generated training and testing sets were used. Training and testing sets were sequences of 20,000 and 2,000 actions respectively. Both sequences contained an equal mixture of successful and unsuccessful actions (where some precondition of the action was not satisfied, and so no change occurred in the world). In some domains (e.g. Rovers), portions of the state space can only be traversed once, and in these cases multiple shorter sequences of 400 actions were generated from randomly generated starting states. In line with previous work (Amir and Chang, 2008), incomplete observations were simulated by randomly selecting a fraction (10%, 25% or 50%) of fluents (including negations) from the world to observe after each action. The remaining fluents were discarded and the reduced state vector was generated from the observed fluents. Sensor noise was simulated similarly by flipping the value of each bit in the state vector with probability 1% and 5%.

7.1 RESULTS

We first tested the performance of different kernels on learning the implicit action model, comparing results for a standard (non-kernelised) perceptron, a voted (non-kernelised) perceptron, and a voted kernel perceptron. Both the DNF kernel and k-DNF kernel with $k = 2, 3$ and 5 were tested. Performance was measured in terms of the F-score of the predictions on the test sets.

The fully observable, noiseless cases are easily learnt by any of the perceptrons tested. After 5,000 training examples, the F-score on the test set is 1, in almost all cases. Per-

Table 1: Domain Characteristics

Domain	**Actions**		**Predicates**	
	No.	Max arity	No.(+types)	Max arity
BlocksWorld	4	2	5	2
Depots	5	4	6 (+6)	2
ZenoTravel	5	6	8 (+4)	2
DriverLog	6	4	6 (+4)	2
Rovers	9	6	25 (+7)	3

Table 2: Number of Objects in Training and Testing Worlds

Domain	Training	Testing
BlocksWorld	13 blocks	30 blocks
Depots	1 depot	4 depots
	2 distributors	4 distributors
	2 trucks	4 trucks
	3 pallets	10 pallets
	3 hoists	8 hoists
	10 crates	8 crates
ZenoTravel	5 cities	10 cities
	3 planes	5 planes
	7 people	10 people
DriverLog	3 road junctions	20 road junctions
	3 drivers	5 drivers
	7 packages	25 packages
	3 trucks	5 trucks
Rovers	2 rovers	4 rovers
	4 waypoints	8 waypoints
	3 objectives	4 objectives
	3 cameras	4 cameras
	3 modes	3 modes
	2 stores	4 stores
	1 lander	1 lander

formance of the voted perceptron, with or without the various kernels, is almost identical (results not shown). With the introduction of unobserved fluents or noise, the voted perceptron performs better than the standard perceptron. However, the DNF kernel does not improve performance, with the unkernelised voted perceptron learning significantly more accurate action models. In contrast, the k-DNF kernels all produce significantly more accurate models than the DNF kernel or no kernel ($p < 0.05$, repeated measures ANOVA with post-hoc Bonferroni t-test). Figure 4 gives a comparison of the relative performance of each model. The use of k-DNF kernels therefore represents a significant improvement on previous work which used only the DNF kernel. In light of these results, the 3-DNF kernel was selected for the remainder of the experiments.

Next, we extracted explicit rules from the implicit action models. There was no statistically significant difference between the F-scores of predictions made by the perceptron models and those made by the extracted rules (repeated measures ANOVA, $p > 0.05$). We also compared the resulting models to the original domain descriptions using a measure of error rate (Zhuo *et al.*, 2010). The error rate for a single action is defined as the number of extra or missing fluents in the preconditions and effects (E_{pre} and E_{eff} respectively) divided by the number of possible fluents in the preconditions and effects (T): $Error(a) = \frac{1}{2T}(E_{pre} + E_{eff})$. The error rate of a domain model with a set of actions A is: $Error(A) = \frac{1}{|A|}\sum_{a \in A} Error(a)$.

The error rates indicate that the learnt models are close to the actual STRIPS domain definitions, falling below 0.1 after around 5,000 examples in all cases (Figure 5). In particular, for fully observable, noiseless domains the correct STRIPS model is given by the extracted rules in fewer than

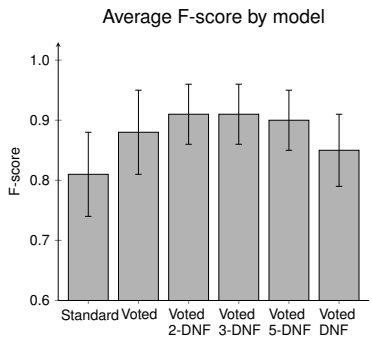


Figure 4: Comparison of the performance of different perceptrons learning action models from 20,000 random actions in STRIPS domains, averaged across all domains, levels of noise and partial observability. Error bars are standard error. Performance is significantly different between models which use a k-DNF kernel and those which do not.

2,000 training examples, except for the most complex domain, Rovers. Comparisons with other approaches in the literature are difficult due to differences in the learning settings. Nevertheless it is notable that the error rates of the learnt action models are low in comparison to action models learnt by (Yang *et al.*, 2007) for the same domains: their error rates at 90% observability (the highest reported) range from around 0.04 (ZenoTravel) to 0.1 or above (DriverLog and Depots) to more than 0.6 (Rovers). An example action model is shown in Figure 6, demonstrating that the method derives compact STRIPS-like rules even with high levels of incompleteness and noise in the observations.

We also calculated the F-scores for predictions made by the learnt rules on our (noiseless, fully observable) test sets (Figure 5). The F-scores are above 0.9 for all noise levels at 25% observability and above, for all domains except Rovers, indicating that in practice the rules correctly predict most fluents. The Rovers F-scores are somewhat lower, because there are fewer training examples per action than for the other domains, and more possible fluents.

Furthermore, our learning is fast. The longest-running example in the experiments (Rovers with 20,000 training examples, 5% noise, fully observable) takes under 1.5 hours on a single Intel Xeon 5160 processor to train the classifiers, and run rule extraction and combination. The ZenoTravel example (Figure 6) runs in under 2 minutes.

8 CONCLUSIONS AND FUTURE WORK

The results demonstrate that our approach successfully learns STRIPS operators from noisy, incomplete observations, in contrast to previous work which either generates explicit operators but cannot tolerate noise and incomplete examples, or tolerates noise and incomplete examples but does not generate explicit operators. We also show empirically that the 3-DNF kernel is a more appropriate choice

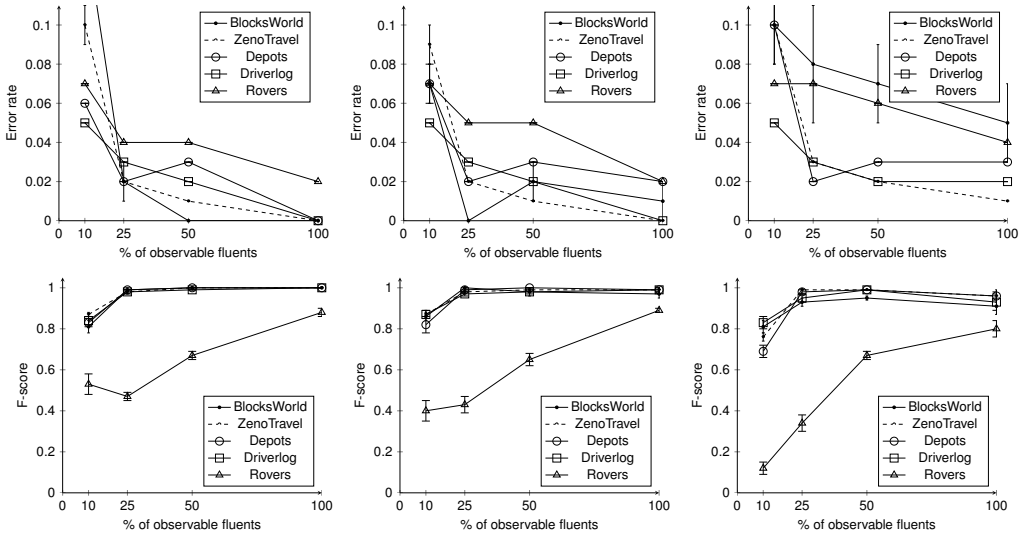


Figure 5: Results from learning explicit action rules from 5,000 training examples at varying levels of observability and noise in simulated planning domains. The error rate measures errors in the learnt domain model relative to the actual domain model (above). The F-score measures performance of the rules on fully observable, noiseless test domains (below).

than the DNF kernel for learning in this setting.

Our approach depends on decomposing the learning problem into two stages: learning implicit action models and then deriving explicit rules from the implicit models. Crucially, the implicit models produce noise-free, complete observations *for the domain model which has been learnt*. An alternative approach to our rule derivation process would be to apply existing action model learning techniques to the observations produced by the implicit models. However such an approach effectively restarts the learning process, ignoring information already learnt and available in the perceptron models, and so is likely to be less efficient.

Our approach also depends on the STRIPS scope assumption (SSA) which essentially identifies the objects which are relevant to the action and fixes their roles. In real-world scenarios the SSA may not apply. Without the SSA, during learning we must also consider state relating to objects which are not listed in the action parameters. Implicit action models in this setting may be learnt using a graphical representation of states combined with a suitable graph kernel (Mourão, 2012). In future work we therefore plan to extend our rule extraction method to derive rules from classifiers trained with graphical state representations. Additional steps will be required to efficiently handle the complexity introduced by the requirement to perform comparisons between graphical state descriptions. There are positive results in PAC-learning existential conjunctive and k-DNF concepts in noise-free structural domains with Boolean relations (Haussler, 1989; Valiant, 1985), which apply to learning from the implicit models, suggesting that our approach will scale to graphical state representations.

```
(:action DEBARK
:parameters (?x1 ?x2 ?x3 )
:precondition (AND (in ?x1 ?x2) (at ?x2 ?x3))
:effect (AND (at ?x1 ?x3) (NOT(in ?x1 ?x2))))

(:action BOARD
:parameters (?x1 ?x2 ?x3 )
:precondition (AND (at ?x1 ?x3) (at ?x2 ?x3))
:effect (AND (NOT(at ?x1 ?x3)) (in ?x1 ?x2)))

(:action FLY
:parameters (?x1 ?x2 ?x3 ?x4 ?x5 )
:precondition (AND (at ?x1 ?x2) (fuel-level ?x1 ?x4)
        (next ?x5 ?x4))
:effect (AND (NOT(at ?x1 ?x2)) (at ?x1 ?x3)
        (NOT(fuel-level ?x1 ?x4)) (fuel-level ?x1 ?x5)))

(:action ZOOM
:parameters (?x1 ?x2 ?x3 ?x4 ?x5 ?x6 )
:precondition (AND (at ?x1 ?x2) (fuel-level ?x1 ?x4)
        (next ?x6 ?x5) (next ?x5 ?x4))
:effect (AND (NOT(at ?x1 ?x2)) (at ?x1 ?x3)
        (NOT(fuel-level ?x1 ?x4)) (fuel-level ?x1 ?x6)))

(:action REFUEL
:parameters (?x1 ?x2 ?x3 ?x4 )
:precondition (AND (fuel-level ?x1 ?x3) (next ?x4 ?x3)
        (next ?x3 ?x4) (at ?x1 ?x2))
:effect (AND (NOT(fuel-level ?x1 ?x3)) (fuel-level ?x1 ?x4))
```

Figure 6: Explicit action model output for the ZenoTravel domain after 5,000 training examples with 10% observability and 5% noise. Missing fluents are in bold italic, incorrect fluents in bold. The error rate of this example is 0.05. Such imperfect rules have quite small effects on performance (F-score 0.85 in this case), but will in future work be improved by eliminating low reliability classifiers.

Acknowledgements

The authors are grateful to the reviewers of this and previous versions of this paper for helpful comments. This work was partially funded by the European Commission through the EU Cognitive Systems project Xperience (FP7-ICT-270273) and the UK EPSRC/MRC through the Neuroinformatics and Computational Neuroscience Doctoral Training Centre, University of Edinburgh.

References

Amir, E. and Chang, A. (2008). Learning partially observable deterministic action models. *JAIR*, **33**, 349–402.

Benson, S. S. (1996). *Learning Action Models for Reactive Autonomous Agents*. Ph.D. thesis, Stanford University.

Croonenborghs, T., Ramon, J., Blockeel, H., and Bruynooghe, M. (2007). Online learning and exploiting relational models in reinforcement learning. In *Proc. of IJCAI 2007*, pages 726–731.

Doğar, M. R., Çakmak, M., Uğur, E., and Şahin, E. (2007). From primitive behaviors to goal directed behavior using affordances. In *Proc. of IROS 2007*, pages 729–734.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, **2**, 189–208.

Freund, Y. and Schapire, R. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, **37**, 277–96.

Gil, Y. (1994). Learning by experimentation: Incremental refinement of incomplete planning domains. In *Proc. of ICML 1994*, pages 87–95.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, **46**(1-3), 389–422.

Halbritter, F. and Geibel, P. (2007). Learning models of relational MDPs using graph kernels. In *Proc. of MICAI 2007*, pages 409–419.

Haussler, D. (1989). Learning conjunctive concepts in structural domains. *Machine Learning*, **4**(1), 7–40.

Khardon, R. and Servedio, R. A. (2005). Maximum margin algorithms with Boolean kernels. *JMLR*, **6**, 1405–1429.

Khardon, R. and Wachman, G. M. (2007). Noise tolerant variants of the perceptron algorithm. *JMLR*, **8**, 227–248.

Khardon, R., Roth, D., and Servedio, R. A. (2005). Efficiency versus convergence of Boolean kernels for online learning algorithms. *JAIR*, **24**, 341–356.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artif. Intell.*, **97**(1-2), 273–324.

Metta, G. and Fitzpatrick, P. (2003). Early integration of vision and manipulation. *Adaptive Behavior*, **11**(2), 109–128.

Mitchell, T. (1982). Generalization as search. *Artif. Intell.*, **18**(2), 203–226.

Modayil, J. and Kuipers, B. (2008). The initial development of object knowledge by a learning robot. *Robotics and Autonomous Systems*, **56**(11), 879–890.

Mourão, K. (2012). *Learning Action Representations Using Kernel Perceptrons*. Ph.D. thesis, University of Edinburgh.

Mourão, K., Petrick, R. P. A., and Steedman, M. (2009). Learning action effects in partially observable domains (1). In *Proc. of ICAPS 2009 Workshop on Planning and Learning*, pages 15–22.

Mourão, K., Petrick, R. P. A., and Steedman, M. (2010). Learning action effects in partially observable domains (2). In *Proc. of ECAI 2010*, pages 973–974.

Pasula, H., Zettlemoyer, L. S., and Kaelbling, L. P. (2007). Learning symbolic models of stochastic domains. *JAIR*, **29**, 309–352.

Rodrigues, C., Gérard, P., and Rouveirol, C. (2010). Incremental learning of relational action models in noisy environments. In *Proc. of ILP 2010*, pages 206–213.

Sadohara, K. (2001). Learning of Boolean functions using support vector machines. In *Proc. of ALT*, pages 106–118.

Valiant, L. G. (1985). Learning disjunctions of conjunctions. In *Proc. of IJCAI 1985 - Volume 1*, pages 560–566.

Van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworth-Heinemann, 2nd edition.

Wang, X. (1995). Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proc. of ICML 1995*, pages 549–557.

Yang, Q., Wu, K., and Jiang, Y. (2007). Learning action models from plan examples using weighted MAX-SAT. *Artif. Intell.*, **171**(2-3), 107–143.

Zhuo, H. H., Yang, Q., Hu, D. H., and Li, L. (2010). Learning complex action models with quantifiers and logical implications. *Artif. Intell.*, **174**(18), 1540–1569.

Markov Chains on Orbits of Permutation Groups

Mathias Niepert
Universität Mannheim
mniepert@gmail.com

Abstract

We present a novel approach to detecting and utilizing symmetries in probabilistic graphical models with two main contributions. First, we present a scalable approach to computing generating sets of permutation groups representing the symmetries of graphical models. Second, we introduce orbital Markov chains, a novel family of Markov chains leveraging model symmetries to reduce mixing times. We establish an insightful connection between model symmetries and rapid mixing of orbital Markov chains. Thus, we present the first lifted MCMC algorithm for probabilistic graphical models. Both analytical and empirical results demonstrate the effectiveness and efficiency of the approach.

1 Introduction

Numerous algorithms exploit model symmetries with the goal of reducing the complexity of the computational problems at hand. Examples are procedures for detecting symmetries of first-order theories [7] and propositional formulas [2] in order to avoid the exhaustive exploration of a partially symmetric search space. More recently, symmetry detection approaches have been applied to answer set programming [11] and (integer) linear programming [26, 27, 34, 30]. A considerable amount of attention to approaches utilizing model symmetries has been given by work on "lifted probabilistic inference [36, 9]." Lifted inference is mainly motivated by the large graphical models resulting from statistical relational formalism such as Markov logic networks [38]. The unifying theme of lifted probabilistic inference is that inference on the level of instantiated formulas is avoided and instead lifted to the first-order level. Notable approaches are lifted belief propagation [41, 22], bisimulation-based approximate inference algorithms [40], first-order knowledge compilation techniques [44, 16], and lifted importance sampling approaches [17]. With the exception of some results for restricted model classes [41, 44, 21], there is a somewhat superficial understanding of the underlying principles of graphical model symmetries and the probabilistic inference algorithms utilizing such symmetries. Moreover, since most of the existing approaches are designed for relational models, the applicability to other types of probabilistic graphical models is limited.

The presented work contributes to a deeper understanding of the interaction between model symmetries and the complexity of inference by establishing a link between the degree of symmetry in graphical models and polynomial approximability. We describe the construction of colored graphs whose automorphism groups are equivalent to those of the graphical models under consideration. We then introduce the main contribution, *orbital Markov chains*, the first general class of Markov chains for lifted inference. Orbital Markov chains combine the compact representation of symmetries with generating sets of permutation groups with highly efficient product replacement algorithms. The link between model symmetries and polynomial mixing times of orbital Markov chains is established via a path coupling argument that is constructed so as to make the coupled chains coalesce whenever their respective states are located in the same equivalence class of the state space. The coupling argument applied to orbital Markov chains opens up novel possibilities of analytically investigating classes of symmetries that lead to polynomial mixing times.

Complementing the analytical insights, we demonstrate empirically that orbital Markov chains converge faster to the true distribution than state of the art Markov chains on well-motivated and established sampling problems such as the problem of sampling independent sets from graphs. We also show that existing graph automorphism algorithms are applicable to compute symmetries of very large graphical models.

2 Background and Related Work

We begin by recalling some basic concepts of group theory and finite Markov chains both of which are crucial for understanding the presented work. In addition, we give a brief overview of related work utilizing symmetries for the design of algorithms for logical and probabilistic inference.

2.1 Group Theory

A symmetry of a discrete object is a structure-preserving bijection on its components. For instance, a symmetry of a graph is a graph automorphism. Symmetries are often represented with permutation groups. A group is an abstract algebraic structure $(\mathfrak{G}, \circ)$, where $\mathfrak{G}$ is a set closed under a binary associative operation $\circ$ such that there is a identity element and every element has a unique inverse. Often, we refer to the group $\mathfrak{G}$ rather than to the structure $(\mathfrak{G}, \circ)$. We denote the size of a group $\mathfrak{G}$ as $|\mathfrak{G}|$. A permutation group *acting on* a finite set Ω is a finite set of bijections $\mathfrak{g} : \Omega \to \Omega$ that form a group.

Let Ω be a finite set and let $\mathfrak{G}$ be a permutation group acting on Ω. If $\alpha \in \Omega$ and $\mathfrak{g} \in \mathfrak{G}$ we write $\alpha^{\mathfrak{g}}$ to denote the image of α under $\mathfrak{g}$. A cycle $(\alpha_1\ \alpha_2\ ...\ \alpha_n)$ represents the permutation that maps α_1 to α_2, α_2 to α_3,..., and α_n to α_1. Every permutation can be written as a product of disjoint cycles where each element that does not occur in a cycle is understood as being mapped to itself. We define a relation $\sim$ on Ω with $\alpha \sim \beta$ if and only if there is a permutation $\mathfrak{g} \in \mathfrak{G}$ such that $\alpha^{\mathfrak{g}} = \beta$. The relation partitions Ω into equivalence classes which we call *orbits*. We use the notation $\alpha^{\mathfrak{G}}$ to denote the orbit $\{\alpha^{\mathfrak{g}} \mid \mathfrak{g} \in \mathfrak{G}\}$ containing α. Let $f : \Omega \to \mathbb{R}$ be a function from Ω into the real numbers and let $\mathfrak{G}$ be a permutation group acting on Ω. We say that $\mathfrak{G}$ is an *automorphism group for* (Ω, f) if and only if for all $\omega \in \Omega$ and all $\mathfrak{g} \in \mathfrak{G}$, $f(\omega) = f(\omega^{\mathfrak{g}})$.

2.2 Finite Markov chains

Given a finite set Ω a *finite Markov chain* defines a random walk $(X_0, X_1, ...)$ on elements of Ω with the property that the conditional distribution of X_{n+1} given $(X_0, X_1, ..., X_n)$ depends only on X_n. For all $x, y \in \Omega$ $P(x, y)$ is the chain's probability to transition from x to y, and $P^t(x, y) = P^t_x(y)$ the probability of being in state y after t steps if the chain starts at x. A Markov chain is *irreducible* if for all $x, y \in \Omega$ there exists a t such that $P^t(x, y) > 0$ and *aperiodic* if for all $x \in \Omega$, $\mathsf{gcd}\{t \geq 1 \mid P^t(x, x) > 0\} = 1$. A chain that is both irreducible and aperiodic converges to its unique stationary distribution.

The total variation distance d_{tv} of the Markov chain from its stationary distribution π at time t with initial state x is defined by

$$d_{\mathsf{tv}}(P^t_x, \pi) = \frac{1}{2} \sum_{y \in \Omega} |P^t(x, y) - \pi(y)|.$$

For $\varepsilon > 0$, let $\tau_x(\varepsilon)$ denote the least value T such that $d_{\mathsf{tv}}(P^t_x, \pi) \leq \varepsilon$ for all $t \geq T$. The *mixing time* $\tau(\varepsilon)$ is defined by $\tau(\varepsilon) = \max\{\tau_x(\varepsilon) \mid x \in \Omega\}$. We say that a Markov chain is *rapidly mixing* if the mixing time is bounded by a polynomial in n and $\log(\varepsilon^{-1})$, where n is the size of each configuration in Ω.

2.3 Symmetries in Logic and Probability

Algorithms that leverage model symmetries to solve computationally challenging problems more efficiently exist in several fields. Most of the work is related to the computation of symmetry breaking predicates to improve SAT solver performance [7, 2]. The construction of our symmetry detection approach is largely derived from that of symmetry detection in propositional theories [7, 2]. More recently, similar symmetry detection approaches have been put to work for answer set programming [11] and integer linear programming [34]. Poole introduced the notion of lifted probabilistic inference as a variation of variable elimination taking advantage of the symmetries in graphical models resulting from probabilistic relational formalisms [36]. Following Poole's work, several algorithms for lifted probabilistic inference were developed such as lifted and counting belief propagation [41, 22], bi-simulation-based approximate inference [40], general purpose MCMC algorithm for relational models [29] and, more recently, first-order knowledge compilation techniques [44, 16]. In contrast to existing methods, we present an approach that is applicable to a much larger class of graphical models.

3 Symmetries in Graphical Models

Similar to the method of symmetry detection in propositional formulas [7, 2, 8] we can, for a large class of probabilistic graphical models, construct a colored undirected graph whose automorphism group is equivalent to the permutation group representing the model's symmetries. We describe the approach for sets of partially weighted propositional formulas since Markov logic networks, factor graphs, and the weighted model counting framework can be represented using sets of (partially) weighted formulas [38, 44, 16]. For the sake of readability, we describe the colored graph construction for partially weighted *clauses*. Using a more involved transformation, however, we can extend it to sets of partially weighted formulas. Let $\mathcal{S} = \{(f_i, w_i)\}, 1 \leq i \leq n$, be a set of

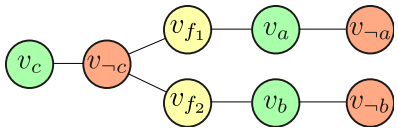


Figure 1: The colored graph resulting from the set of weighted clauses of Example 3.1.

partially weighted clauses with $w_i \in \mathbb{R}$ if f_i is weighted and $w_i = \infty$ otherwise. We define an *automorphism of* $\mathcal{S}$ as a permutation mapping (a) unnegated variables to unnegated variables, (b) negated variables to negated variables, and (c) clauses to clauses, respectively, such that this permutation maps $\mathcal{S}$ to an identical set of partially weighted clauses. The set of these permutations forms the automorphism group of $\mathcal{S}$.

The construction of the colored undirected graph $G(\mathcal{S}) = (V, E)$ proceeds as follows. For each variable a occurring in $\mathcal{S}$ we add two nodes v_a and $v_{\neg a}$ modeling the unnegated and negated variable, respectively, to V and the edge $\{v_a, v_{\neg a}\}$ to E. We assign color 0 (1) to nodes corresponding to negated (unnegated) variables. This coloring precludes permutations that map a negated variable to an unnegated one or vice versa. We introduce a distinct color c_∞ for unweighted clauses and a color c_w for each distinct weight w occurring in $\mathcal{S}$. For each clause f_i with weight $w_i = w$ we add a node v_{f_i} with color c_w to V. For each unweighted clause f_i we add a node v_{f_i} with color c_∞ to V. Finally, we add edges between each clause node v_{f_i} and the nodes of the negated and unnegated variables occurring in f_i. Please note that we can incorporate evidence by introducing two novel and distinct colors representing *true* and *false* variable nodes.

Example 3.1. Let $\{f_1 := (a \vee \neg c, 0.5), f_2 := (b \vee \neg c, 0.5)\}$ be a set of weighted clauses. We introduce 6 variable nodes $v_a, v_b, v_c, v_{\neg a}, v_{\neg b}, v_{\neg c}$ where the former three have color 1 (green) and the latter three color 0 (red). We connect the nodes v_a and $v_{\neg a}$; v_b and $v_{\neg b}$; and v_c and $v_{\neg c}$. We then introduce two new clause nodes v_{f_1}, v_{f_2} both with color 2 (yellow) since they have the same weight. We finally connect the variable nodes with the clause nodes they occur in. Figure 1 depicts the resulting colored graph. A generating set of $\mathsf{Aut}(G(\mathcal{S}))$, the automorphism group of this particular colored graph, is $\{(v_a\ v_b)(v_{\neg a}\ v_{\neg b})(v_{f_1}\ v_{f_2})\}$.

The following theorem states the relationship between the automorphisms of $\mathcal{S}$ and the colored graph $G(\mathcal{S})$.

Theorem 3.2. *Let* $\mathcal{S} = \{(f_i, w_i)\}, 1 \leq i \leq n$, *be a set of partially weighted clauses and let* $\mathsf{Aut}(G(\mathcal{S}))$ *be the automorphism group of the colored graph constructed for* $\mathcal{S}$. *There is a one-to-one correspondence between* $\mathsf{Aut}(G(\mathcal{S}))$ *and the automorphism group of* $\mathcal{S}$.

Given a set of partially weighted clauses $\mathcal{S}$ with variables $\mathbf{X}$ we have, by Theorem 3.2, that if we define a distribution Pr over random variables $\mathbf{X}$ with features f_i and weights w_i, $1 \leq i \leq n$, then $\mathsf{Aut}(G(\mathcal{S}))$ is an automorphism group for $(\{0,1\}^{\mathbf{X}}, \Pr)$. Hence, we can use the method to find symmetries in a large class of graphical models. The complexity of computing generating sets of $\mathsf{Aut}(G(\mathcal{S}))$ is in NP and not known to be in P or NP-complete. For graphs with bounded degree the problem is in P [25]. There are specialized algorithms for finding generating sets of automorphism groups of colored graphs such as Saucy[8] and Nauty[28] with remarkable performance. We will show that Saucy computes *irredundant* sets of generators of automorphism groups for graphical models with millions of variables. The size of these generating sets is bounded by the number of graph vertices.

We briefly position the symmetry detection approach in the context of existing algorithms and concepts.

3.1 Lifted Message Passing

There are two different lifted message passing algorithms. Lifted First-Order Belief Propagation [41] operates on Markov logic networks whereas Counting Belief Propagation [22] operates on factor graphs. Both approaches leverage symmetries in the model to partition variables and features into equivalence classes. Each variable class (supernode/clusternode) contains those variable nodes that would send and receive the same messages were (loopy) belief propagation (BP) run on the original model. Each feature class (superfeature/clusterfactor) contains factor nodes that would send and receive the same BP messages.

The colored graph construction provides an alternative approach to partitioning the variables and features of a graphical model. We simply compute the *orbit partition* induced by the permutation group $\mathsf{Aut}(G(\mathcal{S}))$ acting on the set of variables and features. For instance, the orbit partition of Example 3.1 is $\{\{a, b\}, \{c\}, \{f_1, f_2\}\}$. In general, orbit partitions have the following properties: For two variables v_1, v_2 in the same orbit we have that (a) v_1 and v_2 have identical marginal probabilities and (b) the variable nodes corresponding to v_1 and v_2 would send and receive the same messages were BP run on the original model; and for two features f_1 and f_2 in the same orbit we have that the factor nodes corresponding to f_1 and f_2 would send and receive the same BP messages.

3.2 Finite Partial Exchangeability

The notion of exchangeability was introduced by de Finetti [14]. Several theorems concerning finite (partial) exchangeability have been stated [10, 14]. Given

a finite sequence of n binary random variables $\mathbf{X}$, we say that $\mathbf{X}$ is *exchangeable* with respect to the distribution Pr if, for every $\mathbf{x} \in \{0,1\}^n$ and *every* permutation $\mathfrak{g}$ acting on $\{0,1\}^n$, we have that $\Pr(\mathbf{X} = \mathbf{x}) = \Pr(\mathbf{X} = \mathbf{x}^{\mathfrak{g}})$. This is equivalent to saying that the symmetric group $\mathsf{Sym}(n)$ is an automorphism group for $(\{0,1\}^n, \Pr)$. Whenever we have finite exchangeability, there are $n+1$ orbits each containing the variable assignments with Hamming weight $i, 0 \leq i \leq n$. Hence, every exchangeable probability distribution over n binary random variables is a unique mixture of draws from the $n+1$ orbits. In some cases of partial exchangeability, namely when the orbits can be specified using a *statistic*, one can use this for a more compact representation of the distribution as a product of mixtures [10]. The symmetries that have to be present for such a re-parameterization to be feasible, however, are rare and constitute one end of the symmetry spectrum.

Therefore, a central question is how *arbitrary symmetries*, compactly represented with irredundant generators of permutation groups, can be utilized for efficient probabilistic inference algorithms that go beyond (a) single variable marginal inference via lifted message passing and (b) the limited applicability of finite partial exchangeability. In order to answer this question, we turn to the major contribution of the present work.

4 Orbital Markov Chains

Inspired by the previous observations, we introduce *orbital Markov chains*, a novel family of Markov chains. An orbital Markov chain is always derived from an existing Markov chain so as to leverage the symmetries in the underlying model. In the presence of symmetries orbital Markov chains are able to perform wide-ranging transitions reducing the time until convergence. In the absence of symmetries they are equivalent to the original Markov chains. Orbital Markov chains only require a generating set of a permutation group $\mathfrak{G}$ acting on the chain's state space as additional input. As we have seen, these sets of generators are computable with graph automorphism algorithms.

Let Ω be a finite set, let $\mathcal{M}' = (X'_0, X'_1, ...)$ be a Markov chain with state space Ω, let π be a stationary distribution of $\mathcal{M}'$, and let $\mathfrak{G}$ be an automorphism group for (Ω, π). The *orbital Markov chain* $\mathcal{M} = (X_0, X_1, ...)$ *for* $\mathcal{M}'$ is a Markov chain which at each integer time $t+1$ performs the following steps:

1. Let X'_{t+1} be the state of the original Markov chain $\mathcal{M}'$ at time $t+1$;
2. Sample X_{t+1}, the state of the orbital Markov chain $\mathcal{M}$ at time $t+1$, uniformly at random from ${X'}_{t+1}^{\mathfrak{G}}$, the orbit of X'_{t+1}.

The orbital Markov chain $\mathcal{M}$, therefore, runs at every time step $t \geq 1$ the original chain $\mathcal{M}'$ first and samples the state of $\mathcal{M}$ at time t uniformly at random from the orbit of the state of the original chain $\mathcal{M}'$ at time t.

First, let us analyze the complexity of the second step which differs from the original Markov chain. Given a state X_t and a permutation group $\mathfrak{G}$ we need to sample an element from ${X_t}^{\mathfrak{G}}$, the orbit of X_t, uniformly at random. By the orbit-stabilizer theorem this is equivalent to sampling an element $\mathfrak{g} \in \mathfrak{G}$ uniformly at random and computing ${X_t}^{\mathfrak{g}}$. Sampling group elements nearly uniform at random is a well-researched problem [6] and computable in polynomial time in the size of the generating sets with product replacement algorithms [35]. These algorithms are implemented in several group algebra systems such as GAP[15] and exhibit remarkable performance. Once initialized, product replacement algorithms can generate pseudo-random elements by performing a small number of group multiplications. We could verify that the overhead of step 2 during the sampling process is indeed negligible.

Before we analyze the conditions under which orbital Markov chains are aperiodic, irreducible, and have the same stationary distribution as the original chain, we provide an example of an orbital Markov chain that is based on the standard Gibbs sampler which is commonly used to perform probabilistic inference.

Example 4.1. Let $\mathbf{V}$ be a finite set of random variables with probability distribution π, and let $\mathfrak{G}$ be an automorphism group for $(\times_{V \in \mathbf{V}} V, \pi)$. The orbital Markov chain for the *Gibbs sampler* is a Markov chain $\mathcal{M} = (X_0, X_1, ...)$ which, being in state X_t, performs the following steps at time $t+1$:

1. Select a variable $V \in \mathbf{V}$ uniformly at random;
2. Sample $X'_{t+1}(V)$, the value of V in the configuration X'_{t+1}, according to the conditional π-distribution of V given that all other variables take their values according to X_t;
3. Let $X'_{t+1}(W) = X_t(W)$ for all variables $W \in \mathbf{V} \setminus \{V\}$; and
4. Sample X_{t+1} from ${X'}_{t+1}^{\mathfrak{G}}$, the orbit of X'_{t+1}, uniformly at random.

We call this Markov chain the *orbital Gibbs sampler*. In the absence of symmetries, that is, if $\mathfrak{G}$'s only element is the identity permutation, the orbital Gibbs sampler is equivalent to the standard Gibbs sampler.

Let us now state a major result of this paper. It relates properties of the orbital Markov chain to those of the Markov chain it is derived from. A detailed proof can be found in the appendix.

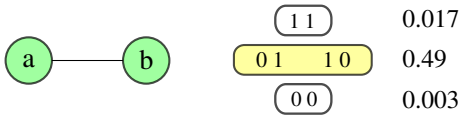


Figure 2: An undirected graphical model over two binary variables and one symmetric potential function with corresponding distribution shown on the right. There are three orbits each indicated by one of the rounded rectangles.

Theorem 4.2. *Let Ω be a finite set and let $\mathcal{M}'$ be a Markov chain with state space Ω and transition matrix P'. Moreover, let π be a probability distribution on Ω, let $\mathfrak{G}$ be an automorphism group for (Ω, π), and let $\mathcal{M}$ be the orbital Markov chain for $\mathcal{M}'$. Then,*

(a) if $\mathcal{M}'$ is aperiodic then $\mathcal{M}$ is also aperiodic;

(b) if $\mathcal{M}'$ is irreducible then $\mathcal{M}$ is also irreducible;

(c) if π is a reversible distribution for $\mathcal{M}'$ and, for all $\mathfrak{g} \in \mathfrak{G}$ and all $x, y \in \Omega$ we have that $P'(x, y) = P'(x^{\mathfrak{g}}, y^{\mathfrak{g}})$, then π is also a reversible and, hence, a stationary distribution for $\mathcal{M}$.

The condition in statement (c) requiring for all $\mathfrak{g} \in \mathfrak{G}$ and all $x, y \in \Omega$ that $P'(x, y) = P'(x^{\mathfrak{g}}, y^{\mathfrak{g}})$ conveys that the original Markov chain is compatible with the symmetries captured by the permutation group $\mathfrak{G}$. This rather weak assumption is met by all of the practical Markov chains we are aware of and, in particular, Metropolis chains and the standard Gibbs sampler.

Corollary 4.3. *Let $\mathcal{M}'$ be the Markov chain of the Gibbs sampler with reversible distribution π. The orbital Gibbs sampler for $\mathcal{M}'$ is aperiodic and has π as a reversible and, hence, a stationary distribution. Moreover, if $\mathcal{M}'$ is irreducible then the orbital Gibbs sampler is also irreducible and it has π as its unique stationary distribution.*

We will show both analytically and empirically that, in the presence of symmetries, the orbital Gibbs sampler converges at least as fast or faster to the true distribution than state of the art sampling algorithms. First, however, we want to take a look at an example that illustrates the advantages of the orbital Gibbs sampler.

Example 4.4. Consider the undirected graphical model in Figure 2 with two binary random variables and a symmetric potential function. The probabilities of the states 01 and 10 are both 0.49. Due to the symmetry in the model, the states 10 and 01 are part of the same orbit. Now, let us assume a standard Gibbs sampler is in state 10. The probability for it to transition to one of the states 11 and 00 is only 0.02 and, by definition of the standard Gibbs sampler, it cannot transition directly to the state 01. The chain is "stuck" in the state 10 until it is able to move to 11 or 00. Now, consider the orbital Gibbs sampler. Intuitively, while it is "waiting" to move to one of the low probability states, it samples the two high probability states horizontally uniformly at random from the orbit $\{01, 10\}$. In this particular case the orbital Gibbs sampler converges faster than the standard Gibbs sampler, a fact that we will also show analytically.

4.1 Mixing Time of Orbital Markov Chains

We will make our intuition about the faster convergence of the orbital Gibbs sampler more concrete. We accomplish this by showing that the more symmetry there is in the model the faster a coupling of the orbital Markov chain will coalesce and, therefore, the faster the chain will converge to its stationary distribution.

There are several methods available to prove rapid mixing of a finite Markov chain. The method we will use here is that of a *coupling*. A coupling for a Markov chain $\mathcal{M}$ is a stochastic process (X_t, Y_t) on $\Omega \times \Omega$ such that (X_t) and (Y_t) considered marginally are *faithful copies* of $\mathcal{M}$. The *coupling lemma* expresses that the total variation distance of $\mathcal{M}$ at time t is limited from above by the probability that the two chains have not *coalesced*, that is, have not met at time t (see for instance Aldous [1]). Coupling proofs on the joint space $\Omega \times \Omega$ are often rather involved and require complex combinatorial arguments. A possible simplification is provided by the *path coupling* method where a coupling is only required to hold on a subset of $\Omega \times \Omega$ (Bubley and Dyer [4]). The following theorem formalizes this idea.

Theorem 4.5 (Dyer and Greenhill [12])**.** *Let δ be an integer valued metric defined on $\Omega \times \Omega$ taking values in $\{0, ..., D\}$. Let $S \subseteq \Omega \times \Omega$ such that for all $(X_t, Y_t) \in \Omega \times \Omega$ there exists a path $X_t = Z_0, ..., Z_r = Y_t$ between X_t and Y_t with $(Z_l, Z_{l+1}) \in S$ for $0 \leq l \leq r$ and*

$$\sum_{l=0}^{r-1} \delta(Z_l, Z_{l+1}) = \delta(X_t, Y_t).$$

Define a coupling $(X, Y) \to (X', Y')$ of the Markov chain $\mathcal{M}$ on all pairs $(X, Y) \in S$. Suppose there exists $\beta \leq 1$ with $\mathbf{E}[\delta(X', Y')] \leq \beta\delta(X, Y)$ for all $(X, Y) \in S$. If $\beta < 1$ then the mixing time $\tau(\varepsilon)$ of $\mathcal{M}$ satisfies

$$\tau(\varepsilon) \leq \frac{\ln(D\varepsilon^{-1})}{1 - \beta}.$$

If $\beta = 1$ and there exists an $\alpha > 0$ such that $\Pr[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] \geq \alpha$ for all t, then

$$\tau(\varepsilon) \leq \left\lceil \frac{eD^2}{\alpha} \right\rceil \left\lceil \ln(\varepsilon^{-1}) \right\rceil.$$

We selected the *insert/delete* Markov chain for independent sets of graphs for our analysis. Sampling independent sets is a classical problem motivated by numerous applications and with a considerable amount of recent research devoted to it [24, 13, 45, 42, 37]. The coupling proof for the orbital version of this Markov chain provides interesting insights into the construction of such a coupling and the influence of the graph symmetries on the mixing time. The proof strategy is in essence applicable to other sampling algorithms.

Let $G = (V, E)$ be a graph. A subset X of V is an *independent set* if $\{v, w\} \notin E$ for all $v, w \in X$. Let $\mathcal{I}(G)$ be the set of all independent sets in a given graph G and let λ be a positive real number. The partition function $Z = Z(\lambda)$ and the corresponding probability measure π_λ on $\mathcal{I}(G)$ are defined by

$$Z = Z(\lambda) = \sum_{X \in \mathcal{I}(G)} \lambda^{|X|} \quad \text{and} \quad \pi_\lambda(X) = \frac{\lambda^{|X|}}{Z}.$$

Approximating the partition function and sampling from $\mathcal{I}(G)$ can be accomplished using a rapidly mixing Markov chain with state space $\mathcal{I}(G)$ and stationary distribution π_λ. The simplest Markov chain for independent sets is the so-called *insert/delete* chain [13]. If X_t is the state at time t then the state at time $t+1$ is determined by the following procedure:

1. Select a vertex $v \in V$ uniformly at random;
2. If $v \in X_t$ then let $X_{t+1} = X_t \backslash \{v\}$ with probability $1/(1+\lambda)$;
3. If $v \notin X_t$ and v has no neighbors in X_t then let $X_{t+1} = X_t \cup \{v\}$ with probability $\lambda/(1+\lambda)$;
4. Otherwise let $X_{t+1} = X_t$.

Using a path coupling argument one can show that the *insert/delete* chain is rapidly mixing for $\lambda \leq 1/(\Delta - 1)$ where Δ is the maximum degree of the graph [13]. We can turn the *insert/delete* Markov chain into the orbital *insert/delete* Markov chain $\mathcal{M}(\mathcal{I}(G))$ simply by adding the following fifth step:

5. Sample X_{t+1} uniformly at random from its orbit.

By Corollary 4.3 the orbital *insert/delete* chain for independent sets is aperiodic, irreducible, and has π_λ as its unique stationary distribution. We can now state the following theorem concerning the mixing time of this Markov chain. It relates the graph symmetries to the mixing time of the chain. The proof of the theorem is based on a path coupling that is constructed so as to make the two chains coalesce whenever their respective states are located in the same orbit. A detailed and instructive proof can be found in the appendix.

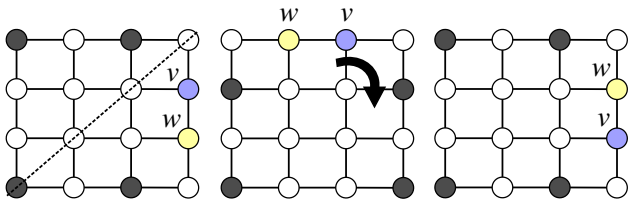


Figure 3: Two independent sets $X \cup \{v\}$ and $X \cup \{w\}$ of the 4x4 grid located in the same orbit. The first permutation is the reflection on the sketched diagonal the second a clockwise 90° rotation.

Theorem 4.6. *Let $G = (V, E)$ be a graph with maximum degree Δ, let λ be a positive real number, and let $\mathfrak{G}$ be an automorphism group for $(\{0,1\}^V, \pi_\lambda)$. Moreover, let $(X \cup \{v\}) \in \mathcal{I}(G)$, let $(X \cup \{w\}) \in \mathcal{I}(G)$, let $\{v, w\} \in E$, and let $\rho = \Pr[(X \cup \{v\}) \notin (X \cup \{w\})^{\mathfrak{G}}]$. The orbital* insert/delete *chain $\mathcal{M}(\mathcal{I}(G))$ is rapidly mixing if either $\rho \leq 0.5$ or $\lambda \leq 1/((2\rho - 1)\Delta - 1)$.*

The theorem establishes the important link between the graph automorphisms and the mixing time of the orbital *insert/delete* chain. The more symmetries the graph exhibits the larger the orbits and the sooner the chains coalesce. Figure 3 depicts the 4x4 grid with two independent sets $X \cup \{v\}$ and $X \cup \{w\}$ with $\{v, w\} \in E$ and $(X \cup \{v\}) \in (X \cup \{w\})^{\mathfrak{G}}$. Since $\rho < 1$ for nxn grids, $n \geq 4$, we can prove (a) rapid mixing of the orbital *insert/delete* chain for larger λ values and (b) more rapid mixing for identical λ values.

The next corollary follows from Theorem 4.6 and the simple fact that $X' \in X^{\mathfrak{G}}$ for all $X \subseteq V, X' \subseteq V$ with $|X| = |X'|$ whenever $\mathfrak{G}$ is the symmetric group on V.

Corollary 4.7. *Let $G = (V, E)$ be a graph, let λ be a positive real number, and let $\mathfrak{G}$ be an automorphism group for $(\{0,1\}^V, \pi_\lambda)$. If $\mathfrak{G}$ is the symmetric group $\mathsf{Sym}(V)$ then $\mathcal{M}(\mathcal{I}(G))$ is rapidly mixing with $\tau(\varepsilon) \leq |V| \ln(|V|\varepsilon^{-1})$.*

By analyzing the coupling proof of Theorem 4.6 and, in particular, the moves leading to states (X', Y') with $|X'| = |Y'|$ and the probability that X' and Y' are located in the same orbit in these cases, it is possible to provide more refined bounds. Moreover, to capture the full power of orbital Markov chains, a coupling argument should not merely consider pairs of states with Hamming distance 1. Indeed, the strength of the orbital chains is that, in the presence of symmetries in the graph topology, there is a non-zero probability that states with large Hamming distance (up to $|V|$) are located in the same orbit. The method presented here is also applicable to Markov chains known to mix rapidly for larger λ values than the *insert/delete* chain such as the *insert/delete/drag* chain [13].

social network model [41]					
people	20	50	100	250	500
vertices	1740	10200	40700	251750	1003500
edges	2120	10350	50600	314000	1253000
time [s]	0.04	0.15	0.81	22.5	261.3
features	860	5150	20300	125750	501500
orbs w/o	7	7	7	7	7
orbs w/	238	1244	6237	30192	78303
kxk grid model					
k	20	50	100	250	500
vertices	800	5000	20000	125000	500000
edges	1160	7400	29800	187000	749000
time [s]	0.02	0.03	0.2	0.6	2.5

Table 1: Number of vertices and edges of the colored graphs, the runtime of SAUCY, and the number of (super-)features of the social network model without and with 10% evidence.

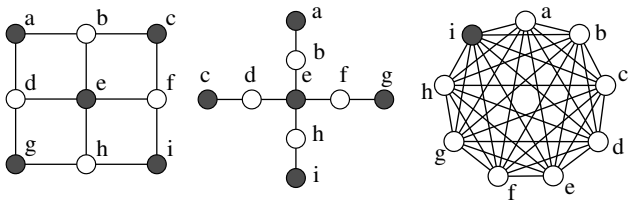


Figure 4: From left to right: the 3-*grid*, the 3-*connected cliques*, and the 3-*complete graph* models.

5 Experiments

Two graphical models were used to evaluate the symmetry detection approach. The "Friends & Smokers" Markov logic network where for a random 10% of all people it is known (a) whether they smoke or not and (b) who 10 of their friends are [41]. Moreover, we used the kxk grid model, an established and well-motivated lattice model with numerous applications [37]. All experiments were conducted on a PC with an AMD Athlon dual core 5400B 1.0 GHz processor and 3 GB RAM. Table 1 lists the results for varying model sizes. SAUCY's runtime scales roughly quadratic with the number of vertices and it performs better for the kxk grid models. This might be due to the larger sets of generators for the permutation groups of the social network model. Table 1 also lists the number of features of the ground social network model (features), the number of feature orbits without (orbs w/o) and with (orbs w/) 10% evidence.

We proceeded to compare the performance of the orbital Markov chains with state-of-the-art algorithms for sampling independent sets. We used GAP[15], a system for computational discrete algebra, and the ORB package[31][1] to implement the sampling algorithms. The experiments can easily be replicated by installing GAP and the ORB package and by running the GAP files available at a dedicated code repository[2]. For the evaluation of the sampling algorithms we selected three different graph topologies exhibiting varying degrees of symmetry:

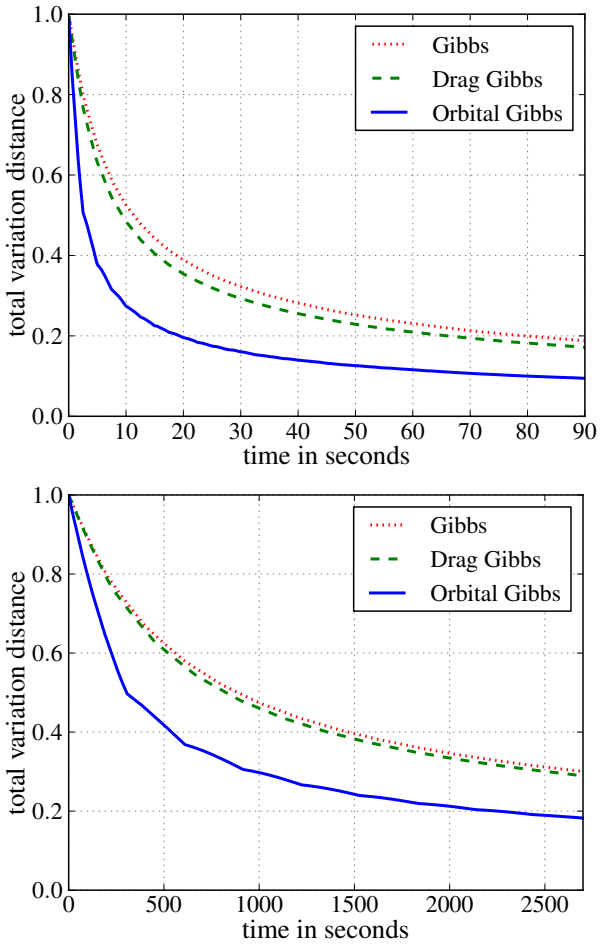


Figure 5: The results of the three Gibbs samplers for the 5-*grid model* (top) and the 6-*grid model* (bottom).

The k-*grid model* is the 2-dimensional kxk grid. An instance of the model for $k = 3$ is depicted in Figure 4 (left). Here, the generating set of the permutation group $\mathfrak{G}$ computed by SAUCY is $\{(a\ c)(d\ f)(g\ i), (a\ i)(b\ f)(d\ h)\}$ and $|\mathfrak{G}| = 8$. The permutation group $\mathfrak{G}$ partitions the set $\{0,1\}^9$ in 102 orbits with each orbit having a cardinality in $\{1, 2, 4, 8\}$.

The k-*connected cliques model* is a graph with $k+1$ distinct cliques each of size $k-1$ and each connected with one edge to the same vertex. Statistical relational formalisms such as Markov logic networks often lead to similar graph topologies. An in-

[1] http://www.gap-system.org/Packages/orb.html

[2] http://code.google.com/p/lifted-mcmc/

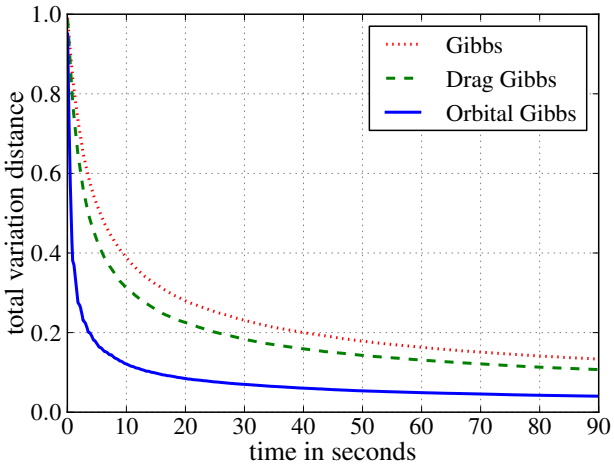


Figure 6: The results of the three Gibbs samplers for the 5-*connected cliques* model.

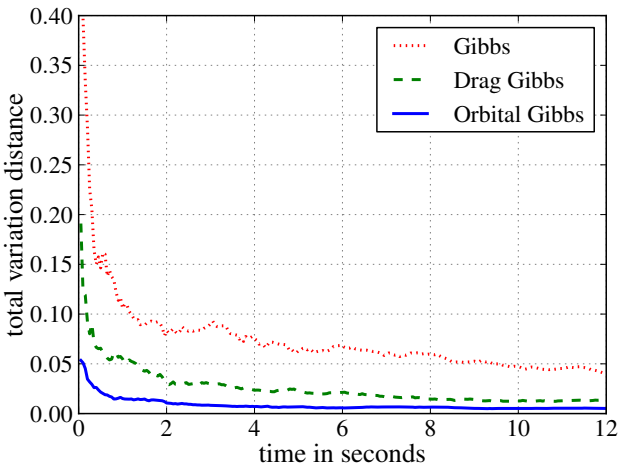


Figure 7: The results of the three Gibbs samplers for the 5-*complete graph* model.

stance for $k = 3$ is depicted in Figure 4 (center). Here, the generating set of $\mathfrak{G}$ computed by SAUCY is $\{(a\ g)(b\ f), (a\ c)(b\ d), (a\ i)(b\ h)\}$ and $|\mathfrak{G}| = 24$. The permutation group $\mathfrak{G}$ partitions the set $\{0,1\}^9$ in 70 orbits with cardinalities in $\{1, 4, 6, 12, 24\}$.

The *k-complete graph model* is a complete graph with k^2 vertices. Figure 4 (right) depicts an instance for $k = 3$. Here, the generating set of $\mathfrak{G}$ computed by SAUCY is $\{(b\ c), (b\ d), (b\ e), (b\ f), (b\ g), (b\ h), (b\ i), (a\ b)\}$ and $|\mathfrak{G}| = 9! = 362880$. The permutation group $\mathfrak{G}$ partitions the set $\{0,1\}^9$ in 10 orbits with each orbit having a cardinality in $\{1, 9, 36, 84, 126\}$.

SAUCY needed at most 5 ms to compute the sets of generators for the permutation groups of the three models for $k = 6$. We generated samples of the probability measure π_λ on $\mathcal{I}(G)$ for $\lambda = 1$ and the three different graph topologies by running (a) the *insert/delete* chain, (b) the *insert/delete/drag* chain [13], and (c) the orbital *insert/delete* chain. Each chain was started in the state corresponding to the empty set and no burn-in period was used. The orbital *insert/delete* chain did not require more RAM and needed 50 microseconds per sample which amounts to an overhead of about 25% relative to the 40 microseconds of the *insert/delete* chain. The 25% overhead remained constant and independent of the size of the graphs. Since the sampling algorithms create large files with all accumulated samples, I/O overhead is included in these times. For each of the three topologies and each of the three Gibbs samplers, we computed the total variation distance between the distribution approximated using *all* accumulated samples and the true distribution π_1. Figure 5 plots the total variation distance over elapsed time for the *k-grid model* for $k = 5$ and $k = 6$. The orbital *insert/delete* chain (Orbital Gibbs) converges the fastest. The *insert/delete/drag* chain (Drag Gibbs) converges faster than the *insert/delete* chain (Gibbs) but since there is a small computational overhead of the *insert/delete/drag* chain the difference is less pronounced for $k = 6$. The same results are observable for the other graph topologies (see Figures 6 and 7) where the orbital Markov chain outperforms the others. In summary, the larger the cardinalities of the orbits induced by the symmetries the faster converges the orbital Gibbs sampler relative to the other chains.

6 Discussion

The mindful reader might have recognized a similarity to *lumping* of Markov chains which amounts to partitioning the state space of the chain [5]. Computing the coarsest lumping quotient of a Markov chain with a bisimulation procedure is linear in the *number of nonzero probability transitions* of the chain and, hence, in most cases exponential in the number of random variables. Since merely counting equivalence classes in the Pólya theory setting is a #P-complete problem [18] there are clear computational limitations to this approach. Orbital Markov chains, on the other hand, combine the advantages of a compact representation of symmetries as generating sets of permutation groups with highly efficient product replacement algorithms and, therefore, provide the advantages of *lumping* while avoiding the intractable explicit computation of the partition of the state space.

One can apply orbital Markov chains to other graphical models that exhibit symmetries such as the Ising model. Since Markov chains in general and Gibbs samplers in particular are components in numerous algo-

rithms (cf. [43, 20, 38, 19, 23, 3]), we expect orbital Markov chains to improve the algorithms' performance when applied to problems that exhibit symmetries. For instance, sampling algorithms for statistical relational languages are obvious candidates for improvement. Future work will include the integration of orbital Markov chains with algorithms for marginal as well as maximum a-posteriori inference. We will also apply the symmetry detection approach to make existing inference algorithms more efficient by, for instance, using symmetry breaking constraints in combinatorial optimization approaches to maximum a-posteriori inference in Markov logic networks (cf. [39, 32, 33]).

While we have shown that permutation groups are computable with graph automorphism algorithms for a large class of models it is also possible to *assume* certain symmetries in the model in the same way (conditional) independencies are assumed in the design stage of a probabilistic graphical model. Orbital Markov chains could easily incorporate these symmetries in form of permutation groups.

Acknowledgments

Many thanks to Guy Van den Broeck, Kristian Kersting, Martin Mladenov, and Babak Ahmadi for insightful discussions concerning lifted inference, to Jürgen Müller for helpful remarks on the product replacement algorithm, and to all those who have contributed to the GAP system and the ORB package.

References

[1] D. Aldous. Random walks on finite groups and rapidly mixing Markov chains. In *Seminaire de Probabilites XVII*, pages 243–297. 1983.

[2] F. A. Aloul, K. A. Sakallah, and I. L. Markov. Efficient symmetry breaking for boolean satisfiability. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 271–276, 2003.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[4] R. Bubley and M. Dyer. Path coupling: A technique for proving rapid mixing in markov chains. In *Proceedings of the 38th Symposium on Foundations of Computer Science*, pages 223–231, 1997.

[5] P. Buchholz. Exact and ordinary lumpability in finite markov chains. *Journal of Applied Probability*, 31(1):pp. 59–75, 1994.

[6] F. Celler, C. R. Leedham-Green, S. H. Murray, A. C. Niemeyer, and E. O'brien. Generating random elements of a finite group. *Communications in Algebra*, 23(13):4931–4948, 1995.

[7] J. Crawford. A theoretical analysis of reasoning by symmetry in first-order logic. In *Proceedings of the Workshop on Tractable Reasoning*, 1992.

[8] P. T. Darga, K. A. Sakallah, and I. L. Markov. Faster symmetry discovery using sparsity of symmetries. In *Proceedings of the 45th annual Design Automation Conference*, pages 149–154, 2008.

[9] R. de Salvo Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1319–1325, 2005.

[10] P. Diaconis and D. Freedman. De finetti's generalizations of exchangeability. In *Studies in Inductive Logic and Probability*, volume II. 1980.

[11] C. Drescher, O. Tifrea, and T. Walsh. Symmetry-breaking answer set solving. *AI Commun.*, 24(2):177–194, 2011.

[12] M. Dyer and C. Greenhill. A more rapidly mixing markov chain for graph colorings. *Random Struct. Algorithms*, 13(3-4):285–317, 1998.

[13] M. Dyer and C. Greenhill. On markov chains for independent sets. *Journal of Algorithms*, 35(1):17–49, 2000.

[14] B. Finetti. *Probability, induction and statistics: the art of guessing*. Probability and mathematical statistics. Wiley, 1972.

[15] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4.12*, 2008.

[16] V. Gogate and P. Domingos. Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 256–265, 2011.

[17] V. Gogate, A. Jha, and D. Venugopal. Advances in lifted importance sampling. In *Proceedings of the 26th Conference on Artificial Intelligence*, 2012.

[18] L. A. Goldberg. Computation in permutation groups: counting and randomly sampling orbits. In *Surveys in Combinatorics*, pages 109–143. Cambridge University Press, 2001.

[19] C. Gomes, J. Hoffmann, A. Sabharwal, and B. Selman. From sampling to model counting. In *Proceedings of the 20th International Joint*

Conference on Artificial Intelligence, pages 2293–2299, 2007.

[20] O. Häggström. *Finite Markov Chains and Algorithmic Applications*. London Mathematical Society Student Texts. Cambridge University Press, 2002.

[21] M. Jaeger. Lower complexity bounds for lifted inference. *CoRR*, abs/1204.3255, 2012.

[22] K. Kersting, B. Ahmadi, and S. Natarajan. Counting belief propagation. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 277–284, 2009.

[23] P. Liang, M. I. Jordan, and B. Taskar. A permutation-augmented sampler for dp mixture models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 545–552, 2007.

[24] M. Luby and E. Vigoda. Fast convergence of the glauber dynamics for sampling independent sets. *Random Struct. Algorithms*, 15(3-4):229–241, 1999.

[25] E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Computer and System Sciences*, 25(1):42–65, 1982.

[26] F. Margot. Exploiting orbits in symmetric ilp. *Math. Program.*, 98(1-3):3–21, 2003.

[27] F. Margot. Symmetry in integer linear programming. In *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer-Verlag, 2010.

[28] B. D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[29] B. Milch and S. J. Russell. General-purpose mcmc inference over relational structures. In *Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence*, pages 349–358, 2006.

[30] M. Mladenov, B. Ahmadi, and K. Kersting. Lifted linear programming. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, 2012.

[31] J. Müller, M. Neunhöffer, and R. Wilson. Enumerating big orbits and an application: b acting on the cosets of fi_{23}. *Journal of Algebra*, 314(1):75–96, 2007.

[32] M. Niepert. A delayed column generation strategy for exact k-bounded map inference in markov logic networks. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 384–391, 2010.

[33] M. Niepert, J. Noessner, and H. Stuckenschmidt. Log-linear description logics. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 2153–2158, 2011.

[34] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Math. Program.*, 126(1):147–178, 2011.

[35] I. Pak. The product replacement algorithm is polynomial. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 476–485, 2000.

[36] D. Poole. First-order probabilistic inference. In *Proceedings of the 18th Joint Conference on Artificial Intelligence*, pages 985–991, 2003.

[37] R. Restrepo, J. Shin, P. Tetali, E. Vigoda, and L. Yang. Improved mixing condition on the grid for counting and sampling independent sets. In *Proceedings of the 52nd Symposium on Foundations of Computer Science*, pages 140–149, 2011.

[38] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[39] S. Riedel. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 468–475, 2008.

[40] P. Sen, A. Deshpande, and L. Getoor. Bisimulation-based approximate lifted inference. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

[41] P. Singla and P. Domingos. Lifted first-order belief propagation. In *Proceedings of the 23rd Conference on Artificial Intelligence*, pages 1094–1099, 2008.

[42] A. Sly. Computational transition at the uniqueness threshold. In *Proceedings of the 51st Symposium on Foundations of Computer Science*, pages 287–296, 2010.

[43] M. Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society Series B*, 62(4):795–809, 2000.

[44] G. Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Advances in Neural Information Processing Systems*, pages 1386–1394, 2011.

[45] D. Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the 38th ACM Symposium on Theory of computing*, pages 140–149, 2006.

Local Structure Discovery in Bayesian Networks

Teppo Niinimäki
Helsinki Institute for Information Technology HIIT
Department of Computer Science
University of Helsinki, Finland
teppo.niinimaki@cs.helsinki.fi

Pekka Parviainen
CSC and Scilifelab
Royal Institute of Technology (KTH)
Stockholm, Sweden
pekkapa@kth.se

Abstract

Learning a Bayesian network structure from data is an NP-hard problem and thus exact algorithms are feasible only for small data sets. Therefore, network structures for larger networks are usually learned with various heuristics. Another approach to scaling up the structure learning is local learning. In local learning, the modeler has one or more target variables that are of special interest; he wants to learn the structure near the target variables and is not interested in the rest of the variables. In this paper, we present a score-based local learning algorithm called SLL. We conjecture that our algorithm is theoretically sound in the sense that it is optimal in the limit of large sample size. Empirical results suggest that SLL is competitive when compared to the constraint-based HITON algorithm. We also study the prospects of constructing the network structure for the whole node set based on local results by presenting two algorithms and comparing them to several heuristics.

1 INTRODUCTION

A Bayesian network is a representation of a joint probability distribution. It consists of two parts: the structure and the parameters. The structure of a Bayesian network is represented by a directed acyclic graph (DAG) that expresses the conditional independence relations among variables and the parameters determine the local conditional distributions for each variable.

Structure learning in Bayesian networks, that is, finding a DAG that fits to the data, has drawn lots of interest in recent years. There are two rather distinct approaches to structure learning. In score-based approach (Cooper and Herskovits, 1992; Heckerman et al., 1995) each DAG gets a score based on how well it fits to the data and the goal is to find the DAG that maximizes the score. On the other hand, the constraint-based approach (Pearl, 2000; Spirtes et al., 2000) is based on testing conditional independencies among variables and constructing a DAG that represents these relations. Both approaches are compatible in the sense that (under some general assumptions), no matter which approach one uses, exact algorithms converge to the same DAG when the number of samples tends to infinity.

Structure learning in Bayesian networks is NP-hard (Chickering, 1996; Chickering et al., 2004). Due to the NP-hardness of the problem, it is unlikely that there are exact algorithms that run in polynomial time. Indeed, the fastest exact algorithms (Silander and Myllymäki, 2006) run in exponential time. Although there have been several attempts to scale up the exact algorithms (de Campos et al., 2009; Parviainen and Koivisto, 2009; Jaakkola et al., 2010; Cussens, 2011; Malone et al., 2011), these methods quickly become infeasible with larger datasets.

As the exact algorithms do not work for large data sets, one often has to resort to heuristics. A common strategy is to use different variants of greedy search. An alternative approach to the problem is local learning. When one has a large data set, often it is a case that all variables are not equally interesting. One might have one or a handful of *target variables* and the goal is to learn the structure only in the vicinity of the targets. An example for a problem where local learning is feasible is prediction. If we know the Markov blanket of a target variable, that is, its parents, its children and the parents of its children, the remaining variables do not give any more information. Thus, if it is known that we are going to predict the values of only a handful of variables, we can just learn their Markov blankets and ignore the structure of the rest of the network.

Local learning is not a new approach. There have been several studies especially on constraint-based lo-

cal learning (Nägele et al., 2007; Aliferis et al., 2010a,b). These studies have shown that local learning can be a powerful tool in practice. In this paper we study the prospects of the score-based local learning of Bayesian network structures. We present a *score-based local learning algorithm (SLL)* that is a variant of the generalized local learning (GLL) framework (Aliferis et al., 2010a). Assuming a consistent scoring criterion is used, we conjecture that our algorithm is theoretically sound in the same sense as greedy equivalence search (GES) (Chickering, 2002) and HITON (Aliferis et al., 2010a), that is, it is guaranteed to find the true Markov blanket of the target node when the sample size tends to infinity.

Optimality guarantees in the limit or lack thereof do not tell anything about results with finite sample size. Thus, we conducted experiments and compared our algorithm to other heuristics. Based on the experiments, we found our algorithm promising.

2 PRELIMINARIES

2.1 BAYESIAN NETWORKS

The structure of a Bayesian network is represented by a *directed acyclic graph (DAG)*. Formally, a DAG is a pair (N, A) where N is the node set and A is the arc set. If there is an arc from u to v, that is, $uv \in A$, we say that u is a *parent* of v and v is a *child* of u. If v is either a parent or a child of u, they are said to be *neighbors*. Further, if there is a directed path from u to v in A then v is a *descendant* of u. Nodes v and u are are said to be *spouses* of each other if they have a common child and there is no arc between v and u. We denote the set of the parents of a node v in A by A_v. Further, we use $H^A(v)$ and $S^A(v)$ to denote the sets of the neighbors and spouses of v in A, respectively. When the node set is clear from the context, we identify a DAG by its arc set A. The cardinality of N is denoted by n.

Each node v corresponds to a random variable and the DAG expresses conditional independence assumptions between variables. Random variables u and v are said to be conditionally independent in distribution p given a set S of random variables if $p(u, v|S) = p(u|S)p(v|S)$. A DAG *contains* or *represents* a joint distribution of the random variables if the joint distribution satisfies the *local Markov condition*, that is, every variable is conditionally independent of its non-descendants given its parents. Such a distribution can be specified using local *conditional probability distributions (CPD)* which specify the distribution of a random variable given its parents A_v. CPDs are usually taken from a parameterized class of probability distributions, like discrete or Gaussian distributions. Thus, the CPD of variable v is determined by its parameters θ_v; the type and the number of parameters is specified by the particular class of probability distributions. The parameters of a Bayesian network are denoted by θ which consists of the parameters of each CPD. Finally, a Bayesian network is a pair (A, θ).

A distribution p is said to be *faithful* to a DAG A if all conditional independencies in p are implied by A. We say that a DAG A^p is a *perfect map* of a distribution p if A^p contains p and p is faithful to A^p. By $p[Z]$ we denote the marginal distribution of p on set Z.

The conditional independencies implied by a DAG can be extracted using a d-separation criterion; this is equivalent to local Markov condition. The *skeleton* of a DAG A is an undirected graph that is obtained by replacing all directed arcs $uv \in A$ with undirected edges between u and v. A *path* in a DAG is a cycle-free sequence of edges in the corresponding skeleton. A node v is a *head-to-head node* along a path if there are two consecutive arcs uv and wv on that path. Nodes v and u are *d-connected* by nodes Z along a path from v to u if every head-to-head node along the path is in Z or has a descendant in Z and none of the other nodes along the path is in Z. Nodes v and u are *d-separated* by nodes Z if they are not d-connected by Z along any path from v to u. If A^p is a perfect map of p then v and u are conditionally independent give Z in p if and only if v and u are d-separated by Z in A^p. If v and u are conditionally independent in p given Z we use notation $v \perp\!\!\!\perp_p u|Z$.

The *Markov blanket* of node v is the smallest node set S such that v is conditionally independent of all other nodes given S. The Markov blanket of node v consists of the parents of v, the children of v, and the spouses of v.

Nodes s, t, and u form a *v-structure* in a DAG if s and t are spouses and u is their common child. A v-structure is denoted by (s, u, t). Two DAGs are said to be *Markov equivalent* if they can contain the same set of distributions, or equivalently, imply the same set of conditional independence statements. It can be shown that two DAGs are Markov equivalent if and only if they have the same skeleton and same v-structures (Verma and Pearl, 1990).

2.2 CONSISTENT SCORING CRITERION

Score-based structure learning methods in Bayesian networks assign a score to each DAG based on how well the DAG fits to the data according to some statistical principle. A function $f(A, D)$ that assigns the score to a DAG A based on the data D is called a *scoring criterion*. A scoring criterion is *decomposable* if the score of a DAG is a sum of local scores that only depend

on a node and its parents. A local score function is denoted by $f(v, A_v, D_v, D_{A_v})$, where D_v and D_{A_v} are the data on a node v and a node set A_v, respectively. Now, a decomposable score for a DAG A can be written as

$$f(A, D) = \sum_{v \in N} f(v, A_v, D_v, D_{A_v}).$$

Let the data D consists of m independent and identically distributed (i.i.d.) samples from a distribution p on N. A scoring criterion f is said to be *consistent* if in the limit as m grows, the following two properties hold:

1. If A' contains p and A does not, then $f(A', D) > f(A, D)$.
2. If A' and A both contain p, and A' has fewer parameters, then $f(A', D) > f(A, D)$.

A related and in our case more useful property is local consistency. Let A be any DAG and A' be the DAG that results from adding the arc uv to A. A scoring criterion f is said to be *locally consistent* if in the limit as m grows, the following two properties hold for all u and v:

1. If $u \not\perp\!\!\!\perp_p v | A_v$, then $f(A', D) > f(A, D)$.
2. If $u \perp\!\!\!\perp_p v | A_v$, then $f(A', D) < f(A, D)$.

Intuitively, adding an arc that eliminates an independence constraint that does not hold in the data-generating distribution increases the score, and adding an arc that does not eliminate such a constraint decreases the score.

A scoring criterion is said to be *score equivalent* if two Markov equivalent DAGs always have the same score. With a score equivalent scoring criterion one can learn a structure of a Bayesian network up to an equivalence class but cannot distinguish the DAGs inside the class.

For example, commonly used BDeu score is locally consistent (Chickering, 2002) and score equivalent (Heckerman et al., 1995).

2.3 STRUCTURE DISCOVERY PROBLEM

The problem of structure discovery in Bayesian networks is to find, given data D, a DAG that in some sense is the best representation of the data. We call this global learning. In score-based approach the goodness of a DAG is measured by the score $f(A, D)$. In constraint-based approach one would like to find a DAG that is a perfect map of the data-generating distribution.

In the local learning problem the output is the neighbor and/or spouse sets for a target node, not a DAG. The goodness of the result can be measured by comparing learned neighbor and spouse sets to the corresponding sets in the A^p, the perfect map of the data-generating distribution. Note that this comparison can be made only if A^p is known.

3 LOCAL LEARNING ALGORITHM

In this section, we present a score-based local learning algorithm algorithm (SLL) that finds the Markov blanket of a given target node. SLL works in two phases: First, one learns the neighbors of the target and then the rest of the Markov blanket. SLL is a score-based variant of the constraint-based local learning algorithm by Aliferis et al. (2010a,b); the main difference between SLL and the algorithm by Aliferis et al. is that they use independence tests to identify the neighbors and spouses whereas we recognize them from optimal Bayesian networks. In this section, we also analyze the behavior of the algorithm in the limit. We also consider constructing a Bayesian network for the whole node set using the local structures that have been found.

3.1 FINDING PARENTS AND CHILDREN

We start by learning the potential neighbors of a target node. The idea of the algorithm is as follows. The input of the algorithm consists of the data D on the node set N and a target node $t \in N$. During the execution of the algorithm, we update two sets: Set O consists of the nodes that have not been analyzed yet and set H consists of nodes that are currently considered as potential neighbors of t. In the beginning, set H is empty and O contains all nodes but t. After the initialization, nodes in O are considered one by one: One learns an optimal DAG on the target, the current set of its potential neighbors and the new node under consideration. For the structure learning we use a subroutine OptimalNetwork which returns the highest scoring DAG on a given node set. The subroutine can use the dynamic programming algorithm of Silander and Myllymäki (2006) or any other exact algorithm. After an optimal DAG is found, the nodes that are the neighbors of t in that particular DAG form a new set of potential neighbors and the rest of the nodes are discarded. After all nodes are either discarded or in the set of potential neighbors, the set of the potential neighbors is returned.

The pseudocode of the above procedure is shown in Algorithm 1.

Next, we will show that if the data was generated from

Algorithm 1 FindPotentialNeighbors

Input: Data D on node set N, a target node $t \in N$.
Output: The potential neighbors of t.

Initialization: $O \leftarrow N \setminus \{t\}$, $H(t) \leftarrow \emptyset$
while O is nonempty **do**
  Choose $v \in O$
  $O \leftarrow O \setminus \{v\}$
  $Z \leftarrow \{t, v\} \cup H(t)$
  $A \leftarrow \text{OptimalNetwork}(Z, D_Z)$
  $H(t) \leftarrow H^A(t)$
end while
return $H(t)$

a distribution p, Algorithm 1 is guaranteed to find all correct neighbors, that is, all nodes that are neighbors of t in A^p will be included in $H(t)$ in the limit. The guarantee holds under the following assumptions.

Assumption 1. *The data D consists of i.i.d. samples from a distribution p that is faithful to a DAG A^p.*

Assumption 2. *The procedure* OptimalNetwork *uses a locally consistent and score equivalent scoring criterion.*

Lemma 3. *Let $H^{A^p}(t)$ be the neighbors of the target t in A^p. When Assumptions 1 and 2 hold, Algorithm 1 will return a set $H(t)$ such that $H^{A^p}(t) \subseteq H(t)$.*

Proof. If nodes t and v are not conditionally independent in p given any set $X \subseteq N \setminus \{t, v\}$, then any Bayesian network that contains p must have an arc between t and v. Thus, if we have two networks A and A' on $Y \cup \{t, v\}$ such that A' has an arc between t and v and A has not and otherwise they are the same, then by the local consistency, A' has higher score than A. Thus, the highest scoring network must have an arc between t and v. As the above reasoning applies to every set $Y \subseteq N \setminus \{t, v\}$, v will be in $H(t)$ when Algorithm 1 stops. □

Based on the previous lemma we know that if a node v is a neighbor of a node t in A^p, then it will be in $H(t)$. However, we have no guarantees that no nodes that are non-adjacent to t in A^p are included in $H(t)$. Indeed, Aliferis et al. (2010a) point out that if we have a perfect map A^p described in Figure 1 an extra node might be added to $H(t)$. To see this, assume that t is our target and its true neighbors u and s are in $H(t)$ and w has been discarded. Then $t \not\perp\!\!\!\perp_p v|X$ for all $X \subseteq \{u, s\}$ and thus, by local consistency, adding an arc between t and v always increases the score and therefore it is always included in the optimal DAG.

The neighbor relation is symmetric: if v is a child of t then t is a parent of v. This allows us to try to remove

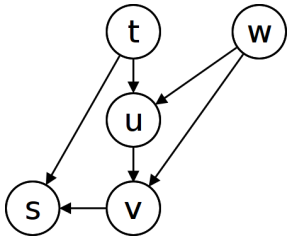


Figure 1: A DAG where non-adjacent nodes t and v are not conditionally independent given any subset of the neighbors of t.

extra nodes from $H(t)$ using a simple symmetry correction in similar fashion as Aliferis et al. (2010a): t and v are neighbors in A^p only if both v is a potential neighbor of t and t is a potential neighbor of v. Algorithm 2 uses symmetry correction to find the neighbors of a target node.

Algorithm 2 FindNeighbors

Input: Data D on node set N, a target node $t \in N$.
Output: The neighbors of t.

$H^*(t) \leftarrow \text{FindPotentialNeighbors}(D, t)$
for all $v \in H^*(t)$ **do**
  $H(v) \leftarrow \text{FindPotentialNeighbors}(D, v)$
  if $t \notin H(v)$ **then**
    $H^*(t) \leftarrow H^*(t) \setminus \{v\}$
  end if
end for
return $H^*(t)$

To analyze the optimality of the algorithm, we use the below lemma which shows that if t and v are not neighbors in A^p but they are dependent given every subset of the neighbors of t then they are conditionally independent given some subset of the neighbors of v. Formally, let $E^{A^p}(t)$ be the set consisting of the nodes $v \in N \setminus \{t\}$ such that $t \not\perp\!\!\!\perp_p v|X$ for all $X \subseteq H^{A^p}(t)$.

Lemma 4 (Aliferis et al. (2010a), Lemma 2)**.** *If $v \in E^{A^p}(t) \setminus H^{A^p}(t)$, then $t \notin E^{A^p}(v) \setminus H^{A^p}(v)$.*

However, it is not clear whether the output of Algorithm 1 is a subset of the the nodes that are conditionally dependent on t given any subset of the neighbors of t. As we are not aware of any counterexamples, we conjecture the follows.

Conjecture 5. *Let $H(t)$ be the output of Algorithm 1. Then $H(t) \subseteq E^{A^p}(t)$.*

The next lemma shows the optimality of Algorithm 2 in the limit assuming Conjecture 5 holds.

Lemma 6. *Let $H^{A^p}(t)$ be the neighbors of the target t in A^p. When Assumptions 1 and 2 and Conjecture 5*

hold, Algorithm 2 will return a set $H^*(t)$ such that $H^*(t) = H^{A^p}(t)$.

Proof. First, let us prove that $H^{A^p}(t) \subseteq H^*(t)$. By Lemma 3, $H^{A^p}(t) \subseteq H(t)$ and $H^{A^p}(v) \subseteq H(v)$ for all v. Thus, if $v \in H^{A^p}(t)$ then $v \in H(t)$ and $t \in H(v)$.

Let us prove that $H^*(t) \subseteq H^{A^p}(t)$. By Conjecture 5 we observe that $H(t) \subseteq E^{A^p}(t)$. Suppose that we have nodes $t, v \in N$ such that $v \in H(t)$ and $t \perp\!\!\!\perp_p v|X$ for some $X \subseteq N \setminus \{t, v\}$. Therefore, it must be that $v \in E^{A^p}(t)$ and $v \notin H^{A^p}(t)$. Thus, $v \in E^{A^p}(t) \setminus H^{A^p}(t)$. By Lemma 4, we have that $t \notin E^{A^p}(v) \setminus H^{A^p}(v)$. By symmetry, $t \notin H^{A^p}(v)$ and thus $t \notin E^{A^p}(v)$. Therefore, $t \notin H(v)$ and Algorithm 2 does not include t and v as neighbors of each other.

□

3.2 FINDING THE MARKOV BLANKET

Learning the spouses of the target t is quite similar to the learning the neighbors of t. In addition to the data D and target node t we take as input the set $H^*(t)$, learned by Algorithm 2, consisting of the neighbors of t. The set $S(t)$ will consist the potential spouses of t. As all potential spouses are neighbors of a neighbor of t, we need to go through only a subset of all nodes. The set of neighbors $H^*(t)$ is fixed and we keep updating the set $S(t)$. All nodes that have a common child with t and are not neighbors of t are kept in and the rest are discarded. Again, the nodes that remain in $S(t)$ once all nodes have been considered, are considered potential spouses of t.

The procedure is summarized in Algorithm 3.

Algorithm 3 FINDPOTENTIALSPOUSES

Input: Data D on node set N, a target node $t \in N$, neighbors of the target $H^*(t)$.
Output: Potential spouses of t.

Find set H': the neighbors of the nodes in $H^*(t)$.
Initialization: $O \leftarrow H' \setminus (H^*(t) \cup \{t\})$, $S(t) \leftarrow \emptyset$
while O is nonempty **do**
  Choose $v \in O$
  $O \leftarrow O \setminus \{v\}$
  $Z \leftarrow \{t, v\} \cup H^*(t) \cup S(t)$
  $A \leftarrow$ OPTIMALNETWORK(Z, D_Z)
  $S(t) \leftarrow S^A(t)$
end while
return $S(t)$

The following lemma shows that the set $S(t)$ will not contain any nodes that are not spouses of the target t in A^p.

Lemma 7. *Let $S^{A^p}(t)$ be the spouses of the target t in A^p. When Assumptions 1 and 2 and Conjecture 5 hold, Algorithm 3 will return a set $S(t)$ such that $S(t) \subseteq S^{A^p}(t)$.*

Proof. Let t be the target, u its neighbor and v a neighbor of u. A spouse v is added to $S(t)$ only when there is a v-structure (t, u, v) in A.

Suppose that there is no v-structure (t, u, v) in A^p. Then $t \not\perp\!\!\!\perp_p v|X$ for all $X \subseteq N \setminus \{t, u, v\}$. Now, if a DAG A has the v-structure (t, u, v) then A_t does not contain u and thus $t \not\perp\!\!\!\perp_p v|A_t$. This means that, by local consistency, adding an arc vt to A increases the score. Symmetrically, A_v does not contain u and by local consistency, adding an arc tv increase the score. Since it is always possible to add at least one of these arcs without introducing a cycle, A cannot be the optimal graph.

□

Lemma 7 does not guarantee that all spouses are found (in the limit) using Algorithm 3. Indeed, Figure 1 shows an example where Algorithm 3 leaves one spouse out. Let t be our target. Then, its neighbors are s and u. Consider learning an optimal DAG on $\{s, t, u, v\}$. We notice that t and v are not conditionally independent given any subset of $\{s, u\}$ and thus by local consistency the optimal network must contain an arc between t and v. Therefore, v cannot be part of a v-structure and is discarded. However, in the original graph v is involved in a v-structure (t, s, v). To find spouses, we use Algorithm 4.

Algorithm 4 FINDSPOUSES

Input: Data D on node set N, a target node $t \in N$, neighbors of the target $H^*(t)$, neighbors of the neighbors of the target $H^*(v)$ for all $v \in H^*(t)$.
Output: The spouses of t.

$S^*(t) \leftarrow$ FINDPOTENTIALSPOUSES$(D, t, H^*(t))$
for all $v \in N \setminus (\{t\} \cup H^*(t))$ **do**
  $S(v) \leftarrow$ FINDPOTENTIALSPOUSES$(D, v, H^*(v))$
  if $t \in S(v)$ **then**
    $S^*(t) \leftarrow S^*(t) \cup \{v\}$
  end if
end for
return $S^*(t)$

It is not known whether Algorithm 4 is guaranteed to find all spouses in the limit. Finding a counterexample seems difficult and thus we conjecture as follows.

Conjecture 8. *Let $S^{A^p}(t)$ be the spouses of the target t in A^p. When Assumptions 1 and 2 hold, Algorithm 4 will return a set $S^*(t)$ such that $S^*(t) = S^{A^p}(t)$.*

Once we have found the neighbors $H^*(t)$ and spouses $S^*(t)$ of a target node t, the Markov blanket is simply

$H^*(t) \cup S^*(t)$. The following conjecture that holds if Conjectures 5 and 8 hold summarizes the theoretical guarantees.

Conjecture 9. *When Assumptions 1 and 2 hold, the local learning algorithm is optimal in the limit, that is, when the sample size m approaches infinity the algorithm always finds the correct Markov blanket for every target.*

3.3 TIME AND SPACE REQUIREMENT

In both Algorithm 1 and 3, the while loop is executed at most $n-1$ times. The time and space requirement inside the loop is dominated by the procedure OptimalNetwork. On a node set Z it runs in $O(|Z|^2 2^{|Z|})$ time and $O(|Z|2^{|Z|})$ space, where in the worst case $|Z| = O(n)$. Thus, these algorithms have a worst case time requirement $O(n^3 2^n)$ and space requirement $O(n2^n)$. In practice, however, the networks are often relatively sparse and the running times are significantly lower than in the worst case; see the experiments. Algorithms 2 and 4 call Algorithms 1 and 3 at most n times, respectively. Thus, the total time requirement is at most $O(n^4 2^n)$.

The above algorithms were presented for computing the Markov blanket for a single target. If one computes Markov blankets for all nodes, one can store and reuse the the potential neighbor and spouse sets. Thus, in the worst case the Markov blanket for all nodes can be found in $O(n^4 2^n)$ time.

3.4 FROM LOCAL TO GLOBAL

Above we have learned Markov blankets of target nodes. Next, we introduce two methods to construct a DAG on the whole node set based on the local results.

The first method uses a constraint-based approach. As mentioned in the previous section, when computing the local neighbor sets we do not need to do the symmetry correction separately for each node. Instead, we can first find the potential neighbors for each node and then get the actual neighbors and build the skeleton using AND-rule; an edge between u and v is added to the skeleton only if both u and v are potential neighbors of each other. Similar way we can also skip the separate symmetry checks when finding the spouses of each node. As a result, we can use the procedure SLL+C described in Algorithm 5 to construct a DAG.

On lines 1–4 of the Algorithm 5 the skeleton E is built and on lines 5–19 the v-structures are directed; this specifies the Markov equivalence class of the DAG. To direct the rest of the arcs which is done on line 20 we can use the rules listed in Pearl (2000). Note that in practice there may be conflicts between v-structures.

Algorithm 5 SLL+C

Input: Data D on node set N.
Output: Directed acyclic graph A.

```
1: for all v ∈ N do
2:    H(v) ← FindPotentialNeighbors(D, v)
3: end for
4: E ← {{u, v} | v ∈ H(u) and u ∈ H(v)}
5: for all v ∈ N do
6:    H*(v) ← {u | {v, u} ∈ E}
7:    S(v) ← FindPotentialSpouses(D, v, H*(v))
8: end for
9: A ← ∅
10: for all v ∈ N do
11:    for all u ∈ S(v) do
12:       for all w is a common child of v and u do
13:          if possible without introducing cycles then
14:             remove {v, w} and {u, w} from E
15:             add vw and uw to A
16:          end if
17:       end for
18:    end for
19: end for
20: direct the rest of the edges without introducing
    cycles or (if possible) additional v-structures
21: return A
```

With a finite sample size, one v-structure could, for example, force an arc to be oriented from u to v and another v-structure from v to u.

Another way to construct a DAG from the local results is to use a heuristic such as greedy search with some constraints imposed by the local neighbors. As we will see later in the experiments section, this often leads to a structure with better score compared to the constraint-based approach. Another advantage is that there is no need to find the spouses of the nodes, which is often the more computationally intensive phase in SLL. Algorithm 6 describes SSL+G procedure which uses OR-rule to build a skeleton E of potential edges from the local neighbor sets. Then it calls a greedy search as a subroutine, which may only add an arc if the corresponding edge is present in the skeleton. This approach is similar to the MMHC algorithm by Tsamardinos et al. (2006a) and comes with no correctness guarantees in the limit.

4 EXPERIMENTS

We implemented the SLL algorithm as well as both SLL+C and SLL+G algorithms in C++. As OptimalNetwork subroutine we used the dynamic programming algorithm by Silander and Myllymäki (2006) with fallback to greedy equivalence search (GES) (Chickering, 2002) if the number of nodes in the input is larger

Algorithm 6 SLL+G

Input: Data D on node set N.
Output: Directed acyclic graph A.

```
for all v ∈ N do
    H(v) ← FindPotentialNeighbors(D, v)
end for
E ← {{u, v} | v ∈ H(u) or u ∈ H(v)}
A ← GreedySearch(D, E)
return A
```

than 20. In the dynamic programming we limited the maximum in-degree of a node to 5. The implementation is available at `http://www.cs.helsinki.fi/u/tzniinim/uai2012/`. These algorithms were compared against several state-of-the-art algorithms, namely the constraint-based HITON (Aliferis et al., 2010a) for local structure discovery and greedy search (GS), greedy equivalence search (GES) and the max-min hill-climbing (MMHC) (Tsamardinos et al., 2006a) for global structure discovery. The association test used by HITON and MMHC was the built in Assoc function of the implementation, that is, G^2 test according to Aliferis et al. (2010a) and Tsamardinos et al. (2006a). For HITON the maximum conditioning set size was set to 5. BDeu prior with equivalent sample size 1 was used in all algorithms when applicable.

In our experiments we used discrete data generated from real world Bayesian networks. We chose a subset of (smaller) data sets which were used by Tsamardinos et al. (2006a) and are freely available online[1]. These are listed in Table 1. Some of the networks have been generated by tiling 3, 5 or 10 copies of the original network using a method by Tsamardinos et al. (2006b). For each network we had 10 independently generated random datasets containing 500, 1000 and 5000 i.i.d. samples.

To accommodate to limited space we chose a representative sample of four of the networks for which we have figures: Alarm5, Child10, Insurance5, and Hailfinder. The figures for the rest of the networks are available as an online appendix.

4.1 LOCAL LEARNING

In the local learning experiments the goal was to learn both the neighbor sets and the Markov blankets for all nodes. We compared SLL to the constraint-based HITON from Causal Explorer toolkit[2] by Aliferis et al. (2011).

For both the neighbor sets and the Markov blankets we measured the local Hamming distance between the learned set of nodes and the true set of nodes. The Hamming distance of two sets A and B is the number of elements which are contained in only one of the sets, that is, $|A \setminus B| + |B \setminus A|$. For each data set we computed the Sum of Local Hamming Distances over all possible target nodes, call it SLHD.

Network	Num vars	max in/out -degree	domain size
Alarm	37	4 / 5	2–4
Alarm3	111	4 / 5	2–4
Alarm5	185	4 / 6	2–4
Barley	48	4 / 5	2–67
Child	20	2 / 7	2–6
Child3	60	3 / 7	2–6
Child5	100	2 / 7	2–6
Child10	200	2 / 7	2–6
Hailfinder	56	4 / 16	2–11
Insurance	27	3 / 7	2–5
Insurance3	81	4 / 7	2–5
Insurance5	135	5 / 8	2–5

Table 1: Bayesian networks used in the experiments. Information from Tsamardinos et al. (2006a).

Figure 2 shows the average SLHDs as a function of the size of the data set for learning the neighbor nodes. As expected, the accuracy of the learned neighbor set improves as the number of data samples increases. In most of the cases SLL beats HITON. The average SLHDs for learning the Markov blankets are shown in Figure 3. The results are similar to those for neighbor nodes.

4.2 GLOBAL LEARNING

In the global learning experiments the goal was to learn the entire structure (DAG) of the underlying Bayesian network. We considered both SLL+C and SLL+G heuristics. As the greedy search algorithm we used the steepest ascent hill-climbing with a TABU list of the last 100 structures and a stopping criterion of 15 steps without improvement in the maximum score. These parameters were chosen to be same as in the MMHC and greedy search implementation by Tsamardinos et al. (2006a). We compared our algorithms to greedy search (GS) and max-min hill-climbing (MMHC) from Causal Explorer toolkit as well as greedy equivalent search (GES) from TETRAD software[3].

Some of the algorithms return a DAG but the others return a PDAG, a partially directed acyclic graph which

[1] `http://www.dsl-lab.org/supplements/mmhc_paper/mmhc_index.html`

[2] `http://www.dsl-lab.org/causal_explorer/`

[3] `http://www.phil.cmu.edu/projects/tetrad/`

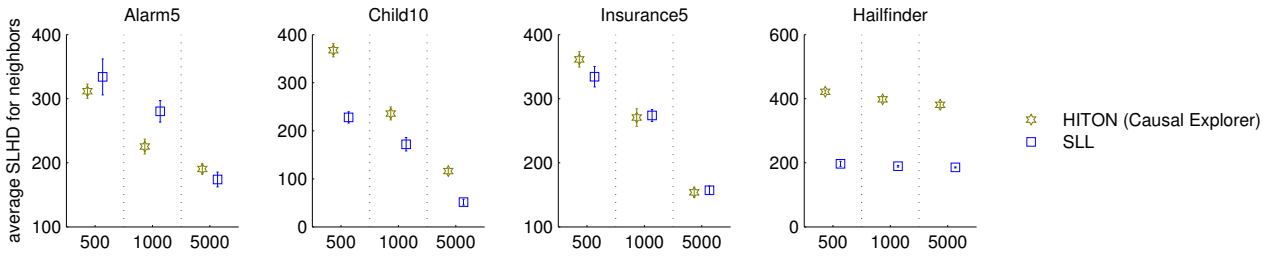


Figure 2: Average SLHD (Sum of Local Hamming Distances) between the returned neighbor sets (parents and children) and the true neighbor sets for different data sizes (500, 1000 and 5000 samples). Standard deviations are shown as vertical bars.

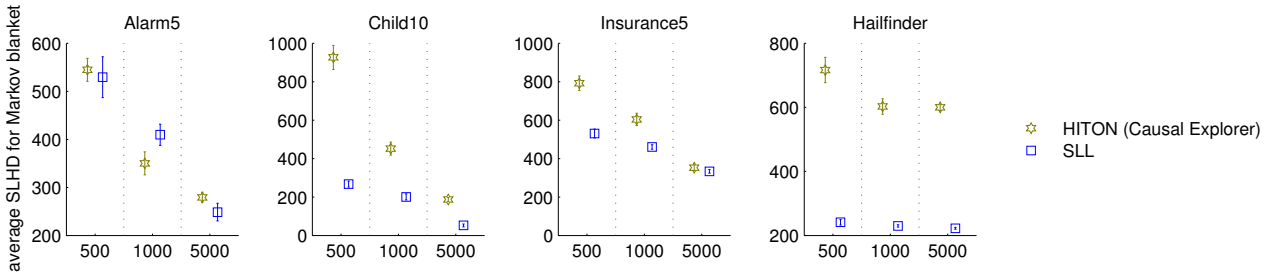


Figure 3: Average SLHD between the returned Markov blankets and the true Markov blankets.

corresponds to a Markov equivalence class. Compared to a DAG, in the corresponding PDAG the edges which are not oriented the same way in all members of the class are undirected. Since the DAGs belonging to a same equivalence class cannot be distinguished using score equivalent scores, we converted all returned DAGs to PDAGs and compared the PDAGs. If the algorithm failed (i.e. did not return a valid DAG) the run was ignored. This only affected GES, which failed to return an acyclic graph in several runs.

We used two measures the evaluate the goodness of the PDAGs returned by the algorithms: normalized BDeu score and Structural Hamming Distance from the true structure. The normalized BDeu score was obtained by computing the BDeu score with equivalent sample size 1 for a DAG extension of the PDAG and dividing the result by the score of the true structure; the lower the normalized score, the better the structure fits to the data. Average normalized scores are shown in Figure 4. The basic greedy search seems to do surprisingly well, even compared to the GES. For some networks SLL+G works better than MMHC, for other it is the other way around. SLL+C loses to the other methods in most of the cases.

The Structural Hamming Distance (SHD) between two PDAGs is defined as the number of edges which are missing/extra or of wrong type (reversed or directed in one PDAG and undirected in the other). We measured the distances between the resulting PDAGs and the true structures. Figure 5 shows the average SHDs. Compared to the observations for scores, here SLL+C performs clearly better and a basic greedy search much worse.

4.3 TIME CONSUMPTION

Figure 6 shows the average running times. For local learning the times include learning both the neighbors and the Markov blankets for all nodes. As in SLL+C the most of the time is spent in local learning part which is the same as running the SLL algorithm for all nodes, these times are combined. The times for HITON are not directly comparable to SLL since it does the symmetry correction for each node separately. In spite of this extra work, HITON is often not any slower than SLL. Also for global learning the rival algorithms are usually faster than SLL+G and SLL+C.

5 DISCUSSION

In this paper we have studied prospects of score-based local learning. We have conjectured that the score-based local learning provides same the theoretical guarantees as greedy equivalence search (GES) (Chickering, 2002) and constraint-based local learning (Aliferis et al., 2010a). A natural avenue for future research is to prove these guarantees or lack thereof.

Our experiments suggest that our method provides

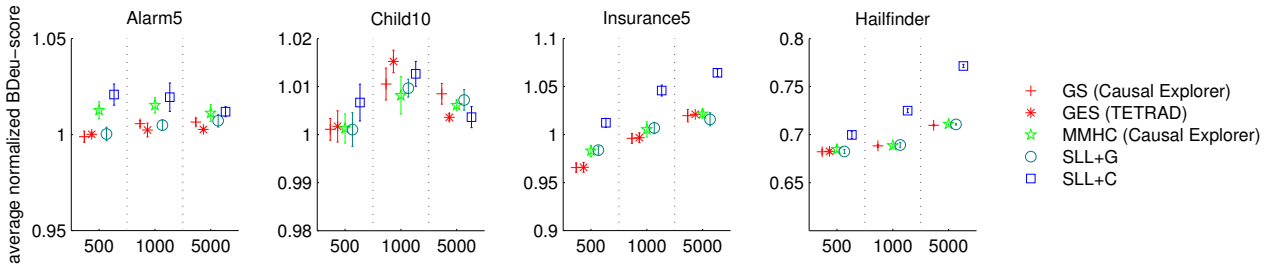


Figure 4: Average normalized score of the returned network structures. Smaller is better and 1.0 corresponds to the true network.

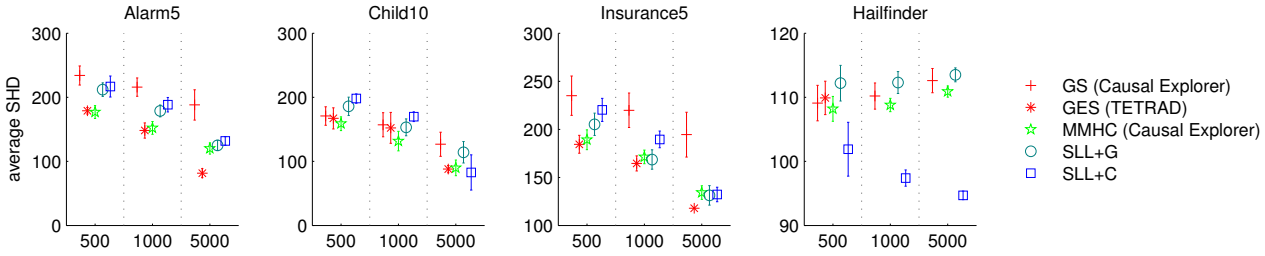


Figure 5: Average SHD (Structural Hamming Distance) between the returned structures and the true structure.

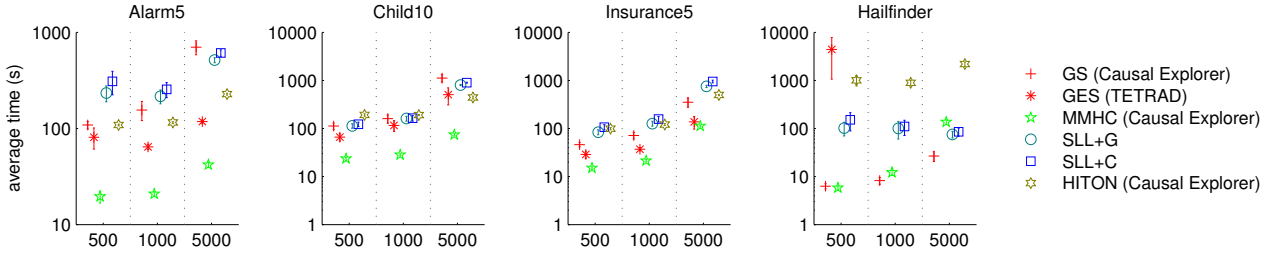


Figure 6: Average running times for the algorithms.

a competitive alternative for constraint-based local learning. Our algorithm seems to often find local neighborhoods more accurately than the competitor. However, as a downside our algorithm usually consumes significantly more time. We hope that our algorithm could bridge the gap between exact algorithms and constraint-based local learning by providing a way to trade off between accuracy and scalability. The local-to-global approach seem to perform less well compared to various other heuristics. However, in some data sets local-to-global algorithms outperformed all benchmarks in structural Hamming distance especially when there was lots of data. This suggests that with further development, score-based local-to-global heuristic could be competitive in certain types of networks.

References

Constantin F. Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D. Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification part I: Algorithms and empirical evaluation. *Journal of Machine Learning Research*, 11:171–234, 2010a.

Constantin F. Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D. Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification part II: Analysis and extensions. *Journal of Machine Learning Research*, 11:235–284, 2010b.

Constantin F. Aliferis, Ioannis Tsamardinos, and Alexander Statnikov. Causal explorer: A probabilistic network learning toolkit for discovery. Software, 2011. URL `http://discover.mc.vanderbilt.edu/discover/public/causal_explorer/`.

David Maxwell Chickering. Learning Bayesian networks is NP-Complete. In Doug Fisher and Hans-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics*, pages 121–130. Springer-Verlag, 1996.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Reseach*, 3:507–554, 2002.

David Maxwell Chickering, David Heckerman, and Chris Meek. Large-sample learning of Bayesian networks is NP-Hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

James Cussens. Bayesian network learning with cutting planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 153–160. AUAI Press, 2011.

Cassio P. de Campos, Zhi Zeng, and Qiang Ji. Structure learning of Bayesian networks using constraints. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 113–120. Omnipress, 2009.

David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, JMLR: W&CP, pages 358–365, 2010.

Brandon Malone, Changhe Yuan, Eric A. Hansen, and Susan Bridges. Improving the scalability of optimal Bayesian network learning with external-memory frontier breadth-first branch and bound search. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 479–488. AUAI Press, 2011.

Andreas Nägele, Mathäus Dejori, and Martin Stetter. Bayesian substructure learning - approximate learning of very large network structures. In *European Conference on Machine Learning (ECML)*, pages 238–249, 2007.

Pekka Parviainen and Mikko Koivisto. Exact structure discovery in Bayesian networks with less space. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 436–443. AUAI Press, 2009.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge university Press, 2000.

Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the Twenty-Second Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 445–452. AUAI Press, 2006.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer Verlag, 2000.

Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78, 2006a.

Ioannis Tsamardinos, Alexander Statnikov, Laura E. Brown, and Constantin F. Aliferis. Generating realistic large Bayesian networks by tiling. In *The 19th International FLAIRS Conference*, pages 592–597, 2006b.

Thomas S. Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 255–270. Elsevier, 1990.

Hilbert Space Embeddings of POMDPs

Yu Nishiyama[1] **Abdeslam Boularias**[2] **Arthur Gretton**[2,3] **Kenji Fukumizu**[1]
[1]The Institute of Statistical Mathematics, {nishiyam,fukumizu}@ism.ac.jp
[2]Max Planck Institute for Intelligent Systems, boularias@tuebingen.mpg.de
[3]Gatsby Computational Neuroscience Unit, CSML, UCL, arthur.gretton@gmail.com

Abstract

A nonparametric approach for policy learning for POMDPs is proposed. The approach represents distributions over the states, observations, and actions as embeddings in feature spaces, which are reproducing kernel Hilbert spaces. Distributions over states given the observations are obtained by applying the kernel Bayes' rule to these distribution embeddings. Policies and value functions are defined on the feature space over states, which leads to a feature space expression for the Bellman equation. Value iteration may then be used to estimate the optimal value function and associated policy. Experimental results confirm that the correct policy is learned using the feature space representation.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) are general models for sequential control problems in partially observable environments, in which an agent executes an action under uncertainty while reward delivery and state transitions vary depending on the state and action. The objective is to find an optimal policy that maximizes a value function defined on beliefs (distributions over states), and determined by the reward function. When the value is the expected sum of discounted rewards, the optimal policy and the optimal value function are computed by solving a Bellman equation. Solutions to the Bellman equation are generally very difficult to obtain, with a number of approaches being employed. These include tractable parametric models of the system (parametric POMDPs [Poupart et al., 2006, Even-dar, 2005]); Monte Carlo methods for more complex models (Monte Carlo POMDPs [Silver and Veness, 2010, Thrun, 2000]); and methods that take advantage of the piece-wise linear and convex (PWLC) property (PBVI [Pineau et al., 2003], Perseus [Spaan and Vlassis, 2005], HSVI [Smith and Simmons, 2004]), where these last approaches use the fact that value functions computed by finite-step value iteration algorithms are PWLC in the beliefs [Sondik, 1971, Porta. et al., 2006]. All these methods have drawbacks, however: parametric models can cause bias errors if they oversimplify, Monte Carlo sampling methods can be computationally costly, and PWLC-based methods require the observations and actions to be discrete.

Our nonparametric approach is based on a series of recent works in which probability distributions are represented as embeddings in reproducing kernel Hilbert spaces (RKHSs) [Fukumizu et al., 2008, Gretton et al., 2007, Gretton et al., 2012]. Probability distributions can be embedded as points in RKHSs, and expectations of RKHS functions may be obtained as inner products with these embeddings, using the kernel trick. For a sufficiently rich RKHS (i.e., a characteristic RKHS), every probability distribution has a unique embedding, and the distance between embeddings is a metric on distributions [Fukumizu et al., 2008, Sriperumbudur et al., 2010]. Recently, inference methods which make use of the embeddings have been proposed, including for Hidden Markov Models [Song et al., 2009, Song et al., 2010], and belief propagation [Song et al., 2010, Song et al., 2011]. More recently, the embeddings have been used in performing value iteration and optimal policy learning in Markov decision processes (MDPs) [Grunewalder et al., 2012]; and the Kernel Bayes' Rule (KBR) [Fukumizu et al., 2011] was proposed, which updates a prior distribution embedding via feature space operations to obtain a posterior embedding.

In this paper, we propose kernel POMDPs (kPOMDPs), a new approach to solving the Bellman

equations and determining policy, based on a nonparametric model defined in appropriate RKHSs. All probability distributions required for the algorithm are represented as embeddings in RKHSs, including the beliefs over the states, the transition models, the observation models, and the predictive distributions over subsequent states given actions, observations, and current beliefs. These embeddings are updated sequentially based on actions and observations. Likewise, value functions are defined over feature representations of states, and policies map RKHS representations of states to actions, which leads to an expression for the Bellman equation in feature space.

We use the feature representation of the Bellman equations to define a value iteration algorithm for POMDPs, which directly estimates both the expected immediate rewards and expected values of posterior beliefs in feature space. As in the earlier work cited above, expectations are represented by inner products in respective state and observation feature spaces, and can be efficiently computed using the kernel trick. While the original Bellman operator has contractive and isotonic properties, i.e., the original value iteration is guaranteed to converge monotonically [Porta. et al., 2006], the resulting kernel Bellman operator does not. That said, these properties can be enforced following a simple correction, as proposed by [Grunewalder et al., 2012]. Approaches from the classical POMDP literature for initializing and enhancing efficiency can be applied, including ways to set initial values over the state feature space, and pruning methods for action edges.

Since kPOMDPs are nonparametric, the embeddings of the distributions used in the algorithm are learned from training samples. Note that in training, we must have access to samples from the hidden state, although for the test phase no such observations are necessary. This setting is reasonable in cases where hidden states are relatively costly to obtain: while it might be possible to observe them initially when learning the system dynamics, we would not have access to them during the value iteration phase, when learning an optimal policy.

Advantages of kPOMDPs include

- kPOMDPs can handle complex distributions over states and observations, since kPOMDPs are nonparametric.
- kPOMDPs can handle a wide variety of data types, as RKHS kernels have been defined on many domains: these include discrete, continuous, and structured data.
- kPOMDPs can be applied to high-dimensional POMDPs, since the computation scales with training sample size, and not with state and observation dimensionality. The convergence of the distribution embedding is at a rate independent of the dimension of the underlying space [Fukumizu et al., 2011, Gretton et al., 2012].

Our experiments confirm that the proposed kernel value iteration algorithm learns the correct policy on POMDP benchmark tasks.

This paper is organized as follows. We introduce the notations for POMDPs in the next section, and review recent kernel methods for probability distributions in Section 3. We present kernel POMDPs in Section 4, where Bellman equations in feature spaces (Subsection 4.1), the empirical expression and value iteration algorithms (Subsection 4.2) are shown. Experiments follow for an online planning algorithm.

2 POMDPs

A POMDP is specified by a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z \rangle$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$[1] is the transition function where $T(s, a, s') = \Pr(s'|s, a)$ represents the distribution of the next state s' given an action a in a state s, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function given an action a in a state s, $\mathcal{O}$ is the set of observations, $Z : \mathcal{S} \times \mathcal{O} \to [0, \infty)$[2] is the observation function where $Z(s, o) = \Pr(o|s)$ represents the distribution of observation o given a state s. We assume that R is bounded.

An agent executes an action a under the setting that the true state s is not known, but partial information o can be observed according to $Z(s, o)$. The agent then transitions to its next state s' according to $T(s, a, s')$, receiving a reward $R(s, a)$ and making a new observation o' according to $Z(s', o')$. The agent executes then its next action a', and the process is repeated.

The goal of reinforcement learning in a POMDP is, given an initial belief b_0 for the initial state and a history of actions and observations $h_{t+1} = \{a_0, o_1, ..., a_t, o_{t+1}\}$, to find an optimal policy $\pi^*_{t+1} : (b_0, h_{t+1}) \mapsto a_{t+1}$ that maximizes the value function of the expected sum of discounted rewards with infinite horizon $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t]$, where $\gamma \in (0, 1)$ is a discount factor.

In POMDPs, all the information (b_0, h_{t+1}) is condensed in a sufficient statistic of a belief distribution b over the state set $\mathcal{S}$, and the optimal policy π^*_{t+1} is reduced to $\pi^* : b \mapsto a$. Belief $b_{t+1} (t \geq 0)$ is updated

[1] In discrete case, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$.
[2] In discrete case, $Z : \mathcal{S} \times \mathcal{O} \to [0, 1]$.

according to Bayes' rule

$$b_{t+1}(s_{t+1}) = \frac{Z(s_{t+1}, o_{t+1})P(s_{t+1}|a_t; b_t)}{P(o_{t+1}|a_t; b_t)}, \quad (1)$$

where

$$P(s_{t+1}|a_t; b_t) = \mathbb{E}_{S_t \sim b_t}\left[T(S_t, a_t, s_{t+1})\right],$$
$$P(o_{t+1}|a_t; b_t) = \mathbb{E}_{S_{t+1} \sim P(\cdot|b_t, a_t)}\left[Z(o_{t+1}, S_{t+1})\right].$$

A variable with a prime indicates a value at the next timestep. $b^{a,o'}$ denotes the posterior belief for the next state S' when the next observation o' is observed after executing action a in belief b.

The value function of belief b following a fixed policy π is given by

$$V^\pi(b) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{S_t \sim b_t}\left[R(S_t, \pi(b_t))\right]\right], \quad (2)$$

and it is the fixed point of the Bellman equation

$$V^\pi(b) = Q^\pi(b, \pi(b)),$$
$$Q^\pi(b, a) = \mathbb{E}_{S \sim b}\left[R(S, a)\right] + \gamma \mathbb{E}_{O' \sim P(\cdot|b,a)}\left[V^\pi(b^{a,O'})\right] \quad (3)$$

where $Q^\pi(b, a)$ is the action value function, i.e., the value of executing action a in belief b while future actions $a', a'', \ldots$ are chosen according to policy π.

The optimal policy π^* and its optimal value function $V^*(b)$ are given by the fixed point of the Bellman optimality equation

$$V^*(b) = \max_{a \in \mathcal{A}} Q^*(b, a),$$
$$Q^*(b, a) = \mathbb{E}_{S \sim b}\left[R(S, a)\right] + \gamma \mathbb{E}_{O' \sim p(\cdot|b,a)}\left[V^*(b^{a,O'})\right],$$
$$\pi^* = \arg\max_{a \in \mathcal{A}} Q^*(b, a). \quad (4)$$

$V^*(b)$ can be estimated by the value iteration algorithm $V_d = HV_{d-1} (d \geq 1)$, where V_d is the d-step value function and H is the Bellman operator

$$(HV)(b) = \max_{a \in \mathcal{A}}\left[\mathbb{E}_{S \sim b}\left[R(S, a)\right] + \gamma \mathbb{E}_{O' \sim p(\cdot|b,a)}\left[V(b^{a,O'})\right]\right]. \quad (5)$$

H is known to be isotonic, contractive, and V_d is guaranteed to be more accurate than V_{d-1}; that is, if V_{d-1} satisfies $\varepsilon = \sup_b |V^*(b) - V_{d-1}(b)|$, then V_d has an error bound of $|V^*(b) - \hat{V}_d(b)| \leq \gamma\varepsilon$ where $\gamma \in (0, 1)$ [Ross et al., 2008].

The initial value $V_0(b)$ of belief b used for value iteration can be defined using the initial Q-value $Q_0(s, a)$ over states and actions,

$$V_0(b) = \max_{a \in \mathcal{A}} \mathbb{E}_{S \sim b(\cdot)}\left[Q_0(S, a)\right]. \quad (6)$$

A simple choice for $Q_0(s, a)$ is the reward $Q_0(s, a) = R(s, a)$. Alternatively, a QMDP approximation $Q_0(s, a) = Q^{MDP}(s, a)$ may be used [Littman, 1995], where $Q^{MDP}(s, a)$ is given by running MDP value iteration, and approximating the POMDP by an MDP.

In kernel POMDPs, all the expectations appearing above are computed nonparametrically, without explicitly estimating the distributions $b(S)$, $P(S'|a; b)$, $P(O'|a; b)$, $b^{a,o'}(S')$ and transition and observation models, T and Z. Instead, we obtain feature representations of the distributions (distribution embeddings) and the models (conditional embedding operators), as described in Section 3.

We end this section with a key to matching the probabilistic and reinforcement learning results presented in this section with their nonparametric, kernel-based counterparts in the following section. Bayes' rule (1) becomes the kernel Bayes' rule eq.(12), and its empirical counterpart leads to the updates (15)(16); the Bellman equations (3)(4) take the form of Claims 1-4, the Bellman operator (5) becomes the kernel Bellman operator (19), and the value initializations (6) lead to the initializations (20),(21).

3 Kernel Methods for Probabilities

In the present section, we provide an overview of distribution embeddings in reproducing kernel Hilbert spaces [Smola et al., 2007, Sriperumbudur et al., 2010, Gretton et al., 2012]. The embeddings are represented as mean (nonlinear) features of distributions, hence may also be referred to as *mean embeddings*. We also recall conditional embedding operators [Song et al., 2009, Song et al., 2010], and the Kernel Bayes' Rule (KBR) [Fukumizu et al., 2011]. Mean embeddings can be updated using conditional embedding operators; in particular, KBR allows us to obtain posterior embeddings given prior embeddings in feature spaces.

3.1 Embedding Distributions

Let $\mathcal{H}_\mathcal{X}$ be an RKHS associated with a bounded and measurable positive definite kernel $k_\mathcal{X} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ over domain $(\mathcal{X}, \mathcal{B}_\mathcal{X})$, with $\langle \cdot, \cdot \rangle_{\mathcal{H}_\mathcal{X}}$ the corresponding inner product. The embedding of a distribution P over $(\mathcal{X}, \mathcal{B}_\mathcal{X})$ in $\mathcal{H}_\mathcal{X}$ is given by the RKHS element $\mu_X = \mathbb{E}_{X \sim P}\left[k_\mathcal{X}(X, \cdot)\right] \in \mathcal{H}_\mathcal{X}$. μ_X coincides with the unique element satisfying $\langle \mu_X, f \rangle_{\mathcal{H}_\mathcal{X}} = \mathbb{E}_{X \sim P}\left[f(X)\right]$ for all $f \in \mathcal{H}_\mathcal{X}$, which means that the expectation of any function $f \in \mathcal{H}_\mathcal{X}$ can be computed as an inner product of the embedding μ_X and f in $\mathcal{H}_\mathcal{X}$, without explicitly estimating the distribution P.

The element μ_X can be estimated by a linear combination of feature vectors on samples $(X_1, \ldots, X_n)$ over $\mathcal{X}$, i.e., $\hat{\mu}_X = \Upsilon\boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top \in \mathbb{R}^n$ and $\Upsilon = (k_\mathcal{X}(\cdot, X_1), \ldots, k_\mathcal{X}(\cdot, X_n))$. Using the empirical embedding $\hat{\mu}_X$, the expectation can be nonparametrically estimated as

$$\mathbb{E}_{X\sim P}[f(X)] \sim \langle \hat{\mu}_X, f\rangle_{\mathcal{H}_\mathcal{X}} = \boldsymbol{\alpha}^\top \mathbf{f}, \tag{7}$$

where $\mathbf{f} = (f(X_1), \ldots, f(X_n))^\top$.

We use characteristic kernels, e.g., Gaussian kernels and Laplacian kernels, which guarantee that each distribution maps to a unique embedding in the RKHS [Sriperumbudur et al., 2010, Fukumizu et al., 2011].

3.2 Conditional Embedding Operators & Kernel Bayes' Rule (KBR)

Let $\mathcal{H}_\mathcal{X}$ and $\mathcal{H}_\mathcal{Y}$ be RKHSs associated with $k_\mathcal{X}$ and $k_\mathcal{Y}$ over $(\mathcal{X}, \mathcal{B}_\mathcal{X})$ and $(\mathcal{Y}, \mathcal{B}_\mathcal{Y})$, respectively. Let (X, Y) be a random variable taking values on $\mathcal{X}\times\mathcal{Y}$ with distribution P and the density $p(x, y)$. The conditional density functions $\{p(Y|X = x)|x \in \mathcal{X}\}$ define a family of embeddings $\{\mu_{Y|x}\}$ in $\mathcal{H}_\mathcal{Y}$. According to [Song et al., 2009], a mapping from $k_\mathcal{X}(x, \cdot) \in \mathcal{H}_\mathcal{X}$ to $\mu_{Y|x} \in \mathcal{H}_\mathcal{Y}$ for all $x \in \mathcal{X}$ can be characterized by conditional embedding operator $\mathcal{U}_{Y|X} : \mathcal{H}_\mathcal{X} \to \mathcal{H}_\mathcal{Y}$,

$$\mu_{Y|x} = \mathcal{U}_{Y|X} k_\mathcal{X}(x, \cdot) = C_{YX} C_{XX}^{-1} k_\mathcal{X}(x, \cdot), \tag{8}$$

where C_{YX} and C_{XX} are uncentred covariance operators with respect to P,

$$\begin{aligned} C_{YX} &= \mathbb{E}_{(X,Y)\sim P}\left[k_\mathcal{Y}(Y, \cdot) \otimes k_\mathcal{X}(X, \cdot)\right], \\ C_{XX} &= \mathbb{E}_{(X,Y)\sim P}\left[k_\mathcal{X}(X, \cdot) \otimes k_\mathcal{X}(X, \cdot)\right], \end{aligned} \tag{9}$$

and we use the tensor product definition $(a \otimes b)c = a\langle b, c\rangle$. Feature representations of conditional distributions $\mu_{Y|x}$ are obtained by applying the operator $\mathcal{U}_{Y|X}$ to the feature map $k_\mathcal{X}(x, \cdot)$.

Since a posterior distribution is also written as a conditional distribution, the embedding of a posterior can be expressed as a conditional embedding operator [Fukumizu et al., 2011]. Let Π be a prior distribution with density $\pi(x)$, and $(\bar{X}, \bar{Y})$ be a new random variable with distribution Q corresponding to the density $q(x, y) = p(y|x)\pi(x)$. The embedding of a posterior $q(\bar{X}|\bar{Y} = y)$ given y can be expressed by a corresponding conditional embedding operator $\mathcal{U}_{\bar{X}|\bar{Y}}$

$$\mu_{\bar{X}|y} = \mathcal{U}_{\bar{X}|\bar{Y}} k_\mathcal{Y}(y, \cdot) = C_{\bar{X}\bar{Y}} C_{\bar{Y}\bar{Y}}^{-1} k_\mathcal{Y}(y, \cdot), \tag{10}$$

where $C_{\bar{X}\bar{Y}}$ and $C_{\bar{Y}\bar{Y}}$ are covariance operators with respect to Q.

$$\begin{aligned} C_{\bar{X}\bar{Y}} &= \mathbb{E}_{(\bar{X},\bar{Y})\sim Q}\left[k_\mathcal{X}(\bar{X}, \cdot) \otimes k_\mathcal{Y}(\bar{Y}, \cdot)\right], \\ C_{\bar{Y}\bar{Y}} &= \mathbb{E}_{(\bar{X},\bar{Y})\sim Q}\left[k_\mathcal{Y}(\bar{Y}, \cdot) \otimes k_\mathcal{Y}(\bar{Y}, \cdot)\right]. \end{aligned} \tag{11}$$

See Appendix for further details and empirical estimates.

The inference underlying kPOMDPs is accomplished with the embedding operator $\mathcal{U}_{Y|X}$ and posterior embedding operator $\mathcal{U}_{\bar{X}|\bar{Y}}$.

4 Kernel POMDPs (kPOMDPs)

We now present our main results: we formulate POMDPs in feature spaces, and propose a kernelized value iteration algorithm. A key to the notation is given in Table 1.

4.1 Kernel Bellman Equations (KBEs)

To kernelize the Bellman equations, we introduce three RKHSs for state set $\mathcal{S}$, action set $\mathcal{A}$, and observation set $\mathcal{O}$. Let $\mathcal{H}_\mathcal{S}$, $\mathcal{H}_\mathcal{A}$, $\mathcal{H}_\mathcal{O}$ be RKHSs associated with bounded and measurable positive definite kernels $k_\mathcal{S}$, $k_\mathcal{A}$, $k_\mathcal{O}$ over $(\mathcal{S}, \mathcal{B}_\mathcal{S})$, $(\mathcal{A}, \mathcal{B}_\mathcal{A})$, $(\mathcal{O}, \mathcal{B}_\mathcal{O})$, respectively. $\langle\cdot,\cdot\rangle_{\mathcal{H}_\mathcal{S}}$, $\langle\cdot,\cdot\rangle_{\mathcal{H}_\mathcal{A}}$, $\langle\cdot,\cdot\rangle_{\mathcal{H}_\mathcal{O}}$ denote their respective inner products, and $\varphi(S)$, $\psi(A)$, $\phi(O)$ their feature vectors.

The relevant distributions $b(S)$, $P(S'|a; b)$, $P(O'|a; b)$, $b^{a,o'}(S')$ can be embedded in the corresponding RKHSs $\mathcal{H}_\mathcal{S}$, $\mathcal{H}_\mathcal{O}$ as follows:

$$\begin{aligned} \mu_S &= \mathbb{E}_{S\sim b(\cdot)}[\varphi(S)], \\ \mu_{S'|a;\mu_S} &= \mathbb{E}_{S'\sim p(\cdot|a;b)}[\varphi(S')], \\ \mu_{O'|a;\mu_S} &= \mathbb{E}_{O'\sim p(\cdot|a;b)}[\phi(O')], \\ \mu_{S'}^{a,o'} &= \mathbb{E}_{S'\sim b^{a,o'}(\cdot)}[\varphi(S')]. \end{aligned}$$

These embeddings are related via conditional embedding operators expressing transition models T, observation models Z, and posteriors.

Let $\mathcal{U}_{S'|S,A} : \mathcal{H}_\mathcal{S} \otimes \mathcal{H}_\mathcal{A} \to \mathcal{H}_\mathcal{S}$ be the conditional embedding operator for the transition model T, and $\mathcal{U}_{O|S} : \mathcal{H}_\mathcal{S} \to \mathcal{H}_\mathcal{O}$ be the operator for the observation model Z. Let $\mathcal{U}_{\bar{S}|\bar{O}}^{(a,\mu_S)} : \mathcal{H}_\mathcal{O} \to \mathcal{H}_\mathcal{S}$ be the posterior embedding operator corresponding to eq.(10), where we use the prior embedding $\mu_{S'|a;\mu_S}$ in the KBR.

The four embeddings above are related as

$$\begin{aligned} \mu_{S'|a;\mu_S} &= \mathcal{U}_{S'|S,A}\mu_S \otimes k_A(a, \cdot), \\ \mu_{O'|a;\mu_S} &= \mathcal{U}_{O|S}\mu_{S'|a;\mu_S}, \\ \mu_{S'}^{a,o'} &= \mathcal{U}_{\bar{S}|\bar{O}}^{(a,\mu_S)} k_O(o', \cdot). \end{aligned} \tag{12}$$

The sequence of mappings $\mu_S \mapsto \mu_{S'|a;\mu_S} \mapsto \mu_{O'|a;\mu_S} \mapsto \mu_{S'}^{a,o'}$ depends on the action a. A policy π is defined to be a mapping from embeddings μ_S to actions. In evaluating policies, value functions $V(\cdot)$ are defined to be functions of embeddings μ_S, and thus (indirectly) of the actions a and observations o.

Table 1: Notation

	State	Action	Observation	(State, Action)
domain	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{O}$	$\mathcal{S}\times\mathcal{A}$
r.v.	S	A	O	(S,A)
Instance	s	a	o	(s,a)
kernel	$k_{\mathcal{S}}(\cdot,\cdot)$	$k_{\mathcal{A}}(\cdot,\cdot)$	$k_{\mathcal{O}}(\cdot,\cdot)$	$k_{\mathcal{S}\times\mathcal{A}}(\cdot,\cdot)=k_{\mathcal{S}}(\cdot,\cdot)\otimes k_{\mathcal{A}}(\cdot,\cdot)$
RKHS	$\mathcal{H}_{\mathcal{S}}$	$\mathcal{H}_{\mathcal{A}}$	$\mathcal{H}_{\mathcal{O}}$	$\mathcal{H}_{\mathcal{S}\times\mathcal{A}}=\mathcal{H}_{\mathcal{S}}\otimes\mathcal{H}_{\mathcal{A}}$
feature map	$\varphi(s)=k_{\mathcal{S}}(s,\cdot)$	$\psi(a)=k_{\mathcal{A}}(a,\cdot)$	$\phi(o)=k_{\mathcal{O}}(o,\cdot)$	$\vartheta(s,a)=k_{\mathcal{S}\times\mathcal{A}}((s,a),\cdot)$
finite Sample Set	$\mathcal{S}_0$	$\mathcal{A}_0$	$\mathcal{O}_0$	$(\mathcal{S}_0,\mathcal{A}_0)$
feature matrix	Υ	Ψ	Φ	Θ
Gram matrix	$G_S=\Upsilon^\top\Upsilon$	$G_A=\Psi^\top\Psi$	$G_O=\Phi^\top\Phi$	$G_{(S,A)}=G_S\odot G_A$
feature column	$\mathbf{k}_S(s)=\Upsilon^\top\varphi(s)$	$\mathbf{k}_A(a)=\Psi^\top\psi(a)$	$\mathbf{k}_O(o)=\Phi^\top\phi(o)$	$\mathbf{k}_{(S,A)}(s,a)=\Theta^\top\vartheta(a,o)$

Using value functions over embeddings, the expected immediate rewards and expected values of posterior beliefs in the Bellman equations (Section 2) can be computed as inner products in RKHSs,

$$\begin{aligned}\mathbb{E}_{S\sim b}\left[R(S,a)\right]&=\left\langle\mu_S,R\left(\cdot,a\right)\right\rangle_{\mathcal{H}_{\mathcal{S}}},\\ \mathbb{E}_{O'\sim P(\cdot|b,a)}\left[V^\pi(b^{a,O'})\right]&=\left\langle\mu_{O'|a;\mu_S},V^\pi\left(\mu_{S'}^{a,(\cdot)}\right)\right\rangle_{\mathcal{H}_{\mathcal{O}}},\\ \mathbb{E}_{O'\sim P(\cdot|b,a)}\left[V^*(b^{a,O'})\right]&=\left\langle\mu_{O'|a;\mu_S},V^*\left(\mu_{S'}^{a,(\cdot)}\right)\right\rangle_{\mathcal{H}_{\mathcal{O}}},\end{aligned}$$

assuming $R(\cdot,a)\in\mathcal{H}_{\mathcal{S}}$ and $V^\pi\left(\mu_{S'}^{a,(\cdot)}\right),V^*\left(\mu_{S'}^{a,(\cdot)}\right)\in\mathcal{H}_{\mathcal{O}}$.

We are now ready to introduce the kernel Bellman equation as an operation on the embeddings μ_S. Let $\mathcal{P}_{\mathcal{S}}$ be the set of beliefs and $\mathcal{I}_{\mathcal{S}}$ be the set of embeddings of $\mathcal{P}_{\mathcal{S}}$ in $\mathcal{H}_{\mathcal{S}}$.

Claim 1. *Let $R(\cdot,a)\in\mathcal{H}_{\mathcal{S}}$ and $V^\pi\left(\mu_{S'}^{a,(\cdot)}\right)\in\mathcal{H}_{\mathcal{O}}$ for all $a\in\mathcal{A}$ and $\mu_S\in\mathcal{I}_{\mathcal{S}}$. The kernel Bellman equations on $\mathcal{H}_{\mathcal{S}}$ are*

$$\begin{aligned}V^\pi(\mu_S)&=Q^\pi\left(\mu_S,\pi\left(\mu_S\right)\right),\\ Q^\pi(\mu_S,a)&=\left\langle\mu_S,R\left(\cdot,a\right)\right\rangle_{\mathcal{H}_{\mathcal{S}}}+\gamma\left\langle\mu_{O'|a;\mu_S},V^\pi\left(\mu_{S'}^{a,(\cdot)}\right)\right\rangle_{\mathcal{H}_{\mathcal{O}}}.\end{aligned}$$

The solution of these equations following a fixed policy π on $\mathcal{H}_{\mathcal{S}}$ yields a value function $V^\pi(\cdot)$. The kernel Bellman optimality equations take similar form.

Claim 2. *Let $R(\cdot,a)\in\mathcal{H}_{\mathcal{S}}$ and $V^*\left(\mu_{S'}^{a,(\cdot)}\right)\in\mathcal{H}_{\mathcal{O}}$ for all $a\in\mathcal{A}$ and $\mu_S\in\mathcal{I}_{\mathcal{S}}$. The kernel Bellman optimality equations on RKHS $\mathcal{H}_{\mathcal{S}}$ are*

$$\begin{aligned}V^*(\mu_S)&=\max_{a\in\mathcal{A}}Q^*(\mu_S,a),\\ Q^*(\mu_S,a)&=\left\langle\mu_S,R\left(\cdot,a\right)\right\rangle_{\mathcal{H}_{\mathcal{S}}}+\gamma\left\langle\mu_{O'|a;\mu_S},V^*\left(\mu_{S'}^{a,(\cdot)}\right)\right\rangle_{\mathcal{H}_{\mathcal{O}}},\\ \pi^*(\mu_S)&=\arg\max_{a\in\mathcal{A}}Q^*\left(\mu_S,a\right).\end{aligned}\tag{13}$$

The solution of these equations is the optimal value function $V^*(\cdot)$ with corresponding optimal policy $\pi^*(\cdot)$ on $\mathcal{H}_{\mathcal{S}}$. We use the following tuple for solving a POMDP in feature space:

$$\left\langle\mathcal{H}_{\mathcal{S}},\mathcal{H}_{\mathcal{A}},\mathcal{U}_{S|S,A},R,\mathcal{H}_{\mathcal{O}},\mathcal{U}_{O|S},\mathcal{U}_{\bar{S}|\bar{O}}^{(\mathcal{A},\mathcal{I}_{\mathcal{S}})}\right\rangle,$$

where $\mathcal{U}_{\bar{S}|\bar{O}}^{(\mathcal{A},\mathcal{I}_{\mathcal{S}})}=\{\mathcal{U}_{\bar{S}|\bar{O}}^{(a,\mu_S)}|a\in\mathcal{A},\mu_S\in\mathcal{I}_{\mathcal{S}}\}$.

4.2 Empirical Expression

We next provide a finite sample version of the kernel POMDP. Suppose that $D_n=\{(\tilde{s}_i,\tilde{o}_i),\tilde{a}_i,\tilde{R}_i,(\tilde{s}_i',\tilde{o}_i')\}_{i=1}^n$ are n training samples according to a POMDP $\langle\mathcal{S},\mathcal{A},T,R,\mathcal{O},Z\rangle$. Note that state samples $\{(\tilde{s}_i,\tilde{s}_i')\}$ are included, and will used for estimating belief embeddings μ_S, $\mu_{S'}^{a,o'}$. We assume that such samples from the true state are available for training, but not during the test phase.

Let $\mathcal{S}_0$, $\mathcal{O}_0$, $\mathcal{A}_0$, $\mathcal{S}_0'$, $\mathcal{O}_0'$ be finite sets of states, observations, and actions taken from the training samples D_n. The associated feature vectors are

$$\begin{aligned}&\Upsilon=\left(\varphi\left(\tilde{s}_1\right),\ldots,\varphi\left(\tilde{s}_n\right)\right),\Upsilon'=\left(\varphi\left(\tilde{s}_1'\right),\ldots,\varphi\left(\tilde{s}_n'\right)\right),\\ &\Phi=\left(\phi\left(\tilde{o}_1\right),\ldots,\phi\left(\tilde{o}_n\right)\right),\Phi'=\left(\phi\left(\tilde{o}_1'\right),\ldots,\phi\left(\tilde{o}_n'\right)\right),\\ &\Psi=\left(\psi\left(\tilde{a}_1\right),\ldots,\psi\left(\tilde{a}_n\right)\right),\Theta=\left(\vartheta\left(\tilde{s}_1,\tilde{a}_1\right),\ldots,\vartheta\left(\tilde{s}_n,\tilde{a}_n\right)\right).\end{aligned}$$

We build Gram matrices from the feature matrices Υ, Ψ, Φ as $G_S=\Upsilon^\top\Upsilon$, $G_A=\Psi^\top\Psi$, $G_O=\Phi^\top\Phi$, $G_{(S,A)}=G_S\odot G_A$, where $\odot$ denotes the Hadamard (element-wise) product. The embeddings μ_S, $\mu_{O'|a;\mu_S}$, $\mu_{S'}^{a,o'}$ take respective forms

$$\begin{aligned}\hat{\mu}_S&=\Upsilon\boldsymbol{\alpha},\\ \hat{\mu}_{O'|a;\mu_S}&=\Phi\boldsymbol{\beta'}_{a;\boldsymbol{\alpha}},\\ \hat{\mu}_{S'}^{a,o'}&=\Upsilon\boldsymbol{\alpha}'_{a,o'}.\end{aligned}\tag{14}$$

The update rules for the weight vectors $\boldsymbol{\alpha}\mapsto\boldsymbol{\beta'}_{a;\boldsymbol{\alpha}}\mapsto\boldsymbol{\alpha}'_{a,o'}$ corresponding to $\mu_S\mapsto\mu_{O'|a;\mu_S}\mapsto\mu_{S'}^{a,o'}$ in eq.(12) are given as follows:

- The update $\boldsymbol{\alpha}\mapsto\boldsymbol{\beta'}_{a;\boldsymbol{\alpha}}$ uses the empirical estimates of conditional embedding operators $\mathcal{U}_{S'|S,A}$,

$\mathcal{U}_{O|S}$, which results in a linear transformation $\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}} = L_{O|S,a}\boldsymbol{\alpha}$ for all $a \in \mathcal{A}$ by the $n \times n$ matrix $L_{O|S,a}$:

$$(G_S + \varepsilon_S n I_n)^{-1} G_{SS'} \left(G_{(S,A)} + \varepsilon_{(S,A)} n I_n\right)^{-1} G_{(S,A)(S,a)} \tag{15}$$

with $G_{SS'} := \Upsilon^\top \Upsilon'$, $G_{(S,A)(S,a)} := D\left(\mathbf{k}_A(a)\right) G_S$, and $\mathbf{k}_A(a) = \Psi^\top \psi(a)$.

- The update $\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}} \mapsto \boldsymbol{\alpha}'_{a,o'}$ is based on the kernel Bayes' rule. The Gram matrix expression, eq.(25), results in $\boldsymbol{\alpha}'_{a,o'} = R_{S|O}(\hat{\boldsymbol{\beta}}'_{a;\boldsymbol{\alpha}})\mathbf{k}_O(o')$ using a non-negative vector $\hat{\boldsymbol{\beta}}'_{a;\boldsymbol{\alpha}}$ and an $n \times n$ matrix $R_{S|O}(\hat{\boldsymbol{\beta}}'_{a;\boldsymbol{\alpha}})$:

$$\left(D(\hat{\boldsymbol{\beta}}'_{a;\boldsymbol{\alpha}})G_O + \epsilon n I_n\right)^{-1} D(\hat{\boldsymbol{\beta}}'_{a;\boldsymbol{\alpha}}). \tag{16}$$

Given samples D_n, the embeddings $\hat{\mu}_S, \hat{\mu}_{O'|a;\mu_S}, \hat{\mu}^{a,o'}_{S'}$ are identified with n-dimensional vectors $\boldsymbol{\alpha}, \boldsymbol{\beta}'_{a;\boldsymbol{\alpha}}, \boldsymbol{\alpha}'_{a;\boldsymbol{\alpha}} \in \mathbb{R}^n$, respectively, and the Bellman equations (Claims 1, 2) can be represented in terms of these weight vectors. Therefore, the belief and predictive distributions are represented by n-dimensional vectors for any sets $\mathcal{S}$ and $\mathcal{O}$.

Claim 3. *Given samples D_n, the kernel Bellman equation (Claim 1) has the empirical expression*

$$\begin{aligned}\hat{V}^\pi(\boldsymbol{\alpha}) &= \hat{Q}^\pi(\boldsymbol{\alpha}, \pi(\boldsymbol{\alpha})),\\ \hat{Q}^\pi(\boldsymbol{\alpha}, a) &= \boldsymbol{\alpha}^\top \boldsymbol{R}_a + \gamma \boldsymbol{\beta}'^\top_{a;\boldsymbol{\alpha}} \hat{\mathbf{V}}^\pi\left(\boldsymbol{\alpha}'_{a,\mathcal{O}_0}\right),\end{aligned} \tag{17}$$

where $\mathbf{R}_a = (R(\tilde{s}_1, a), \ldots, R(\tilde{s}_n, a))^\top \in \mathbb{R}^n$ is the reward vector on samples $\mathcal{S}_0$ for action a and $\hat{\mathbf{V}}^\pi\left(\boldsymbol{\alpha}'_{a,\mathcal{O}_0}\right) = \left(\hat{V}^\pi\left(\boldsymbol{\alpha}'_{a,\tilde{o}_1}\right), \ldots, \hat{V}^\pi\left(\boldsymbol{\alpha}'_{a,\tilde{o}_n}\right)\right)^\top \in \mathbb{R}^n$ is the posterior belief value vector on samples $\mathcal{O}_0$ given action a.

Claim 4. *Given samples D_n, the kernel Bellman optimality equation (Claim 2) has the expression*

$$\begin{aligned}\hat{V}^*(\boldsymbol{\alpha}) &= \max_{a \in \mathcal{A}} \hat{Q}^*(\boldsymbol{\alpha}, a),\\ \hat{Q}^*(\boldsymbol{\alpha}, a) &= \boldsymbol{\alpha}^\top \boldsymbol{R}_a + \gamma \boldsymbol{\beta}'^\top_{a;\boldsymbol{\alpha}} \mathbf{V}^*\left(\boldsymbol{\alpha}'_{a,\mathcal{O}_0}\right),\\ \hat{\pi}^*(\boldsymbol{\alpha}) &= \arg\max_{a \in \mathcal{A}} \hat{Q}^*(\boldsymbol{\alpha}, a),\end{aligned} \tag{18}$$

where $\mathbf{R}_a = (R(\tilde{s}_1, a), \ldots, R(\tilde{s}_n, a))^\top \in \mathbb{R}^n$ and $\hat{\mathbf{V}}^*\left(\boldsymbol{\alpha}'_{a,\mathcal{O}_0}\right) = \left(\hat{V}^*\left(\boldsymbol{\alpha}'_{a,\tilde{o}_1}\right), \ldots, \hat{V}^*\left(\boldsymbol{\alpha}'_{a,\tilde{o}_n}\right)\right)^\top \in \mathbb{R}^n$.

Figure 1 illustrates the planning forward search using the Bellman equations. Even when the assumptions in Claims 1, 2 do not hold, we use Claims 3, 4 as an approximation, and Claim 4 for value iteration. Let $\hat{H}_n$ be the kernel Bellman operator,

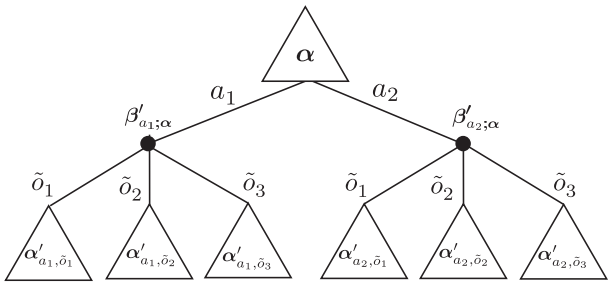


Figure 1: A search tree using kernel Bellman equations. Beliefs are represented by n-dimensional weight vectors $\boldsymbol{\alpha}$ on samples. The expected immediate reward (first term) is given by the linear combination $\boldsymbol{\alpha}^\top \boldsymbol{R}_a$ on samples $\mathcal{S}_0$ and associated with the action link a. The discounted expected value for next beliefs (second term) is given by the linear combination $\gamma \boldsymbol{\beta}'^\top_{a;\boldsymbol{\alpha}} \hat{\mathbf{V}}\left(\boldsymbol{\alpha}'_{a,\mathcal{O}_0}\right)$ on samples $\mathcal{O}_0$ and associated with the observation links $\mathcal{O}_0 = \{\tilde{o}_1, \ldots, \tilde{o}_n\}$. Observation links are expanded with respect to finite set $\mathcal{O}_0$ instead of $\mathcal{O}$. The kernel Bayes' rule is applied to each pair $(a, \tilde{o})$. Values are back-propagated bottom to top starting at initial values $V_0(\cdot)$ in the kernel value iteration algorithm.

$$\left(\hat{H}_n V\right)(\boldsymbol{\alpha}) = \max_{a \in \mathcal{A}} \left[\boldsymbol{\alpha}^\top \boldsymbol{R}_a + \gamma \boldsymbol{\beta}'^\top_{a;\boldsymbol{\alpha}} \mathbf{V}\left(\boldsymbol{\alpha}'_{a,\mathcal{O}_0}\right)\right]. \tag{19}$$

We run value iteration $\hat{V}_d = \hat{H}_n \hat{V}_{d-1} (d \geq 1)$ with an initial value function $V_0(\cdot)$ on weights. The detailed algorithm is shown in Algorithm 1. The computational complexity is given in Subsection 4.3. We use the same value initializations as in distributional POMDPs (Section 2). If $Q_0(\cdot, a) \in \mathcal{H}_\mathcal{S}$ for all $a \in \mathcal{A}$, the initial values $V_0(\cdot)$ over embeddings can be set as

$$V_0(\mu_S) = \max_{a \in \mathcal{A}} \langle \mu_S, Q_0(\cdot, a) \rangle_{\mathcal{H}_\mathcal{S}}, \tag{20}$$

which leads to the empirical expression

$$V_0(\boldsymbol{\alpha}) = \max_{a \in \mathcal{A}} \boldsymbol{\alpha}^\top \mathbf{Q}^0_a, \tag{21}$$

where $\mathbf{Q}^0_a = \left(Q_0(\tilde{s}_1, a), \ldots, Q_0(\tilde{s}_n, a)\right)^\top \in \mathbb{R}^n$ is an initial action value vector on samples $\mathcal{S}_0$. The reward function $Q_0(s, a) = R(s, a)$ or the QMDP value $Q_0(s, a) = Q^{MDP}(s, a)$ can be used for initialization.

Since $\boldsymbol{\alpha}, \boldsymbol{\beta}'_{a;\boldsymbol{\alpha}}, \boldsymbol{\alpha}'_{a;\boldsymbol{\alpha}}$ do not always give nonnegative vectors, the Bellman operator $\hat{H}_n$ is not guaranteed to be isotonic and contractive, although empirically, $\hat{H}_n$ can be used for the value iteration algorithm. $\hat{H}_n$ is corrected to have the isotonic and contractive properties by approximating the weight vectors as probability

Algorithm 1 Kernel Value Iteration ($\boldsymbol{\alpha}$, d)

Input: belief weights $\boldsymbol{\alpha}$, tree depth d
Output: Value $V_d(\boldsymbol{\alpha})$, Policy $\pi_d(\boldsymbol{\alpha})$
if $d = 0$ **then**
  Set: $V_d(\boldsymbol{\alpha}) = \max_{a \in \mathcal{A}} Q_0(\boldsymbol{\alpha}, a)$,
    $\pi_d(\boldsymbol{\alpha}) = \arg\max_{a \in \mathcal{A}} Q_0(\boldsymbol{\alpha}, a)$
else
  for all $a \in \mathcal{A}$ **do**
    Compute: $\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}} = L_{O|S,a}\boldsymbol{\alpha}$
    for all $\tilde{o}_i \in \tilde{\mathcal{O}}_0$ **do**
      if $\left(\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}}\right)_i \neq 0$ **then**
        Compute posterior:
          $\boldsymbol{\alpha}'_{a,\tilde{o}_i} = R_{S|O}(\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}})\mathbf{k}_O(\tilde{o}_i)$
        Set:
          $V(\boldsymbol{\alpha}'_{a,\tilde{o}_i}) \leftarrow$ Kernel Val. Iter.$(\boldsymbol{\alpha}'_{a,\tilde{o}_i}, d-1)$
      end if
    end for
  end for
  Set: $V_d(\boldsymbol{\alpha}) = \max_{a \in \mathcal{A}} Q_d(\boldsymbol{\alpha}, a)$,
    $\pi_d(\boldsymbol{\alpha}) = \arg\max_{a \in \mathcal{A}} Q_d(\boldsymbol{\alpha}, a)$
end if

vectors [Grunewalder et al., 2012]. Let $\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}'_{a;\boldsymbol{\alpha}}, \hat{\boldsymbol{\alpha}}'_{a;\boldsymbol{\alpha}}$ be probability vectors derived from $\boldsymbol{\alpha}, \boldsymbol{\beta}'_{a;\boldsymbol{\alpha}}, \boldsymbol{\alpha}'_{a;\boldsymbol{\alpha}}$ according to $\hat{w}_i = \frac{\max\{w_i,0\}}{\sum_{i=1}^n \max\{w_i,0\}}$ for weight vectors $\boldsymbol{w}$. The corresponding kernel Bellman operator $\hat{H}_n^+$ using the probability vectors $\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}'_{a;\boldsymbol{\alpha}}$ is guaranteed to be isotonic and contractive. The proof is given in the Supplementary material.

Given samples D_n, we use the following tuple for the kernel value iteration:

$$\left\langle (\mathcal{H}_S, \mathbf{k}_S), (\mathcal{H}_A, \mathbf{k}_A), R, (\mathcal{H}_O, \mathbf{k}_O), L_{O|S,\mathcal{A}}, R_{S|O}(\cdot) \right\rangle,$$

where $L_{O|S,\mathcal{A}} = \{L_{O|S,a} | a \in \mathcal{A}\}$.

4.3 Computational Complexity

The update rule $\boldsymbol{\alpha} \mapsto \boldsymbol{\beta'}_{a;\boldsymbol{\alpha}}$ has complexity $O(n^2)$ with respect to the number of samples n for an action a, where the matrix $L_{S|O,a}$ is computed only once in the training phase. The update rule $\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}} \mapsto \boldsymbol{\alpha}'_{a,o'}$ has complexity $O(n^3)$ for an observation o', due to the inversion of an $n \times n$ matrix (eq. 16). In total, the computation of the posterior weights $\boldsymbol{\alpha}'_{a,o'}$ for a pair (a, o') costs $O(n^3)$, compared with $O(|\mathcal{S}|^2)$ for Bayes' rule in eq.(1). Kernel value iteration to depth d costs $O(n^3(n|\mathcal{A}|)^d)$, whereas classical value iteration costs $O(|\mathcal{S}|^2(|\mathcal{O}||\mathcal{A}|)^d)$. The complexity $O(n^3)$ for computing $\boldsymbol{\alpha}'_{a,o'}$ can be reduced to $O(nr^2)$ via a low rank approximations of the $n \times n$ Gram matrices, where r is the rank.

Algorithm 2 Online Planning with Finite Horizon d

Set: T, $t = 0$
Get: Initial observation o
Compute: Initial belief $\boldsymbol{\alpha} = (G_O + n\epsilon_O I_n)^{-1}\mathbf{k}_O(o)$
repeat
  Planning: $a \leftarrow$ Kernel Value Iteration ($\boldsymbol{\alpha}$, d)
  Get: reward and new observation (R, o')
  Compute next belief: $\boldsymbol{\alpha} = R_{S|O}(\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}})\mathbf{k}_O(o')$
  $t = t + 1$
until $t > T$

5 Experiments

We implemented an online planning algorithm in POMDPs using the kernel Bellman equations (Algorithm 2). An example of kPOMDP dynamics is shown in Figure 3. An agent computes the initial belief weights $\boldsymbol{\alpha} \in \mathbb{R}^n$ by $\boldsymbol{\alpha} = (G_O + n\epsilon_O I_n)^{-1}\mathbf{k}_O(o)$ with an initial observation o, which corresponds to a belief estimate without a prior. The agent makes a decision a by kernel value iteration to finite horizon d under the current belief weights $\boldsymbol{\alpha}$. The agent then updates the belief weights using a function $R_{S|O}(\cdot)$ and predictive weights $\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}}$ when obtaining a reward R and a next observation o'. When the prediction fails (e.g., $D(\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}})\mathbf{k}_O(o') = \mathbf{0}$, or in discrete cases $\boldsymbol{\beta}'_{a;\boldsymbol{\alpha}}(o') = 0$) in the case of small training samples, we reset current belief weights and estimated initial belief weights. To make the online planning algorithms more efficient, we computed the inverses of the $n \times n$ matrices in our algorithm by combining low rank approximations based on the incomplete Cholesky factorization and the Woodbury identity [Fine and Scheinberg, 2001]

Figure 2 shows some results on benchmarks [Littman, 1995], where the state, action, and observation sets are finite. Results are $d = 2$, $d = 1$, $d = 1$ for 10×10 Grid World, Network, Hallway, respectively. The exact policy is computed by an agent having complete knowledge about the POMDP environment. The histogram policy computed by running a classical value iteration algorithm, where transition and observation models are estimated by histograms using the same samples as our algorithm. Since the histogram policy requires samples over all the combinations of states and actions to estimate transition models, we introduced prior samples drawn from a uniform prior to test kPOMDP and Histograms under exactly the same conditions. Note that KBR does not need prior samples. We used QMDP initial values and pruned action links based on the QMDP values [Ross et al., 2008]. An action link was pruned if its QMDP value was lower than the current estimated value. The KBR-controller learned the exact policy as the number of training samples

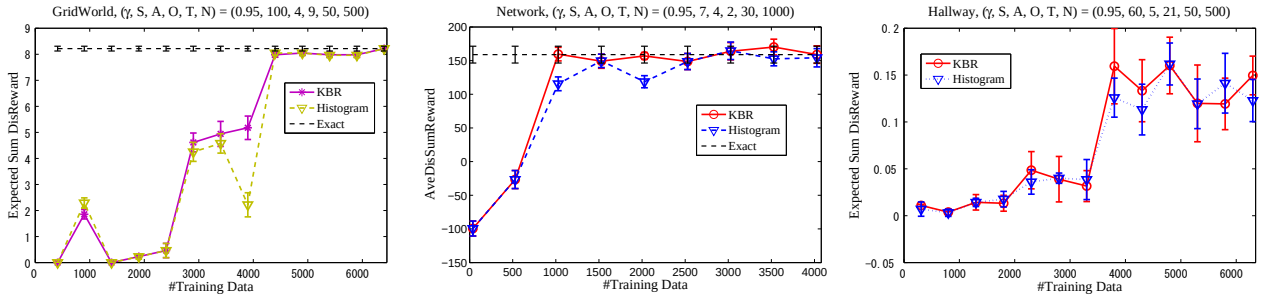


Figure 2: kPOMDPs+KBR controller. The left, middle, and right figures are 10×10 Gridworld, Network, and Hallway problems, respectively. Averaged discounted sum of rewards an agent got in test experiments are plotted against the number of training samples. We ran N experiments, where one experiment consists of T steps. The parameters are accompanied with the titles, (γ, S, A, O, T, N) = (γ, $|\mathcal{S}|$, $|\mathcal{A}|$, $|\mathcal{O}|$, T, N). Training samples are collected by uniform random actions. The leftmost plot for each figure is the result of uniform prior samples where no information about the environment. 10×10 grid world problem is similar to the 4×4 grid world, where state size is 100, reward is delivered by any action at a goal state with value 1, otherwise 0, and observations are 9 wall patterns about 4 nearest neighbors.

was increased in the 10×10 grid world and network problems. In our limited experiments, though KBR and histogram methods sometimes showed different results depending on training data, they gave similar results on average.

We also implemented a simulator of a swing-up cart-balancing system. The system consists of a cart with mass 8 kg running on a 2 m track and a freely swinging pendulum with mass 2 kg attached to the cart with a 50 cm rod. The state of the system is the angle and the angular velocity of the pendulum $(\theta, \dot{\theta})$, however the agent observes only the angle. The agent may apply a horizontal force or action to the cart, chosen from $a \in \{-250, -150, -50, 0, 50, 150, 250\}$N. The dynamics of the system are nonlinear. The states are continuous, but time is discretized in steps of 0.1 s. The objective is to balance the pendulum in the inverted position. Training samples are collected by applying uniform random actions from uniformly random states over $\theta = [-\pi/3, \pi/3]$, $\dot{\theta} = [3, 3]$. The reward function is given by $R(\theta, \dot{\theta}) = \exp(-\theta^2/2\sigma_1^2 - \dot{\theta}^2/10\sigma_2^2)$, where σ_1^2 and σ_2^2 are the variances of uniform distributions over $\theta = [-\pi/3, \pi/3]$, $\dot{\theta} = [3, 3]$, respectively.

Figure 3 shows a visualization of the kPOMDP dynamics and the inverted pendulum results (see caption details). The rightmost figure plots the averaged result of the earned rewards (height=cos(θ)) of the learned policies as a function of training samples. The episode length was 10 sec, i.e., the maximum of the total rewards is 100. The result was averaged over 20 experiments, the planning depth was $d = 1$, and the initial value function was the reward function $R(\theta, \dot{\theta})$. In kPOMDP, we used Gaussian kernels $G(\cdot, \cdot)$ for states and observations $k_{\mathcal{S}}((\theta_1, \dot{\theta}_1), (\theta_2, \dot{\theta}_2)) = G(\theta_1, \theta_2)G(\dot{\theta}_1, \dot{\theta}_2)$, $k_{\mathcal{O}}(\theta_1, \theta_2) = G(\theta_1, \theta_2)$, where the kernel parameters σ were $\sigma = \text{MedDist}/30$ for θ and $\sigma = \text{MedDist}/10$ for $\dot{\theta}$, where MedDist is the median inter-sample distance. The kernel for actions was the identity. We compared kPOMDP to histogram policies where the environment is discretized. Histogram(M) indicates $M \times M$ discretized states over $[-\pi/3, \pi/3] \times [3, 3]$. kPOMDP almost reached the maximum value 100 and showed better results than histogram policies.

6 Summary

We have introduced POMDPs in feature spaces, where beliefs over states are represented as distribution embeddings in feature spaces and updated via the kernel Bayes' rule. The Bellman equations, value functions, and policies are all expressed as functions of this feature representation. We further proposed a policy learning strategy by value iteration in the kernel framework, where the isotonic and contraction properties of the kernel Bellman operator are enforced by a simple correction. Value initialization and action edge pruning can be implemented for kernel POMDPs, following the approach of distributional POMDPs such as QMDP. Experiments confirm that the controller learned in feature space converges to the optimum policy. Our approach serves as a first step towards more powerful kernel-based algorithms for POMDPs.

A Appendix: Kernel Bayes' Rule

A variant of the kernel Bayes' rule algorithm we have used in this paper is described. [Fukumizu et al., 2011] propose a squared regularization form for the empirical

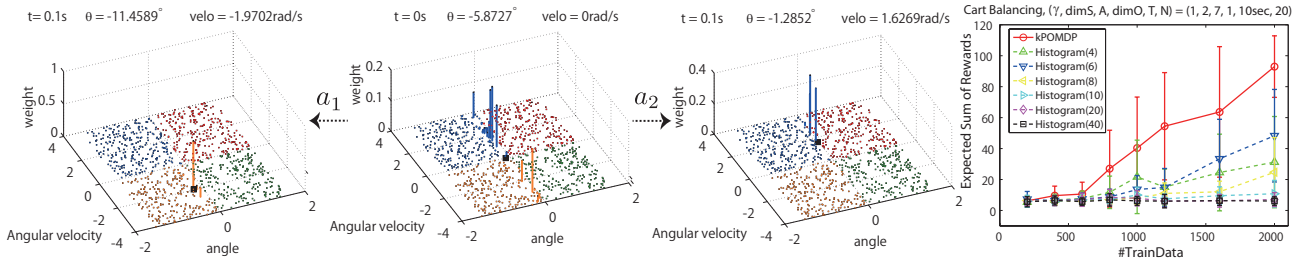


Figure 3: Inverted pendulum results with an example of kPOMDP dynamics. In the three 3D plots, all the points indicate training samples on state set $\mathcal{S} = (\theta, \dot{\theta})$ and z axis indicates the magnitude of weights on their samples (after normalization), i.e., belief embedding weights $\hat{\boldsymbol{\alpha}}$ identifying belief embedding $\hat{\mu}_S$. The true state of the system is also marked by the black point in each 3D plot. The four colors indicate different combinations of signs of states (for instance, blue denotes positive angular velocity and negative angle). The middle figure in the 3D plots shows the initial belief estimate given an initial observation $o = \theta$, estimated by $\boldsymbol{\alpha} = (G_O + n\epsilon_O I_n)^{-1}\mathbf{k}_O(o)$ in Algorithm 2. Since $\dot{\theta}$ is uncertain at the initial point, positive weights spread in the direction of $\dot{\theta}$ axis. The left and right figures in the 3D plots correspond to the updated weights $\boldsymbol{\alpha}'_{a,o'}$ of belief embeddings depending on the executed actions a_1 (positive) and a_2 (negative) after observing a new observation $o' = \theta'$. Angular velocity $\dot{\theta}$ is then well estimated by the kPOMDPs. The rightmost figure shows the averaged result of the total rewards obtained by learned policies as training samples increased. More description is in the text.

posterior embedding in (10),

$$\hat{\mu}_{\bar{X}|y} = \hat{C}_{\bar{X}\bar{Y}} \left(\hat{C}^2_{\bar{Y}\bar{Y}} + \delta n I_n\right)^{-1} \hat{C}_{\bar{Y}\bar{Y}} k_{\mathcal{Y}}(y, \cdot), \qquad (22)$$

where δ is a small regularization parameter to avoid the instability of $\hat{C}^{-1}_{\bar{Y}\bar{Y}}$, since the empirical estimate $\hat{C}_{\bar{Y}\bar{Y}}$ may include negative weights. The estimate (22) is consistent under smoothness assumptions, and converges to $\mu_{\bar{X}|y}$ in the infinite sample limit. Since we always normalize weight vectors to probability vectors as in subsection 4.2, however, we use the simpler and more computationally efficient estimate

$$\hat{\mu}_{\bar{X}|y} = \hat{C}_{\bar{X}\bar{Y}} (\hat{C}_{\bar{Y}\bar{Y}} + \epsilon n I_n)^{-1} k_{\mathcal{Y}}(y, \cdot), \qquad (23)$$

where ϵ is a small regularization parameter. Though the consistency of (23) with the combination of the normalization is not theoretically proven, experiments (Section 5) show good empirical results. In what follows we give the KBR algorithm when the estimate (23) is used with normalized weights.

Denote feature vectors $\varphi(X) = k_{\mathcal{X}}(X, \cdot)$ and $\phi(Y) = k_{\mathcal{Y}}(Y, \cdot)$. Let $U_1, \ldots, U_l$ be l samples drawn from the prior Π and $(X_1, Y_1), \ldots, (X_n, Y_n)$ be n samples drawn from P. Consider an empirical prior embedding $\hat{\mu}_\Pi = \bar{\Upsilon}\boldsymbol{\gamma}$ where $\bar{\Upsilon} = (\varphi(U_1), \ldots, \varphi(U_l))$ are feature mappings of the prior samples and $\boldsymbol{\gamma} \in \mathbb{R}^l$ is a weight vector. Define the $n \times l$ matrix $G_{XU} = \Upsilon^\top \bar{\Upsilon}$, and

$$\Upsilon \circ \Phi := (\varphi(X_1) \otimes \phi(Y_1), \ldots, \varphi(X_n) \otimes \phi(Y_n)),$$
$$\Phi \circ \Phi := (\phi(Y_1) \otimes \phi(Y_1), \ldots, \phi(Y_n) \otimes \phi(Y_n)). \qquad (24)$$

Empirical estimates of the covariance operators $C_{\bar{X}\bar{Y}}$, $C_{\bar{Y}\bar{Y}}$ are then given by $\hat{C}_{\bar{X}\bar{Y}} = (\Upsilon \circ \Phi)\boldsymbol{\beta}$, $\hat{C}_{\bar{Y}\bar{Y}} = (\Phi \circ \Phi)\boldsymbol{\beta}$ with the weight vector $\boldsymbol{\beta} = (G_X + \varepsilon n I_n)^{-1} G_{XU}\boldsymbol{\gamma}$ [Fukumizu et al., 2011]. We approximate $\boldsymbol{\beta}$ by a non-negative vector $\hat{\boldsymbol{\beta}}$, as in subsection 4.2, leading to the following proposition:

Proposition 1. *The empirical estimate eq.(23) has the following Gram matrix expression using a non-negative vector $\hat{\boldsymbol{\beta}}$ for all $y \in \mathcal{Y}$:*

$$\hat{\mu}_{\bar{X}|y} = \Upsilon R_{X|Y}(\hat{\boldsymbol{\beta}})\mathbf{k}_Y(y),$$
$$R_{X|Y}(\hat{\boldsymbol{\beta}}) := \left(D(\hat{\boldsymbol{\beta}})G_Y + \delta n I_n\right)^{-1} D(\hat{\boldsymbol{\beta}}), \qquad (25)$$

where $\mathbf{k}_Y(y) = \Phi^\top \phi(y)$ and $D(\hat{\boldsymbol{\beta}}) = \mathrm{diag}(\hat{\boldsymbol{\beta}})$.

Proof. The proof follows the same reasoning as Proposition 3.4 of [Fukumizu et al., 2011]. □

Let $\boldsymbol{\alpha}(y) = R_{X|Y}(\boldsymbol{\beta})\Phi^\top \phi(y) \in \mathbb{R}^n$. The posterior embedding $\mu_{\bar{X}|y}$ can be estimated by a linear combination of feature vectors on samples $\Upsilon = (\varphi(X_1), \ldots, \varphi(X_n))$ with weights $\boldsymbol{\alpha}(y) \in \mathbb{R}^n$ that depend on $y \in \mathcal{Y}$.

Acknowledgment This work has been supported in part by JSPS KAKENHI (B) 22300098. Authors thank anonymous reviewers for helpful comments.

References

[Even-dar, 2005] Eyal Even-dar. Reinforcement learning in POMDPs without resets. *IJCAI*, 690–695, 2005.

[Fine and Scheinberg, 2001] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *JMLR*, 2:243–264, 2001.

[Fukumizu et al., 2008] K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf. Kernel measures of conditional dependence. In *NIPS2008*.

[Fukumizu et al., 2011] K. Fukumizu, L. Song, and A. Gretton. Kernel Bayes' Rule. In *NIPS2011*.

[Fukumizu et al., 2011] K. Fukumizu, L. Song, and A. Gretton. Kernel bayes rule: Bayesian inference with positive definite kernels. *arXiv:1009.5736*.

[Gretton et al., 2007] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel method for the two-sample-problem. In *NIPS2007*.

[Gretton et al., 2008] A. Gretton, K. Fukumizu, C. Teo, L. Song, B. Schölkopf, and A. Smola. A kernel statistical test of independence. In *NIPS2008*.

[Gretton et al., 2012] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf and A. Smola. A Kernel Two-Sample Test. *JMLR*, 13, 671–721, 2012.

[Grunewalder et al., 2012] S. Grunewalder, G. Lever, L. Baldassarre, M. Pontil and A. Gretton. Modelling transition dynamics in MDPs with RKHS embeddings. In *ICML2012*.

[Hauskrecht, 2000] M. Hauskrecht. Value-Function Approximations for Partially Observable Markov Decision Processes. In *JAIR*, vol 13, pages 33–94, 2000.

[Littman, 1995] M. Littman, A. Cassandra, and L. Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML1995*.

[Pineau et al., 2003] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: an anytime algorithm for POMDPs. In *ICJAI*, pages 1025–1032, 2003.

[Porta. et al., 2006] J. M. Porta, N. Vlassis, and P. Poupart. Point-based value iteration for continuous POMDPs. *JMLR*, 7:2329–2367, 2006.

[Poupart et al., 2006] Pascal Poupart, Nikos A. Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete Bayesian reinforcement learning. *ICML2006*.

[Ross et al., 2008] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *JAIR*, 32(1):663–704, 2008.

[Silver and Veness, 2010] David Silver and Joel Veness. Monte-Carlo Planning in Large POMDPs. *NIPS2010*.

[Smith and Simmons, 2004] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *UAI2004*.

[Smola et al., 2007] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A Hilbert space embedding for distributions. In *ALT2007*.

[Sondik, 1971] E. J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.

[Song et al., 2009] L. Song, J. Huang, A. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *ICML2009*.

[Song et al., 2010] L. Song, B. Boots, S. Siddiqi, G. Gordon, and A. Smola. Hilbert space embeddings of hidden Markov models. In *ICML2010*.

[Song et al., 2010] L. Song, A. Gretton, and C. Guestrin. Nonparametric tree graphical models via kernel embeddings. In *AISTATS*, pages 765–772, 2010.

[Song et al., 2011] L. Song, A. Gretton, D. Bickson, Y. Low, and C. Guestrin. Kernel Belief Propagation. In *AISTATS*, 2011.

[Spaan and Vlassis, 2005] M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *JAIR*, 24:195–220, 2005.

[Sriperumbudur et al., 2010] B. Sriperumbudur, A. Gretton, K. Fukumizu, G. Lanckriet, and B. Schölkopf. Hilbert space embeddings and metrics on probability measures. *JMLR*, 11:1517–1561, 2010.

[Thrun, 2000] S. Thrun. Monte Carlo POMDPs. In *NIPS2000*.

Exploiting Structure in Cooperative Bayesian Games

Frans A. Oliehoek
Dept. of Knowledge Engineering
Maastricht University
Maastricht, The Netherlands

Shimon Whiteson
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands

Matthijs T.J. Spaan
Faculty EEMCS
Delft University of Technology
Delft, The Netherlands

Abstract

Cooperative *Bayesian games* (BGs) can model decision-making problems for teams of agents under imperfect information, but require space and computation time that is exponential in the number of agents. While *agent independence* has been used to mitigate these problems in perfect information settings, we propose a novel approach for BGs based on the observation that BGs additionally possess a different types of structure, which we call *type independence*. We propose a factor graph representation that captures both forms of independence and present a theoretical analysis showing that *non-serial dynamic programming* cannot effectively exploit type independence, while Max-Sum can. Experimental results demonstrate that our approach can tackle cooperative Bayesian games of unprecedented size.

1 Introduction

Cooperative multiagent systems are an essential tool, not only for tackling inherently distributed problems, but also for decomposing problems too complex to be tackled by a single agent (Huhns, 1987; Sycara, 1998; Panait and Luke, 2005; Vlassis, 2007; Buşoniu et al., 2008). However, to make such systems effective, the constituent agents must be able to coordinate their action selection in order to achieve a common goal.

This problem is particularly vexing in the presence of *imperfect information* (Harsanyi, 1967–1968). Even a single-agent system may have incomplete knowledge of the state of its environment (Kaelbling et al., 1998), e.g., due to noisy sensors. However, the presence of multiple agents often greatly exacerbates this problem, as each agent typically has access to only a fraction of the whole system's sensors. In principle, agents could synchronize their beliefs and coordinate their actions by communicating. However, communication is often unreliable and limited by bandwidth constraints, leaving each agent uncertain how others will act.

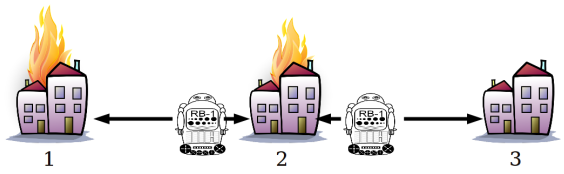


Figure 1: Illustration of multiagent decision making with imperfect information. Both agents know house 2 is on fire. However, each agent receives only a noisy observation of the single neighboring house it can observe in the distance.

Consider the example depicted in Fig. 1. Both fire-fighting agents know there is fire at house H_2 but each receives only a noisy observation about one of the other houses. Consequently, effective decision making is difficult. If agent 1 (left) observes flames in house H_1, it may be tempted to fight fire there rather than at house H_2. However, the efficacy of doing so depends on whether agent 2 (right) will fight fire in house H_2, which in turn depends on whether agent 2 observes flames in house H_3, a fact unknown to agent 1.

Such problems can be modeled as cooperative *Bayesian games (BGs)*, which extend traditional *strategic games* to model imperfect information. BGs can model a great variety of problems and are of great importance for solving sequential decision problems (Emery-Montemerlo et al., 2004; Oliehoek et al., 2008a; Kumar and Zilberstein, 2010; Spaan et al., 2011). In a BG, a *type* for each agent, specifying the private information it holds, is drawn from a distribution. Then, the agents simultaneously select actions conditioned on their individual type, and subsequently receive a team payoff based on the realized types and actions.

Unfortunately, solving cooperative BGs is NP-hard (Tsitsiklis and Athans, 1985). Just representing them

requires space exponential in the number of agents and solving them exactly requires time exponential in the number of agents and types. However, as in cooperative strategic games (Guestrin et al., 2002; Kok and Vlassis, 2006), many problems exhibit *agent independence*, i.e., the global payoff function can be decomposed as the sum of *local payoff functions*, each of which depends on the actions of only a few agents. The resulting *cooperative graphical Bayesian games (CG-BGs)* can be represented more compactly and solved more effectively using inference methods for graphical models such as *non-serial dynamic programming* (NDP) (Bertele and Brioschi, 1972) and the popular MAX-SUM algorithm (Pearl, 1988; Kok and Vlassis, 2005, 2006; Farinelli et al., 2008; Rogers et al., 2011).

This paper is motivated by the observation that BGs also exhibit another form of structure that we call *type independence*, in which the global payoff function can be decomposed as the sum of *contributions*, each of which depends on only one joint type. Such structure has been exploited when cooperative BGs are solved using heuristic search methods (Kumar and Zilberstein, 2010; Oliehoek et al., 2010), though the role of type independence has not been made explicit. However, in the worst case, heuristic search methods are no better than brute-force search and thus can perform much worse than inference-based methods, even when exploiting structure with *and/or search trees* (Marinescu and Dechter, 2009).[1] In addition, such methods do not exploit agent independence.

To address these shortcomings, this paper investigates the exploitation of type independence using inference methods. Specifically, we propose a novel *factor graph* representation that neatly captures *both* agent and type independence and thus enables the exploitation of both forms of structure. Our analysis of the computational complexity of both NDP and MAX-SUM applied to this factor graph reveals that the former cannot effectively exploit type independence; we prove that its computational complexity remains exponential in the number of types (although it still outperforms one of the search methods empirically). On the other hand, we prove that each iteration of MAX-SUM is tractable for problems with small local neighborhoods. Bounding the number of iterations thus yields a polynomial-time approximate method for such cooperative BGs.

In addition, we present extensive experimental results that demonstrate that MAX-SUM finds near-optimal solutions and that the simultaneous exploitation of both agent and type independence leads to dramatically better scalability than several state-of-the-art alternatives in the number of agents, actions, and types. This improved scalability enables the solution of cooperative BGs of unprecedented size.

[1] NDP, for example, is never slower than brute-force search, i.e., conditioning in graphical models (Koller and Friedman, 2009). When the *induced width* is small, NDP can be much faster (Dechter, 1999).

2 Background

We begin with some background on BGs and CGBGs.

Bayesian Games A Bayesian game, also called a *strategic game of imperfect information*, is an augmented strategic game in which the players hold private information (Harsanyi, 1967–1968; Osborne and Rubinstein, 1994). The private information of agent i defines its *type* $\theta_i \in \Theta_i$. The agents' payoffs depend not only on their actions, but also on their types.

Definition 1. A *Bayesian game (BG)* is a tuple $\langle \mathcal{D}, \mathcal{A}, \boldsymbol{\Theta}, \Pr(\boldsymbol{\Theta}), \langle u_1, ...u_n \rangle \rangle$, where $\mathcal{D}$ is the set of n agents, $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ is the set of joint actions $\mathbf{a} = \langle a_1, \ldots, a_n \rangle$, $\boldsymbol{\Theta} = \Theta_1 \times \cdots \times \Theta_n$ is the set of joint types $\boldsymbol{\theta} = \langle \theta_1, \ldots, \theta_n \rangle$, $\Pr(\boldsymbol{\Theta})$ is the distribution over joint types, and $u_i : \boldsymbol{\Theta} \times \mathcal{A} \to \mathbb{R}$ is the payoff function of agent i.

Since agents in a BG can condition their actions on their types, they select policies (as opposed to actions as in strategic games). A joint policy $\boldsymbol{\beta} = \langle \beta_1, ..., \beta_n \rangle$, consists of individual policies β_i for each agent i. Deterministic (pure) individual policies map each type θ_i to an action. Since we focus on *cooperative* Bayesian games, in which all agents share a single team payoff function $u = u_i$ for all i, only deterministic policies need be considered.

In a BG, the traditional Nash equilibrium is replaced by a *Bayesian Nash equilibrium (BNE).* A profile of policies $\boldsymbol{\beta} = \langle \beta_1, ..., \beta_n \rangle$ is a BNE when no agent i has an incentive to switch its policy β_i, given the policies of the other agents $\boldsymbol{\beta}_{\neq i}$. In cooperative BGs, the definition of a solution is simpler. Let the *value* of a joint policy be its expected payoff:

$$V(\boldsymbol{\beta}) = \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \Pr(\boldsymbol{\theta}) u(\boldsymbol{\theta}, \boldsymbol{\beta}(\boldsymbol{\theta})), \tag{1}$$

where $\boldsymbol{\beta}(\boldsymbol{\theta}) = \langle \beta_1(\theta_1), ..., \beta_n(\theta_n) \rangle$ is the joint action specified by $\boldsymbol{\beta}$ for joint type $\boldsymbol{\theta}$. A solution of a cooperative BG is $\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} V(\boldsymbol{\beta})$, which is a Pareto-optimal Bayesian Nash equilibrium.

Cooperative Graphical Bayesian Games. A key difficulty in BGs is that the size of their payoff matrices is exponential in the number of agents. However, many BGs contain *agent independence*, i.e., not all agents directly influence each other, which allows for compact representations and efficient solutions.

Agent independence can be formalized in *graphical (strategic) games* (Kearns et al., 2001; Kearns, 2007) and *graphical Bayesian games* (Soni et al., 2007), in which each agent has an *individual* payoff function that depends on only a subset of agents. Unfortunately, such models are not useful in cooperative games since all agents share a single payoff function in which all agents participate (otherwise they would be irrelevant). Instead, we employ *cooperative graphical BGs (CGBGs)* (Oliehoek et al., 2008b), which use *coordination (hyper-)graphs* (Guestrin et al., 2002; Kok and Vlassis, 2006) to express conditional independence between agents. In CGBGs, the single team payoff function decomposes as the sum of several *local* payoff functions.

Definition 2. A *cooperative graphical Bayesian game (CGBG)* is a tuple $\langle \mathcal{D}, \mathcal{A}, \boldsymbol{\Theta}, \Pr(\boldsymbol{\Theta}), \mathcal{U} \rangle$ where $\mathcal{D}$, $\mathcal{A}$, $\boldsymbol{\Theta}$, and $\Pr(\boldsymbol{\Theta})$ are as before and $\mathcal{U} = \{u^1, \ldots, u^\rho\}$ is the set of *local payoff functions*, each of which involves only a subset of agents. Each u^e has *scope* $\mathbb{A}(u^e)$ specifying the subset of agents on which it depends.

When scopes are restricted, CGBGs can be represented much more compactly than regular cooperative BGs. In fact, if the largest scope has size k, then the representation size is exponential in k, but only linear in n. However, the joint type distribution must also be compact. A typical assumption is that the type probabilities factor as the product of individual type probabilities (Soni et al., 2007; Jiang and Leyton-Brown, 2010). More generally, it is common to represent $\Pr(\boldsymbol{\Theta})$ compactly (e.g., as in (Koller and Milch, 2003; Jiang and Leyton-Brown, 2010)) by means of Bayesian networks (Pearl, 1988; Bishop, 2006) or other graphical models.[2]

Agents in a CGBG aim to maximize the expected sum of local payoffs:

$$\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} \sum_{e \in \mathcal{E}} \sum_{\boldsymbol{\theta}_e \in \boldsymbol{\Theta}_e} \Pr(\boldsymbol{\theta}_e) u^e(\boldsymbol{\theta}_e, \boldsymbol{\beta}_e(\boldsymbol{\theta}_e)) \quad (2)$$

where $\boldsymbol{\theta}_e = \langle \theta_i \rangle_{i \in \mathbb{A}(u^e)}$ is the local joint type, $\boldsymbol{\beta}_e = \langle \beta_i \rangle_{i \in \mathbb{A}(u^e)}$ is the local joint policy, and $\boldsymbol{\beta}_e(\boldsymbol{\theta}_e)$ is the local joint action under $\boldsymbol{\beta}$ given $\boldsymbol{\theta}_e$.[3] The local value for the e-th payoff component is:

$$V^e(\boldsymbol{\beta}_e) = \sum_{\boldsymbol{\theta}_e \in \boldsymbol{\Theta}_e} \Pr(\boldsymbol{\theta}_e) u^e(\boldsymbol{\theta}_e, \boldsymbol{\beta}_e(\boldsymbol{\theta}_e)). \quad (3)$$

[2]While not all joint type distributions admit a compact representation, this is nonetheless a minimal assumption: there is no hope of solving other BGs efficiently, as they cannot even be represented efficiently. Fortunately, the class of real-world problems that admit a compact representation is likely to be quite large.

[3]We abuse notation such that e is an index into the set of local payoff functions and an element of the set of scopes.

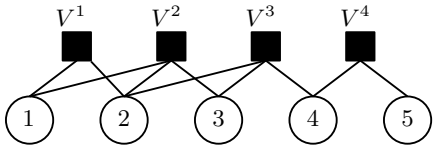


Figure 2: An AI factor graph with five agents. Circular nodes are agents and square nodes are local payoff functions, with edges indicating in which local payoff function each agent participates.

Each local value component can be interpreted as a factor in a *factor graph* (Kschischang et al., 2001; Loeliger, 2004), which generalizes a coordination graph to allow local payoff functions that depend on more than two agents. The resulting bipartite graph has one set of nodes for all the local value functions (the factors) and another set for all the agents (the variables, whose values correspond to policies), as shown in Fig. 2. A local value function V^e is connected to an agent i if and only if $i \in \mathbb{A}(u^e)$. We refer to this as an *agent independence (AI) factor graph*.

A naive solution approach is to simply enumerate all joint policies. However, the required time is $O(|\mathcal{A}_*|^{|\Theta_*|^n})$, where $|\mathcal{A}_*|$ and $|\Theta_*|$ are the size of the largest individual action and type set. Thus, it scales exponentially with the number of agents and types. Fortunately, the agent independence expressed in the factor graph of Fig. 2 can be exploited. For instance, Oliehoek et al. (2008b) apply NDP to this graph. Another option is to use Max-Sum message passing.

3 Representing Type Independence

Applying NDP or Max-Sum to an AI factor graph can reduce the exponential complexity with respect to the number of agents. However, it cannot reduce the exponential complexity with respect to the number of types because this complexity manifests itself in the number of values that the variables can take on.

The key observation motivating this work is that CGBGs also possess a second form of independence, which we call *type independence*, that enables solution approaches to avoid the exponential dependence on the number of types. Unlike agent independence, which only some BGs possess, type independence is an inherent property of all BGs because it is a direct consequence of the fact that the value of a joint policy is an expectation over joint types.

Type independence is expressed in terms of a *contribution* for each joint type:

$$C_{\boldsymbol{\theta}}(\mathbf{a}) \equiv \Pr(\boldsymbol{\theta}) u(\boldsymbol{\theta}, \mathbf{a}). \quad (4)$$

The value of a joint policy $\boldsymbol{\beta}$ can now be interpreted

as a sum of contributions, one for each joint type:

$$V(\boldsymbol{\beta}) = \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} C_{\boldsymbol{\theta}}(\boldsymbol{\beta}(\boldsymbol{\theta})). \quad (5)$$

Type independence results from the additive structure of a joint policy's value shown in (5). The key insight is that each contribution term depends only on the joint action selected when the corresponding joint type $\boldsymbol{\theta}$ occurs. Since that action is selected according to the joint policy $\boldsymbol{\beta}$, the contribution depends only on $\boldsymbol{\beta}(\boldsymbol{\theta})$, the part of the joint policy that specifies the joint action for $\boldsymbol{\theta}$.

Viewed from another perspective, type independence occurs because, in any game, only one individual type is realized for each agent, and that type affects only some contributions. In other words, the individual action $\beta_i(\theta_i)$ selected for type θ_i of agent i affects only those contributions whose joint types involve θ_i. For instance, in the firefighting problem illustrated in Fig. 1, one possible joint type is $\boldsymbol{\theta} = \langle N, N \rangle$ (neither agent observes flames). Clearly, the action $\beta_1(F)$ that agent 1 selects when it has type F (it observes flames), has no effect on the contribution of this joint type.

An optimal joint policy can also be described in terms of contributions, by simply maximizing the value as defined in (5):

$$\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} V(\boldsymbol{\beta}) = \arg\max_{\boldsymbol{\beta}} \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} C_{\boldsymbol{\theta}}(\boldsymbol{\beta}(\boldsymbol{\theta})). \quad (6)$$

Thus, the solution to a BG maximizes the sum of contributions, each of which has restricted scope. This is an important observation, since maximization of an additively factored function with components of restricted scope is exactly the operation that algorithms such as NDP and Max-Sum make more efficient.

To apply these methods, we represent type independence in a factor graph, by creating factors for all the contributions (corresponding to joint types) and variables for all the individual types of all the agents. Fig. 3 shows such a factor graph for the firefighting problem. Unlike the representation that results from reducing a BG to a strategic game played by *agent-type* combinations (Osborne and Rubinstein, 1994), this factor graph does not 'flatten' the utility function. On the contrary, it explicitly represents the contributions of each joint type, thereby capturing type independence.

However, while the factor graph in Fig. 3 represents type independence, it does not represent agent independence because the global payoff represented by the sum of contributions is not decomposed into local payoff functions. Conversely, the factor graph shown in Fig. 2 represents agent independence but not type independence because the value of each local payoff function is not decomposed into contributions. To solve CGBGs efficiently, we need a factor graph formulation that captures both agent and type independence.

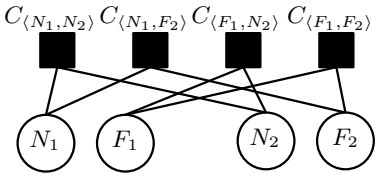


Figure 3: A factor graph for the firefighting problem that captures type independence, e.g., the action that agent 1 selects when it has type N affects only the contribution factors $C_{\langle N_1,N_2 \rangle}$ and $C_{\langle N_1,F_2 \rangle}$ in which it has that type.

To this end, we define a *local contribution* as follows:

$$C^e_{\boldsymbol{\theta}_e}(\mathbf{a}_e) \equiv \Pr(\boldsymbol{\theta}_e) u^e(\boldsymbol{\theta}_e, \mathbf{a}_e). \quad (7)$$

Using this notation, the solution of the CGBG is

$$\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} \sum_{e \in \mathcal{E}} \sum_{\boldsymbol{\theta}_e \in \boldsymbol{\Theta}_e} C^e_{\boldsymbol{\theta}_e}(\boldsymbol{\beta}_e(\boldsymbol{\theta}_e)). \quad (8)$$

Thus, the solution corresponds to the maximum of an additively decomposed function containing a contribution for each local joint type $\boldsymbol{\theta}_e$. This can be expressed in a factor graph in which an individual type θ_i of an agent i is connected to a contribution $C^e_{\boldsymbol{\theta}_e}$ if and only if i participates in u^e and $\boldsymbol{\theta}_e$ specifies θ_i for agent i, as illustrated in Fig. 4. We refer to this graph as the *agent and type independence (ATI) factor graph*. Contributions are separated, not only by the joint type to which they apply, but also by the local payoff function to which they contribute. Consequently, both agent and type independence are neatly expressed.

4 Solving ATI Factor Graphs

ATI factor graphs can be solved using existing methods such as NDP and Max-Sum. In this section, we analyze the computational complexity of doing so.

4.1 Non-Serial Dynamic Programming

NDP can compute the maximum configuration of a factor graph. In the *forward pass*, variables are eliminated one by one according to some prespecified order. Eliminating the kth variable v involves collecting all the factors in which it participates and replacing them with a new factor f^k that represents the sum of the removed factors, given that v selects a best response. Once all variables are eliminated, the *backward pass* iterates through the variables in reverse order of elimination. Each variable selects a best response to the

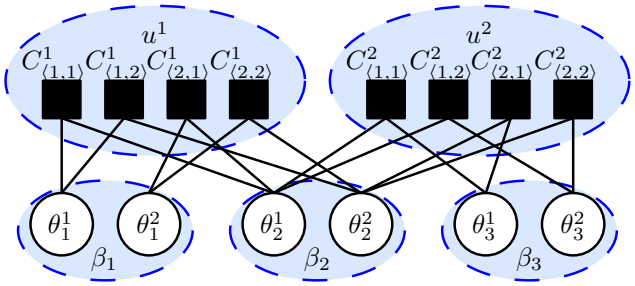


Figure 4: ATI factor graph for a CGBG with three agents, two types per agent, and two local payoff functions. Agents 1 and 2 participate in payoff function u^1 (corresponding to the first four contributions), while agents 2 and 3 participate in u^2 (corresponding to the last four contributions). The factor graph expresses both agent independence (e.g., agent 1 does not participate in u^2) and type independence (e.g, the action agent 1 selects when it receives observation θ^1_1 affects only the first 2 contributions).

variables already visited, eventually yielding an optimal joint policy.

The maximum number of agents participating in a factor encountered during NDP is known as the induced width w of the ordering. The induced width depends on the order in which the variables are eliminated and determining the optimal order is NP-complete (Arnborg et al., 1987). The following result is well-known (see for instance (Dechter, 1999)):

Theorem 1. NDP *requires exponential time and space in the induced width w.*

Though NDP is still exponential, for sparse problems the induced width is much smaller than the total number of variables V, i.e., $w \ll V$, leading to an exponential speedup over naively enumerating the joint variables. However, NDP scales poorly in practice on densely connected graphs (Kok and Vlassis, 2006). In addition, because of the particular shape that type independence induces on the ATI factor graph, we can establish the following:

Theorem 2. *The induced width of an ATI factor graph is lower bounded by the number of individual types and the size of the largest payoff function:*

$$w \geq (k-1) \cdot |\Theta_*|,$$

where k is the largest local scope $k = \max_{e \in \mathcal{E}} |\mathbb{A}(u^e)|$, and Θ_* denotes the smallest individual type set.

Proof. Assume u^e is a payoff function of maximal size k. At some point, NDP will eliminate the first variable connected to one of its contributions, i.e., a type of an agent $i \in \mathbb{A}(u^e)$. Since such a variable is connected to contributions for each profile $\boldsymbol{\theta}_{e \setminus i}$ of types of the other agents participating in u^e, the newly introduced factor will connect all the types of these $(k-1)$ other agents. The minimum number of types of an agent is $|\Theta_*|$ and thus the lower bound holds. □

Theorems 1 and 2 lead directly to the following observation:

Corollary 1. *The computational complexity of* NDP *applied to an ATI factor graph is exponential in the number of individual types.*

Therefore, even given the ATI factor graph formulation, it seems unlikely that NDP can effectively exploit type independence. In particular, we hypothesize that NDP will not perform significantly better when applied to the ATI factor graph instead of the AI factor graph. In fact, it is easy to construct an example where it performs worse.

4.2 Max-Sum

To more effectively exploit type independence, we can solve the ATI factor graph using the Max-Sum message passing algorithm. Max-Sum is an appealing choice for several reasons. First, it performs well in practice on structured problems (Murphy et al., 1999; Kschischang et al., 2001; Kok and Vlassis, 2006; Farinelli et al., 2008; Kuyer et al., 2008). Second, unlike NDP, it is an *anytime* algorithm that can provide results after each iteration, not only at the end (Kok and Vlassis, 2006). Third, as we show below, its computational complexity is exponential only in the size of the largest local payoff function's scope, which is fixed for many classes of CGBGs.

Max-Sum iteratively sends messages between the factors and variables. These messages encode how much payoff the sender expects to be able to contribute to the total payoff. In particular, a message sent from a type i to a contribution j encodes, for each possible action, the payoff it expects to contribute. This is computed as the sum of the incoming messages from other contributions. Similarly, a message sent from a contribution to a type i encodes the payoff it can contribute conditioned on each available action to the agent with type i.[4]

Max-Sum iteratively passes these messages over the edges of the factor graph. Within each iteration, the messages are sent either in parallel or sequentially with a fixed or random ordering. When run on an acyclic factor graph (i.e., a tree), it is guaranteed to converge to an optimal fixed point (Pearl, 1988; Wainwright

[4]For a detailed description of how the messages are computed, see Oliehoek (2010) or Rogers et al. (2011).

et al., 2004). In cyclic factor graphs, there are no guarantees that MAX-SUM will converge.[5] However, experimental results have demonstrated that it works well in practice even when cycles are present (Kschischang et al., 2001; Kok and Vlassis, 2006; Kuyer et al., 2008). This requires normalizing the messages to prevent them from growing ever larger, e.g., by taking a weighted sum of the new and old messages (damping).

Here we show that the computational complexity of one iteration of MAX-SUM on a CGBG is exponential only in the size of the largest local payoff function's scope. In general, it is not possible to bound the number of iterations, since MAX-SUM is not guaranteed to converge. However, MAX-SUM converges quickly in practice and, since it is an anytime algorithm, the number of iterations can be fixed in advance.

Theorem 3. *One iteration of* MAX-SUM *run on the factor graph constructed for a CGBG has cost*

$$O\left(|\mathcal{A}_*|^k \cdot k^2 \cdot \rho\rho_* |\Theta_*|^{2k-1}\right), \tag{9}$$

where ρ_* is the maximum number of edges in which an agent participates.

Proof. It follows directly from the cost of a message and the number of messages sent by factors and variables that the complexity of one iteration of MAX-SUM is

$$O\left(m^k \cdot k^2 \cdot l \cdot F\right), \tag{10}$$

where F is the number of factors, k is the maximum degree of a factor, l is the maximum degree of a variable, and m is the maximum number of values a variable can take.

Now, we interpret (10) for the CGBG setting. In this case, $m = |\mathcal{A}_*|$, $F = O(\rho \cdot |\Theta_*|^k)$ is the number of contributions, and $l = O(\rho_* \cdot |\Theta_*|^{k-1})$, since each edge $e \in \mathcal{E}$ in which a (type of an) agent participates induces $O(|\Theta_*|^{k-1})$ contributions to which it is connected. □

The implication of this theorem it that a MAX-SUM iteration is tractable for small k, the size of the largest local scope. For problems with bounded k, this therefore directly yields a polynomial-time approximate algorithm by bounding the number of iterations. Given these results, we expect that, in general, MAX-SUM will prove more effective than NDP at exploiting type independence. In particular, we hypothesize that MAX-SUM will outperform NDP when applied to an ATI factor graph and will outperform MAX-SUM when applied to an AI factor graph.

[5]However, some variants of the message passing approach have slight modifications that yield convergence guarantees (Globerson and Jaakkola, 2008). Since we found that regular MAX-SUM performs well in our experimental setting, we do not consider such variants here.

5 Experiments

To evaluate the efficacy of using ATI factor graphs, we compare **NDP-ATI**, NDP applied to an ATI factor graph and **Max-Sum-ATI**, MAX-SUM applied to an ATI factor graph with 10 restarts, to several state-of-the-art alternatives. In particular, we compare to: **BaGaBaB**, Bayesian Game Branch and Bound, a heuristic search method for optimally solving CBGs (Oliehoek et al., 2010); **AltMax**, Alternating maximization, starting with a random joint policy, each agent iteratively computes a best-response policy for each of its types, thus hill-climbing to a local optimum; **CE**, Cross Entropy optimization (de Boer et al., 2005) a randomized optimization method that maintains a distribution over joint policies; **MAID-CM**, the state-of-the-art continuation method for solving multiagent influence diagrams (Blum et al., 2006).[6] Apart from MAID-CM, methods were implemented using the MADP Toolbox (Spaan and Oliehoek, 2008); the NDP and MAX-SUM implementations also use LIBDAI (Mooij, 2008).

5.1 Random CGBG Experiments

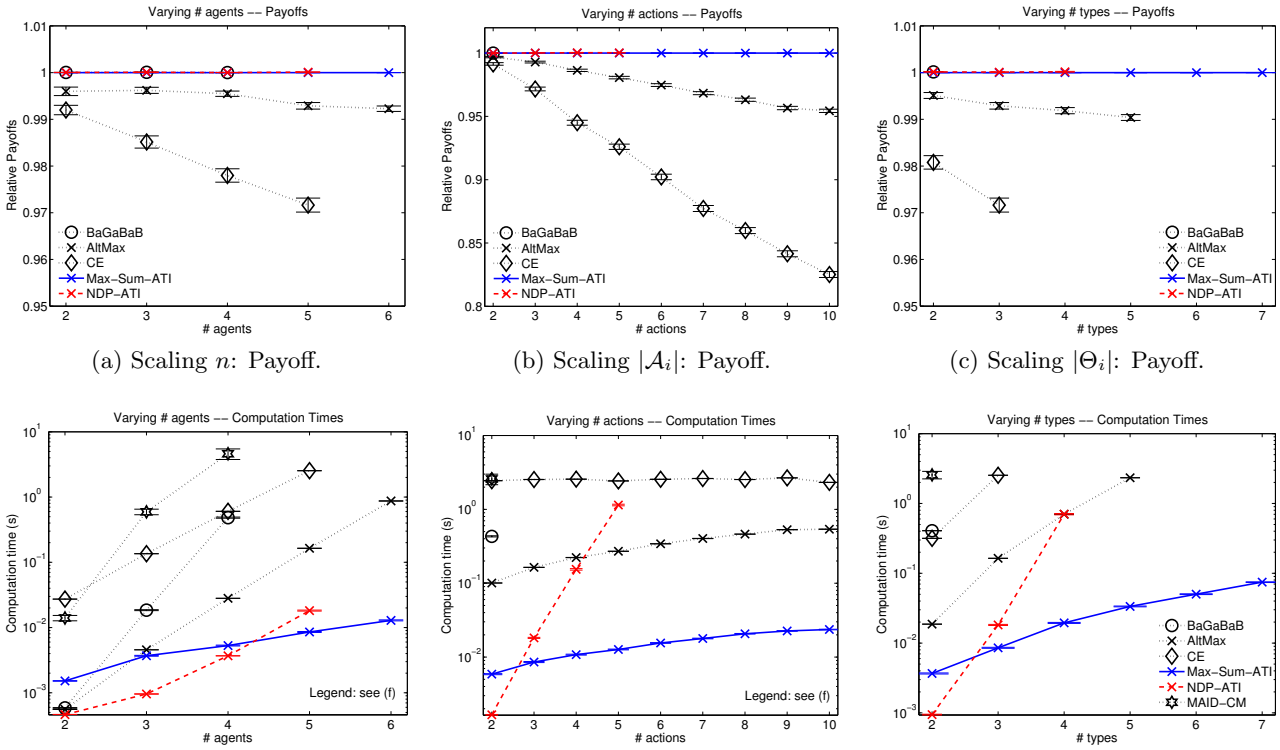
We first present experiments in randomly generated games following a procedure similar to that used by Kok and Vlassis (2006). We start with a set of n agents with no local payoff functions defined. As long as a path does not exist between every pair of agents, we add a local payoff function involving k agents. Payoffs $u^e(\boldsymbol{\theta}_e, \mathbf{a}_e)$ are drawn from a normal distribution $\mathcal{N}(0, 1)$, and the local joint type probabilities $\Pr(\boldsymbol{\theta}_e)$ are drawn from a uniform distribution and then normalized.

These random BGs enable testing on a range of problem parameters. In particular, we investigate the effect of scaling the number of agents n, the number of types for each agent $|\Theta_i|$, and the number of actions for each agent $|\mathcal{A}_i|$. We assume that the scopes of the local payoff functions, the individual action sets, and the individual type sets all have the same size for each agent. For each set of parameters, we generate 1,000 random CGBGs. Each method is run on these instances, limited by 1GB of memory and $5s$ computation time.[7] Payoffs are normalized with respect to those of MAX-SUM-ATI (whose payoff is always 1).

First, we compare NDP-ATI and MAX-SUM-ATI with other methods that do not exploit a factor graph rep-

[6]Since we use the implementation of Blum et al., which does not output the quality of the found solution, we report only computation times.

[7]A data point is not presented if the method exceeded the predefined resource limits on one or more test runs. MAID-CM was run on 30 games and given 30s.



(a) Scaling n: Payoff. (b) Scaling $|\mathcal{A}_i|$: Payoff. (c) Scaling $|\Theta_i|$: Payoff.

(d) Scaling n: Computation times. (e) Scaling $|\mathcal{A}_i|$: Computation times. (f) Scaling $|\Theta_i|$: Computation times.

Figure 5: Comparison of Max-Sum-ATI and NDP-ATI with other methods, scaling the number of agents ((a) and (d)), the number of actions ((b) and (e))), and the number of types ((c) and (f)). In all plots $k = 2$, and the parameters that are not being varied are set as follows: $n = 5, |\mathcal{A}_i| = 3, |\Theta_i| = 3$.

resentation explicitly, the results of which are shown in Fig. 5. These graphs show the payoff and computation time when increasing the number of agents (Fig. 5a and 5d), the number of actions (Fig. 5b and 5e), and the number of types (Fig. 5c and 5f). These results support a number of observations. 1) As predicted, NDP scales well with respect to the number of agents, but not types. It also scales poorly with the number of actions, which indicates that the encountered induced width is quite high in practice. 2) Though NDP cannot successfully exploit type independence, it is still consistently faster than BaGaBaB's heuristic search. 3) The approximate Max-Sum method always finds the optimum when we can compute it. 4) Max-Sum finds substantially better solutions than the other approximate methods AltMax and CE, while running faster than them for all but the smallest problems. 5) MAID-CM, the state-of-the-art for MAIDs, is significantly slower than the other methods.

To make explicit the advantage of exploiting type independence, Fig. 6 compares the performance of Max-Sum-ATI to that of Max-Sum-AI, i.e., Max-Sum applied to an AI factor graph, for games with $k = 2, |\Theta_i| = 4, |a_i| = 4$, larger numbers of agents, and $30s$ allowed per CGBG. Max-Sum-AI scales only to 50 agents, while Max-Sum-ATI scales to 725 (limited only by the allocated memory). While it is not possible to compute optimal solutions for these problems, values found by both methods are close and increase in proportion with the number of payoff components, indicating no degradation in performance.

While these results are for $k = 2$, experiments conducted for $k = 3$ were qualitatively similar, though lack of space prevents a detailed presentation. Note that there is no hope of efficiently solving problems with large k, since such problems cannot even be represented compactly. However, this not a serious practical limitation; on the contrary, even low values of k allow for complicated interactions since each agent may participate in many local payoff functions.

5.2 Generalized Firefighting Experiments

In this section, we aim to demonstrate that the advantages of Max-Sum-ATI extend to a more realistic problem, called Generalized Firefighting, which is like the two-agent firefighting problem of Fig. 1 but with N_H houses and n agents physically spread over a

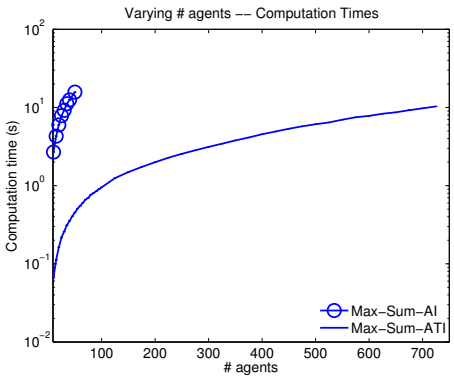


Figure 6: Comparison of the scaling behavior for many agents for MAX-SUM-AI and MAX-SUM-ATI.

2-dimensional area. Each agent observes the N_O nearest houses and may choose to fight fire at any of the N_A nearest houses.

Fig. 7 shows the results of our experiments in this domain, which corroborate our findings on random CGBGs by showing that MAX-SUM has the most desirable scaling behavior in a variety of different parameters.[8] We omit payoff plots because all methods computed solutions with the same payoff on all problem instances; MAX-SUM achieves the optimum in cases where we can compute it. Further investigation of solution quality (omitted due to lack of space) showed that solution quality decreases (there are only penalties in this problem) proportionally with the number of payoff components, again indicating that scalability does not come at the expense of solution quality.

6 Related Work

There is a large body of research that is related to the work presented in this paper. MAX-SUM has been used in decision making for teams of cooperative agents (Kok and Vlassis, 2006; Farinelli et al., 2008; Kuyer et al., 2008; Rogers et al., 2011). However, all of this work deals with cooperative games of perfect information, whereas we deal with imperfect information.

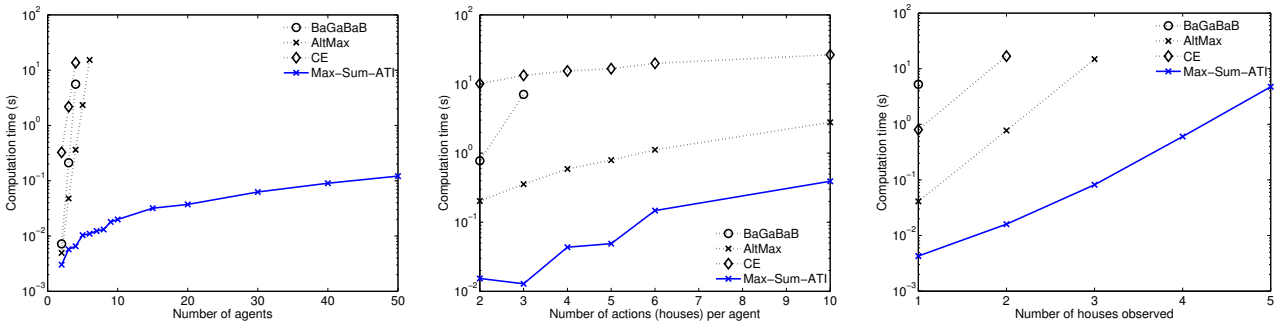
The CGBG model was introduced by Oliehoek et al. (2008b), who also used NDP to optimally solve CGBGs. However, NDP was applied only to the AI factor graph. Here we applied and analyzed application of NDP and MAX-SUM to a newly proposed ATI factor graph.

Two methods that are closely related to our approach are the heuristic search methods introduced by Oliehoek et al. (2010) and Kumar and Zilberstein (2010). The former propose BAGABAB for cooperative BGs, which we evaluated in our comparative experiments. It exploits additivity of the value function and can thus be interpreted as exploiting type independence. However, its performance depends heavily on the tightness of the heuristic used within BAGABAB (and, when problem sizes increase, the heuristic bounds tend to become looser, leading to disproportionate increases in runtime). Kumar and Zilberstein (2010) take a similar approach by performing search with a state-of-the-art heuristic (EDAC) for weighted constraint satisfaction problems to perform a point-based backup (as part of a solution method for sequential decision problems formalized as decentralized partially observable Markov decision processes, Dec-POMDPs). However, the EDAC heuristic provides leverage only in the two-agent case (de Givry et al., 2005). Furthermore, unlike our approach, neither of these methods exploits agent independence.

A closely related framework is the multiagent influence diagram (MAID), which extends decision diagrams to multiagent settings (Koller and Milch, 2003). A MAID represents a decision problem with a Bayesian network containing a set of chance nodes and, for each agent, a set of decision and utility nodes. As in a CGBG, the individual payoff function for each agent is defined as the sum of local payoffs. MAIDs are more general because they can represent sequential and non-identical payoff settings. Therefore, MAID solution methods are in principle applicable to CGBGs. However, these methods aim merely to find a sample Nash equilibrium, which is only guaranteed to be a local optimum. In contrast, our factor-graph formulation enables methods that find the global optimum. The MAID-CM method (Blum et al., 2006), which we also compared against, exploits the structure of the network in an inner loop to more efficiently compute the gradient of the payoff function. However, Blum et al. do not decompose the payoff into contributions and thus do not represent type independence. Moreover, since this inference is performed exactly using a clique-tree based approach, it cannot exploit type independence effectively, as the graphical argument made against NDP extends directly to this setting.

The framework of Bayesian action-graph games (BAGGs) can also model any BG (Jiang and Leyton-Brown, 2010). The BAGG solution method is similar to MAID-CM in that it exploits structure in an inner loop while searching for a sample equilibrium and thus suffers from the same limitations. This framework additionally allows the exploitation of anonymity and context-specific independence, but offers no advantages when these are not present.

[8] We omit NDP-ATI, since MAX-SUM-ATI outperformed it in random games, and MAID-CM, since it could not find solutions for any of these problems in 30s.



(a) Varying n. Each agent observes 2 houses (4 types per agent).

(b) Varying $|\mathcal{A}_i|$ when each agent observes 2 houses (4 types per agent).

(c) Varying number of houses that are observed per agent ($n = 4$, $|\mathcal{A}_i| = 5$).

Figure 7: Computation time results for the Generalized Fire Fighting problem.

7 Conclusions & Future Work

This paper considered the interaction of a team of agents under uncertainty modeled as cooperative graphical Bayesian games (CGBGs). We showed that CGBGs possess two different types of structure, agent and type independence, which can be neatly captured in a novel ATI factor graph formulation.

We considered two standard solution methods: non-serial dynamic programming (NDP) and Max-Sum message passing. The former has a computational complexity that is exponential in the induced tree width of the factor graph, which we proved to be lower bounded by the number of individual types. One iteration of the latter is tractable when there is enough independence between agents: we showed that it is exponential only in k, the maximum number of agents that participate in the same local payoff function.

By bounding the number of iterations, the combination of the ATI factor graph and Max-Sum yields, for any fixed k, a polynomial-time algorithm. While there are no quality guarantees, the complexity results suggest that any algorithm that scales polynomially with respect to the number of types (and thus, via our theorem, tree-width) *cannot* provide a solution within an arbitrary error bound (Kwisthout, 2011). It may be, however, be possible to obtain (non-adjustable) quality bounds for Max-Sum solutions via a technique that eliminates edges from the factor graph (Rogers et al., 2011).

An empirical evaluation showed that solution quality is high: in all our experiments Max-Sum found the optimal solution (when it could be computed). Moreover, the evaluation also revealed that exploiting both agent and type agent independence leads to significantly better performance than current state-of-the-art methods, providing scalability to much larger problems than was previously possible. In particular, we showed that this approach allows for the solution of coordination problems with imperfect information for up to 750 agents, limited only by a 1GB memory constraint. As such, we anticipate that the representation and exploitation of both agent and type independence will be a critical component in future solution methods for cooperative Bayesian games.

For future work, a possible extension of our approach could replace Max-Sum with message passing algorithms for belief propagation that are guaranteed to converge (Globerson and Jaakkola, 2008). In certain cases, such approaches can be shown to compute an optimal solution (Jebara, 2009). A different direction of study would be to use our approach to help solve Dec-POMDPs, by replacing the BG solving components used by Oliehoek et al. (2008b) and Kumar and Zilberstein (2010) with NDP or Max-Sum applied to the ATI factor graph. This can lead to huge scale-up in the number of agents approximate solutions can handle (Oliehoek, 2010). Similarly, it may be possible to improve methods like MAID-CM by performing the inference with Max-Sum in order to exploit type independence. Another avenue is to extend our algorithms to the non-cooperative case by rephrasing the task of finding a sample Nash equilibrium as one of minimizing regret, as suggested by Vickrey and Koller (2002).

Acknowledgements

We would like to thank Nikos Vlassis for extensive discussions, and Kevin Leyton-Brown, Mathijs de Weerdt, Christian Shelton, David Silver and Leslie Kaelbling for their valuable input. Research supported by AFOSR MURI project #FA9550-09-1-0538 and by NWO CATCH project #640.005.003. M.S. is funded by the FP7 Marie Curie Actions Individual Fellowship #275217 (FP7-PEOPLE-2010-IEF).

References

S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Methods*, 8(2):277–284, 1987.

U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, Inc., 1972.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

B. Blum, C. R. Shelton, and D. Koller. A continuation method for Nash equilibria in structured games. *Journal of Artificial Intelligence Research*, 25:457–502, 2006.

P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

L. Buşoniu, R. Babuška, and B. De Schutter. A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, Mar. 2008.

S. de Givry, F. Heras, M. Zytnicki, and J. Larrosa. Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 84–89, 2005.

R. Dechter. Bucket elimination: a unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, Sept. 1999.

R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 136–143, 2004.

A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 639–646, 2008.

A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *Advances in Neural Information Processing Systems 20*, 2008.

C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1523–1530, 2002.

J. C. Harsanyi. Games with incomplete information played by 'Bayesian' players, parts I, II and II. *Management Science*, 14:159–182, 320–334, 486–502, 1967–1968.

M. N. Huhns, editor. *Distributed Artificial Intelligence*. Pitman Publishing Ltd., 1987.

T. Jebara. MAP estimation, message passing, and perfect graphs. In *Proc. of Uncertainty in Artificial Intelligence*, 2009.

A. X. Jiang and K. Leyton-Brown. Bayesian action-graph games. In *Advances in Neural Information Processing Systems 23*, pages 991–999, 2010.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

M. J. Kearns. Graphical games. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, Sept. 2007.

M. J. Kearns, M. L. Littman, and S. P. Singh. Graphical models for game theory. In *Proc. of Uncertainty in Artificial Intelligence*, pages 253–260, 2001.

J. R. Kok and N. Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *RoboCup-2005: Robot Soccer World Cup IX*, Osaka, Japan, July 2005.

J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221, Oct. 2003.

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

A. Kumar and S. Zilberstein. Point-based backup for decentralized POMDPs: Complexity and new algorithms. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1315–1322, 2010.

L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Proc. of the European Conference on Machine Learning*, pages 656–671, 2008.

J. Kwisthout. Most probable explanations in Bayesian networks: Complexity and tractability. *International Journal of Approximate Reasoning*, 52(9), 2011.

H.-A. Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, 2004.

R. Marinescu and R. Dechter. And/or branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491, 2009.

J. M. Mooij. libDAI: library for discrete approximate inference, 2008. URL `http://www.jorismooij.nl/`.

K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proc. of Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

F. A. Oliehoek. *Value-based Planning for Teams of Agents in Stochastic Partially Observable Environments*. PhD thesis, Informatics Institute, University of Amsterdam, Feb. 2010.

F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353, 2008a.

F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 517–524, May 2008b.

F. A. Oliehoek, M. T. J. Spaan, J. S. Dibangoye, and C. Amato. Heuristic search for identical payoff Bayesian games. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1115–1122, May 2010.

M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, July 1994.

L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

J. Pearl. *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

A. Rogers, A. Farinelli, R. Stranders, and N. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2):730–759, 2011.

V. Soni, S. Singh, and M. Wellman. Constraint satisfaction algorithms for graphical games. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 423–430. ACM Press, 2007.

M. T. J. Spaan and F. A. Oliehoek. The MultiAgent Decision Process toolbox: software for decision-theoretic planning in multiagent systems. In *Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, pages 107–121, 2008.

M. T. J. Spaan, F. A. Oliehoek, and C. Amato. Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 2027–2032, 2011.

K. P. Sycara. Multiagent systems. *AI Magazine*, 19(2): 79–92, 1998.

J. Tsitsiklis and M. Athans. On the complexity of decentralized decision making and detection problems. *IEEE Transactions on Automatic Control*, 30(5):440–446, 1985.

D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *Proc. of the National Conference on Artificial Intelligence*, pages 345–351, 2002.

N. Vlassis. *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2007.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14(2):143–166, 2004.

A Case Study in Complexity Estimation: Towards Parallel Branch-and-Bound over Graphical Models

Lars Otten and **Rina Dechter**
Department of Computer Science
University of California, Irvine
{lotten,dechter}@ics.uci.edu

Abstract

We study the problem of complexity estimation in the context of parallelizing an advanced Branch and Bound-type algorithm over graphical models. The algorithm's pruning power makes load balancing, one crucial element of every distributed system, very challenging. We propose using a statistical regression model to identify and tackle disproportionally complex parallel subproblems, the cause of load imbalance, ahead of time. The proposed model is evaluated and analyzed on various levels and shown to yield robust predictions. We then demonstrate its effectiveness for load balancing in practice.

1 INTRODUCTION

This paper explores the application of learning for improved load balancing in the context of distributed search for discrete combinatorial optimization over graphical models (e.g., Bayesian networks, weighted CSPs). Specifically, we consider one of the best exact search algorithms for solving the MPE task over graphical models, AND/OR Branch and Bound (AOBB) [12], ranked first and third, respectively, in the UAI'06 and '08 evaluations and winning all three MPE categories of the 2011 PASCAL Inference Challenge.

We adapt the established concept of parallel tree search [7], where a search tree is explored centrally up to a certain depth and the remaining subtrees are solved in parallel. In the graphical model context we explore the search space of partial instantiations up to a certain point and solve the resulting conditioned subproblems in parallel.

The distributed framework is built with a grid computing environment in mind, i.e., a set of autonomous, loosely connected systems – notably, we cannot assume any kind of shared memory or dynamic load balancing which most parallel or distributed algorithms build upon [1, 5, 7, 6]. The primary challenge is therefore to determine a priori a set of subproblems with balanced complexity, so that the overall parallel runtime will not be dominated by just a few of them. In the optimization context, however, the use of cost and heuristic functions for pruning makes it very hard to reliably predict and balance subproblem complexity ahead of time; in particular, structural parameters like the induced width are not sufficient to differentiate subproblems.

Our suggested approach and the main contribution of this paper is to estimate subproblem complexity by learning a regression model over several subproblem parameters, some static and structural (e.g., induced width, variable domain sizes), others dynamically extracted at runtime (e.g. upper and lower bounds on the subproblem solution based on the heuristic function).

A similar regression-based approach was developed in [11] to predict the problem complexity (called "empirical hardness") of combinatorial auction instances; similarly the successful SAT solver *SATzilla* uses linear regression models to choose among a set of component solvers the one that is predicted to be fastest for a given SAT instance [16].

Other general work on estimating search complexity goes back to [10] and more recently [9], which predict the size of general backtrack trees through random probing. Similar schemes were devised for Branch and Bound algorithms [2], where search is run for a limited time and the partially explored tree is extrapolated. These approaches typically require a substantial amount of probing, which is prohibitively expensive in our setup, where many hundreds, if not thousands of subproblems need to be evaluated quickly.

The contribution of the present paper lies in proposing and studying a general learning approach for estimating subproblem complexity. In particular, we frame the problem as statistical regression analysis, which allows us to leverage established, powerful techniques from machine learning and statistics. Motivated by different parallelization scenarios, we distinguish three distinct levels of learning: based on a single problem instance, based on a specific class of problems, and based on a combination of problem

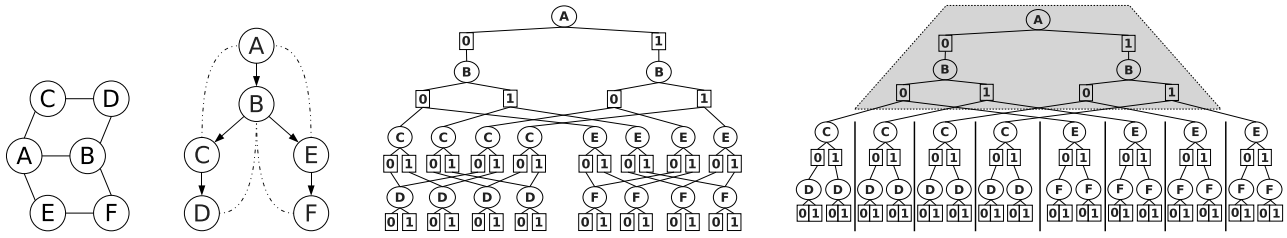


Figure 1: (a) Example primal graph with six variables, (b) its pseudo tree along ordering A, B, C, D, E, F, (c) the corresponding context-minimal AND/OR search graph, and (d) the parallel search space resulting from parallelizing at depth $d = 2$ with eight independent subproblems.

classes. We evaluate, analyze, and contrast these three levels on a sample set of more than 11,000 subproblem samples from four problem classes and demonstrate generally robust prediction performance. We also demonstrate empirically the model's potential for improved load balancing.

The remainder of the paper is organized as follows: Section 2 summarizes the necessary background and outlines the distributed AND/OR Branch and Bound algorithm. Section 3 introduces the proposed regression model for complexity estimation while Section 4 evaluates it on a variety of instances from several problem classes. Section 5 provides selected parallel results that highlight the benefits of the proposed model and Section 6 concludes.

2 BACKGROUND

We assume the usual definitions of a *graphical model* as a set of functions $F = \{f_1, \ldots, f_m\}$ over discrete variables $X = \{X_1, \ldots, X_n\}$, its *primal graph*, *induced graph*, and *induced width*.

2.1 AND/OR SEARCH SPACES

The concept of AND/OR search spaces has been introduced as a unifying framework for advanced algorithmic schemes for graphical models to better capture the structure of the underlying graph [3]. Its main virtue consists in exploiting conditional independencies between variables, which can lead to exponential speedups. The search space is defined using a *pseudo tree*, which captures problem decomposition:

DEFINITION **1 (pseudo tree)** *Given an undirected graph $G = (X, E)$, a* pseudo tree *of G is a directed, rooted tree $\mathcal{T} = (X, E')$ with the same set of nodes X, such that every arc of G that is not included in E' is a back-arc in $\mathcal{T}$, namely it connects a node in $\mathcal{T}$ to an ancestor in $\mathcal{T}$. The arcs in E' may not all be included in E.*

AND/OR Search Trees and Graphs : Given a graphical model instance with variables X and functions F, its primal graph (X, E), and a pseudo tree $\mathcal{T}$, the associated *AND/OR search tree* consists of alternating levels of OR and AND nodes. The structure of the AND/OR search tree is based on the underlying pseudo tree $\mathcal{T}$: the root of the AND/OR search tree is an OR node labeled with the root of $\mathcal{T}$. The children of an OR node X_i are AND nodes labeled with assignments $\langle X_i, x_i \rangle$; the children of an AND node $\langle X_i, x_i \rangle$ are OR nodes labeled with the children of X_i in $\mathcal{T}$, representing conditionally independent subproblems. Different nodes may root identical subproblems and can be merged through *caching*, yielding an *AND/OR search graph* of smaller size, at the expense of using additional memory during search.

Given a graphical model, its primal graph G, and a guiding pseudo tree $\mathcal{T}$ of height h, the size of the AND/OR search tree is $\mathcal{O}(n \cdot k^h)$, while $\mathcal{O}(n \cdot k^{w^*})$ bounds the AND/OR search graph, where w^* is the induced width of G over a depth-first traversal of $\mathcal{T}$ and k bounds the domain size [3]. Figure 1(a) shows an example problem primal graph with six variables, Figure 1(b) depicts a pseudo tree along ordering A, B, C, D, E, F. Figure 1(c) shows the corresponding AND/OR search graph.

AND/OR Branch and Bound : AND/OR Branch and Bound (AOBB) is a state-of-the-art algorithm for solving optimization problems over graphical models. Assuming maximization, it traverses the AND/OR graph in a depth-first manner while keeping track of a current lower bound on the optimal solution cost. During expansion of a node n, this lower bound l is compared with a heuristic upper bound $u(n)$ on the optimal solution below n – if $u(n) \leq l$ the algorithm can prune the subproblem below n [12].

Mini-Bucket Heuristics : The heuristic $h(n)$ that we use in our experiments is the Mini-Bucket heuristic. It is based on Mini-Bucket elimination, an approximate variant of a variable elimination scheme that computes approximations to reasoning problems over graphical models [4]. A control parameter i allows to trade accuracy of the heuristic against its time and space requirements. It was shown that the intermediate functions generated by the Mini-Bucket algorithm MBE(i) can be used to derive a heuristic function that un-

derestimates the minimal cost solution to a subproblem in the AND/OR search graph [12].

2.2 DISTRIBUTED AOBB & LOAD BALANCING

Our distributed implementation of AND/OR Branch and Bound draws from the notion of parallel tree search [7, 6], where a search tree is explored centrally up to a certain depth and the remaining subtrees are solved in parallel. Applied to the search graph from Figure 1(c), for instance, we could obtain eight independent subproblems as shown in Figure 1(d), with a conditioning search space (in gray) spanning the first two levels (variables A and B).

We refer to the boundary between conditioning search space and parallel subproblems as the *parallelization frontier*. Its choice determines the shape and the number of subproblems and is thus crucial for effective parallel *load balancing*. Namely, it is known that for best parallel performance we should spread the parallel workload evenly across all available CPUs, while minimizing overhead. Note that we assume independent worker machines, with limited or very costly communication, hence dynamic load balancing at runtime (cf. [6]) is not applicable.

Algorithm 1 shows pseudo code for our parallelization policy: the parallelization frontier is generated in a breadth-first manner by iteratively selecting the current most complex subproblem, estimated by a complexity estimator $\hat{N}$, and splitting it into its immediate "sub-subproblems", which are in turn added to the frontier. This process is repeated until a desired number of subproblems is obtained, at which point all subproblems are submitted to the distributed environment.

In the context of depth-first Branch and Bound, however, determining the most complex subproblem is extremely difficult and elusive. Due to the pruning power of the algorithm, subproblem runtimes can differ greatly, even when the underlying subgraph structure and the associated asymptotic complexity guarantees (exponential in the induced width of the AND/OR subspace) are identical.

To illustrate, consider the subproblem statistics of two parallel runs shown in Figure 2, where instead the parallelization frontier is placed at a fixed depth $d = 5$ and $d = 4$, respectively, yielding 64 and 144 subproblems (the horizontal axis). In each case we see significant variance in subproblem runtime. In fact, the overall runtime is dominated exclusively by the handful of longest-running subproblems, with most other subproblems finishing long before (note the log scale). Detecting and mitigating this imbalance ahead of time constitutes the central challenge in this line of work, as we elaborate in the next sections.

Algorithm 1 Finding the parallelization frontier

Input: Pseudo tree $\mathcal{T}$ with root X_0, subproblem count p, subproblem complexity estimator $\hat{N}$.
Output: Set F of subproblem root nodes with $|F| \geq p$.

```
1: F ← {⟨X_0⟩}
2: while |F| < p :
3:     n' ← arg max_{n∈F} N̂(n)
4:     F ← F \ {n'}
5:     F ← F ∪ children(n')
```

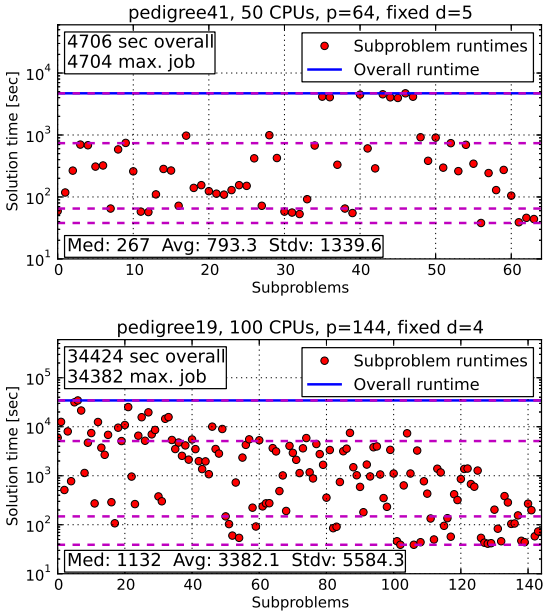


Figure 2: Subproblem statistics for fixed-depth parallelization frontier showing large variance in subproblem runtime. Dashed lines mark 0, 20, 80, and 100 percentile.

3 LEARNING COMPLEXITIES THROUGH REGRESSION

This section introduces our learning approach to subproblem complexity prediction through regression analysis. Previous work has investigated and evaluated various methods for balancing subproblem complexity, directly formulating metrics using human expert knowledge [13, 14]. These metrics were relative in nature, i.e., they only allowed comparison of one subproblem to another within a given overall problem instance. In contrast, the present work does not depend as heavily on expert knowledge and gives absolute complexity estimates.

3.1 GENERAL METHODOLOGY

We identify a subproblem by its search space root node n. We further measure the complexity of the subproblem rooted at n through the size of its explored search space, which is the number of node expansions required for its solution, denoted $N(n)$. We then aim to capture the exponential nature of the search space size by modeling $N(n)$

as an exponential function of various subproblem features $\phi_i(n)$ as follows:

$$N(n) = \exp(\sum_i \lambda_i \phi_i(n)) \quad (1)$$

The exponent has been chosen as a sum so that we can consider the log complexity and obtain the following:

$$\log N(n) = \sum_i \lambda_i \phi_i(n) \quad (2)$$

Given a set of m sample subproblems, finding suitable parameter values λ_j can thus be formulated as a well-known *linear regression* problem, with the *mean squared error* (MSE) as the loss function $L(\lambda)$ we aim to minimize:

$$L(\lambda) = \frac{1}{m} \sum_{k=1}^{m} \Big(\sum_i \lambda_i \phi_i(n_k) - \log N(n_k) \Big)^2 \quad (3)$$

The MSE captures how well the learned regression model fits the training data. In the context of load balancing for parallelism we can consider a secondary metric, the *Pearson correlation coefficient* (PCC), which is simply the normalized covariance between the vector of subproblem complexities and their estimates, normalized by the product of each vector's standard deviation. It is bounded by $[-1, 1]$, where 1 implies perfect linear correlation and -1 anticorrelation. Hence a value close to 1 is desirable, as it signifies a model likely to correctly identify the hardest subproblems.

3.2 SUBPROBLEM FEATURES

Table 1 lists the full set of basic subproblem features ϕ_i that we consider. This list was compiled based on our prior knowledge of what aspects can affect problem complexity. Features can be divided into two distinct classes: "static", which can be precompiled from the problem graph and pseudo tree, and "dynamic" which are computed at runtime, as the parallelization frontier decision is made (note that none of the dynamic features are costly to compute).

Subproblem variable statistics (static):	
1:	Number of variables in subproblem.
2-6:	Min, Max, mean, average, and std. dev. of variable domain sizes in subproblem.
Pseudotree depth/leaf statistics (static):	
7:	Depth of subproblem root in overall search space.
8-12:	Min, max, mean, average, and std. dev. of depth of subproblem pseudo tree leaf nodes, counted from subproblem root.
13:	Number of leaf nodes in subproblem pseudo tree.
Pseudo tree width statistics (static):	
14-18:	Min, max, mean, average, and std. dev. of induced width of variables within subproblem.
19-23:	Min, max, mean, average, and std. dev. of induced width of variables within subproblem, *when conditioning on subproblem root conditioning set*.
Subproblem cost bounds (dynamic):	
24:	Lower bound L on subproblem solution cost, derived from current best overall solution.
25:	Upper bound U on subproblem solution cost, provided by mini bucket heuristics.
26:	Difference $U - L$ between upper and lower bound, expressing "constrainedness" of the subproblem.
Pruning ratios (dynamic), based on running 5000 node expansion probe of AOBB:	
27:	Ratio of nodes pruned using the heuristic.
28:	Ratio of nodes pruned due of determinism (zero probabilities, e.g.)
29:	Ratio of nodes corresponding to pseudo tree leaf.
Sample statistics (dynamic), based on running 5000 node expansion probe of AOBB:	
30:	Average depth of terminal search nodes within probe.
31:	Average node depth within probe (denoted $\bar{d}$).
32:	Average branching degree, defined as $\sqrt[\bar{d}]{5000}$.
Various (static):	
33:	Mini bucket i-bound parameter.
34:	Max. subproblem variable context size minus mini bucket i-bound.

Table 1: Subproblem features for complexity estimation.

3.3 SUBPROBLEM SAMPLE DOMAINS

In order to evaluate training and prediction error of the proposed complexity model from a statistical learning perspective, we need to specify the sample domain over which we will make predictions, for which we aim to generate a model, and from which subproblem samples are assumed to be drawn. In fact, in the following we consider three incrementally more general levels of sample domains and learning, corresponding to three different designs in the context of parallelizing AOBB:

1. **Learning per problem instance:** The sample domain is all subproblems from a single problem instance. This corresponds to learning a new complexity model for every problem instance the parallel scheme encounters (e.g., we would learn separate models for the two instances in Figure 2).

2. **Learning per problem class:** Take the domain to be all subproblems of problems from a specific class. In the parallelization context we learn a separate model for every problem class we consider. For example, we would learn a single model for all pedigree problems, but a different model for other problem classes like protein sidechain prediction.

3. **Learning across problem classes:** Take the sample domain to be all subproblems of all problems from several classes. Ultimately this could translate to a parallel scheme that uses a single complexity model for all problem classes under consideration.

These three levels are increasingly more general and thus potentially more challenging for robust estimation. On the other hand, they require increasingly less computational ef-

fort, since fewer distinct models need to be learned. Lastly, they can present different trade-offs between pre-compiled off-line learning and learning at runtime.

3.4 REGRESSION ALGORITHMS

We investigated a number of algorithms for fitting a linear model. Ordinary least squares (OLS) regression was problematic due to numerical issues (near-singular matrix inversion) and prone to overfitting (due to lack of regularization) and we did not consider if further. Standard ridge regression adds the L_2-norm of the parameter vector λ to the regularized loss function through a term $\alpha(\sum_i \lambda_i^2)^{\frac{1}{2}}$; similarly, lasso regression [15], places an L_1-penalty on the parameter vector by adding the term $\alpha \sum_i |\lambda_i|$. The so-called "Elastic Net" combines both penalty terms [18]. In each case we followed the common approach of determining the regularization parameter α once through initial cross validation and held it fixed subsequently.

In our experiments we found all methods to perform similarly in terms of training and prediction errors, with a slight advantage for the lasso method. We will therefore focus on lasso learning. This method has the additional benefit of "built-in" feature selection: learned models are relatively sparse and thus compact, because the L_1-regularization pushes many parameters λ_i to zero [15].

3.5 NON-LINEAR REGRESSION

In addition to the purely linear regression analysis proposed above, we also explored non-linear approaches. In particular, we took inspiration from [11], which reports improved prediction performance using *quadratic feature expansion*, albeit in the context of combinatorial auctions. Quadratic feature expansion, also referred to as "quadratic regression", works by adding new features in the form of pairwise products of the original features; namely, for every pair of subproblem features ϕ_i, ϕ_j with $i \leq j$, we create a new feature $\phi_i \cdot \phi_j$. We then perform linear regression on the expanded feature set (629 in our case), thereby effectively fitting a polynomial of 2nd degree. Results will be outlined in Section 4.5.

Next we evaluate the proposed regression model on a variety of instances from several different problem classes.

4 EVALUATION AND ANALYSIS

The basis for our evaluation are 31 hard problem instances from four classes: pedigree haplotyping problems, protein side-chain prediction ([17], named "pdb"), "large family" genetic linkage instances, and grid networks ("75-2x-x"). Summary statistics of the different problem classes are given in Table 2. We note that all instances each take several hours, if not days to solve using sequential AOBB.

domain	M	n	k	w	h
pedigree	13	437 – 1272	3 – 7	17 – 39	47 – 102
pdb	5	103 – 172	81	10 – 15	24 – 43
largeFam	8	2569 – 3730	3 – 4	28 – 37	73 – 108
grid	5	624 – 675	2	37 – 39	111 – 124

Table 2: Summary statistics for problem classes used, M gives the number of instances in the class. n denotes number of problem variables, k max. domain size, w induced width, h pseudo tree height.

To compile a set of subproblem samples we revisit experiments with fixed-depth parallelization (cf. Section 2.2): we randomly choose not more than 500 subproblems from a previously recorded fixed-depth parallel run for each instance. This leaves us with about 11,500 sample subproblems (approx. 40% pedigree, 25% protein, 25% largeFam, 10% grids), which is very reasonable for the number of features we have (the variance of the trained linear model scales with p/m, where p is the number of features and m the number of samples, cf. Section 7.3 in [8]).

The empirical evaluation is organized as follows: Sections 4.1 through 4.3 assess the prediction power of our proposed linear regression complexity model according to the three levels of learning outlined in Section 3.3. Section 4.4 inspects feature informativeness and Section 4.5 briefly investigates performance of the quadratic model. Section 4.6 provides a summary of the learning results.

Throughout this section results are presented as a log-log scatter plot of actual versus predicted complexities, each also containing mean squared prediction error ("MSE", on the test set) and Pearson correlation coefficient ("PCC") as well as mean squared training error ("TER").

4.1 LEARNING PER PROBLEM INSTANCE

This first set of experiments is meant to determine the prediction quality of a regression model that is learned for a single instance only. To that end, we consider all subproblem samples from a given problem instance and apply 5-fold cross validation (i.e., partition the samples into 5 subsets, then predict the complexities of each subset by learning a model on the remaining four).

Figure 3 presents scatter plots for six problem instances from the different problem classes considered. We see that results are good for the protein and largeFam instance and still acceptable for *pedigree19* with slightly higher MSE. *Pedigree41* has a relatively low MSE and good PCC, in spite of the plot's flat appearance. In case of the grid instance *75-26-9* the model's discriminatory power is likely limited by the small number of subproblem samples in this case. Finally, we note that the training error ("TER") is very close to the prediction error ("MSE") in all cases, indicating the absence of overfitting.

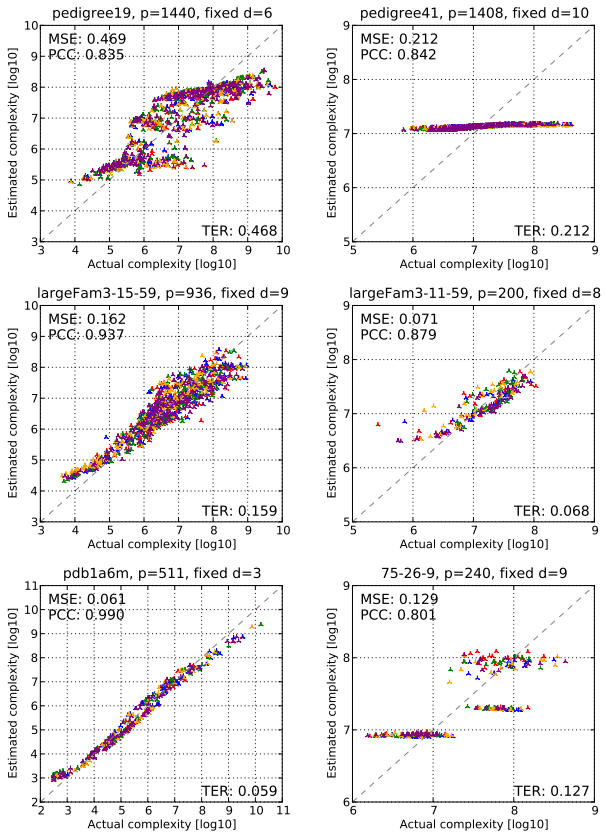


Figure 3: Actual vs. predicted subproblem complexity when learning per problem instance, using 5-fold cross validation.

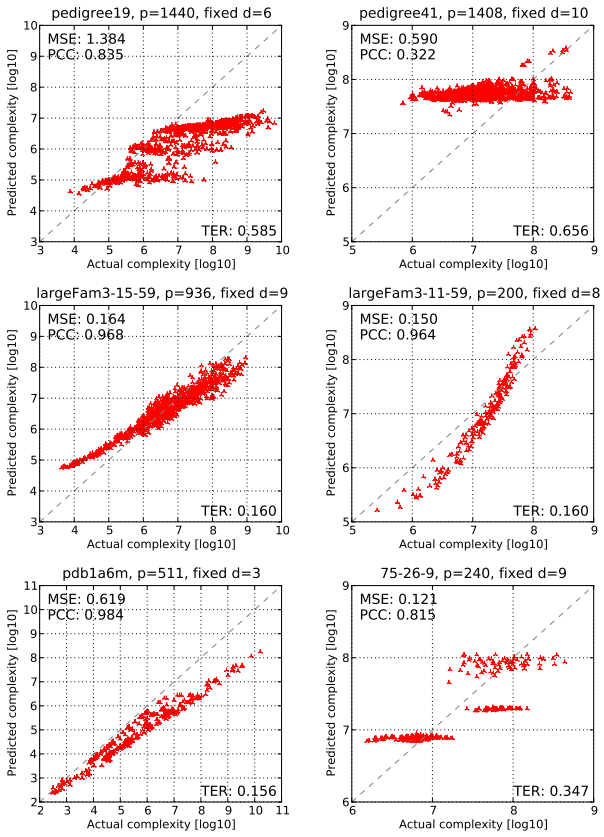


Figure 4: Actual vs. predicted subproblem complexity when learning per problem class.

4.2 LEARNING PER PROBLEM CLASS

Secondly, we aim to assess how well we can learn a model for an entire problem class. That is, we learn only once from sample subproblems of instances from the problem class in question. For testing we perform cross validation on the level of problem instances. Namely, we predict the subproblems of a given instance by fitting a model using subproblem samples of other instances from the same class – but not of the test instance itself.

Results are shown in Figure 4. Compared to Figure 3 in the previous section, estimates for the grid and largeFam instance are very similar and yield almost the same mean squared error. MSE increases for the pedigree and largeFam instances, but the PCC and overall shape of the predictions also remain similar, with the exception of *pedigree41* which sees both MSE and PCC deteriorate.

4.3 LEARNING MULTIPLE PROBLEM CLASSES

Lastly, we investigate how good a model we can learn from subproblems of instances across multiple problem classes. In particular, given a problem instance we learn a regression model on the subproblems of all other instances, regardless of their problem class.

Results are given in Figure 5, analogous to Figures 3 and 4. The two pedigree problems see an improved MSE and PCC (significantly for *pedigree41*), but the other instances suffer from a slightly larger prediction error. However, we again note that the overall shape of the plots remains roughly linear, which is also captured by high PCC values.

4.4 MOST INFORMATIVE FEATURES

Linear regression has the advantage that the resulting models can be straightforward to interpret. Namely, to assess the informativeness of feature ϕ_i we simply look at the absolute value of its coefficient λ_i in the regression model. Assuming a normalized sample set, features with larger absolute values contribute more to the predictions and are thus intuitively more informative.

In addition, recall that the L_1-regularization in lasso regression implicitly performs feature selection by assigning $\lambda_i = 0$ for some i. In our case, training on the entire sample set (11,500 subproblem instances, regularization parameter through cross-validation) yielded non-zero λ_i for nine features, as shown in in Table 3. In addition each feature's

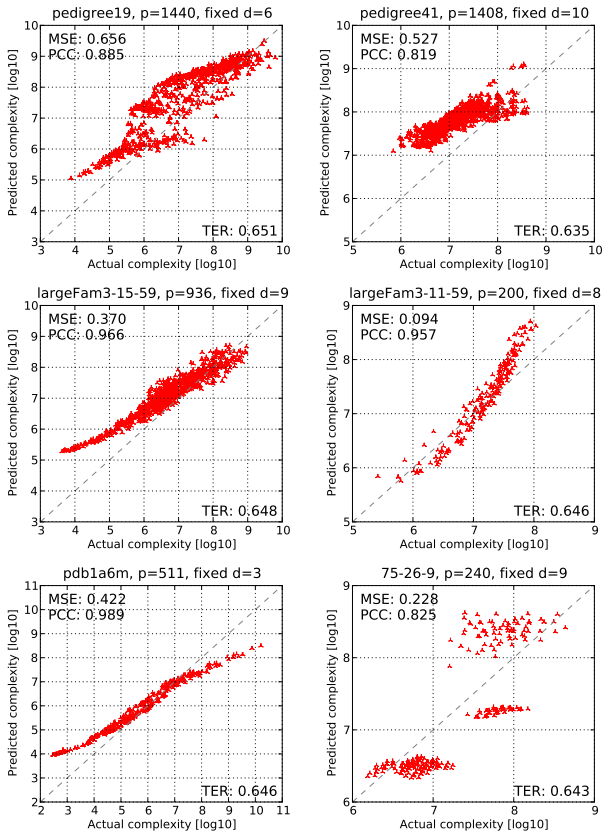


Figure 5: Actual vs. predicted subproblem complexity when learning across problem classes.

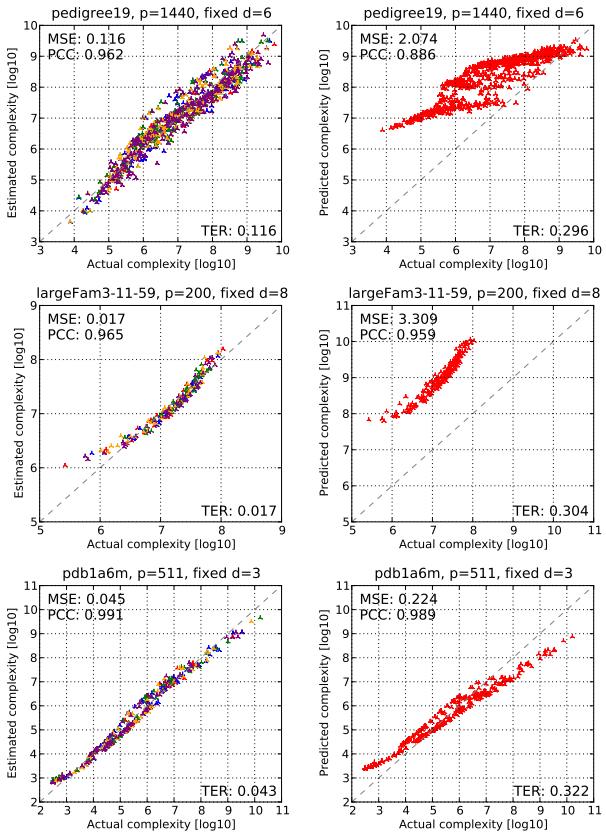


Figure 6: Example prediction results using a quadratic regression model. Left: learning per problem instance, using 5-fold cross validation (cf. Fig. 3). Right: learning from all problem classes (cf. Fig. 5).

cost of omission ("coo") as defined in [11] is given, which measures the normalized difference between the prediction error of the model with all nine features and the prediction error of a model trained with the respective feature omitted (using 5-fold cross-validation in all cases).

The particular set of features can be somewhat misleading, however, since lasso regression tends to pick only one of several highly correlated features. Yet it is useful to gain a conceptual understanding. In particular, we observe that the most informative features are dynamic, extracted from a limited AOBB probe or based on the initial subproblem bounds. Only the fifth feature, max. subproblem context size minus mini-bucket i-bound, is static, with a normalized cost of omission of only 16. This ties in to Section 2.2, where we observed that the asymptotic complexity bound of AOBB (based on static parameters) yields little information in this context.

Feature ϕ_i	$\lvert\lambda_i\rvert$	coo
Average branching degree in probe	0.57	100
Average leaf node depth in probe	0.39	87
Subproblem upper bound minus lower bound	0.22	17
Ratio of nodes pruned by heuristic in probe	0.20	27
Max. context size minus mini bucket i-bound	0.19	16
Ratio of leaf nodes in probe	0.18	10
Subproblem upper bound	0.11	7
Std. dev. of subproblem pseudo tree leaf depth	0.06	2
Depth of subproblem root node in overall space	0.05	2

Table 3: Features present in the linear model trained by lasso regression, with their model coefficients λ_i and their cost of omission "coo" (normalized).

4.5 NON-LINEAR REGRESSION

Here we briefly summarize results from our investigation of quadratic feature expansion, as detailed in Section 3.5. Selected prediction results for three instances are shown in Figure 6. On the left are results of learning per problem instances, on the right we plot the prediction accuracy when learning from all problem classes.

Comparing the plots on the left (per-problem learning) with Figure 3, we note that quadratic regression does a bit better than linear regression in terms of MSE and very similarly with regards to PCC. In contrast to [11], however, we find deteriorated prediction performance when comparing the plots on the right with Figure 5: while the PCC value is

similar, the mean squared prediction error when learning from multiple problem classes increases considerably for *pedigree19* and *largeFam3-11-59*. Notably, however, the training error remains fairly low in both cases, which is indicative of overfitting. And indeed, with over 600 subproblem features and just 31 different instances, the quadratic regression model is likely to pick up specific characteristics of each instance that hurt its predictive performance.

Since quadratic models are also more expensive to train and lack the straightforward interpretability of a linear model, we feel that the latter is better-suited for our purposes.

4.6 INTERPRETATION OF RESULTS

We have trained and evaluated our proposed regression model on the three levels of learning laid out in Section 3.3, trading off between the wider applicability of the learned models and the challenges of capturing increasingly general sample sets. Learning per problem instance provided a good baseline but has limited relevance in practice, since each new instance requires extensive sampling of subproblems to train on. Learning per problem class is more reasonable as the learned model can be reused within the given problem class; our experiments showed good performance. Finally, learning across classes is the most challenging as the sample set is likely to be more diverse and have higher variance, requiring more training samples; however, once we learn a model it can be used throughout.

And indeed, our results in Section 4.3 show that, given our

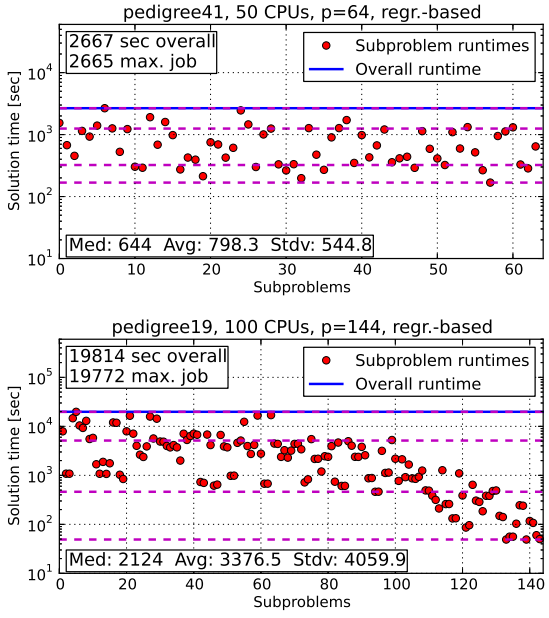


Figure 7: Subproblem statistics for regression-based parallelization (cf. fixed-depth parallelization in Fig. 2), p denotes the number of subproblems.

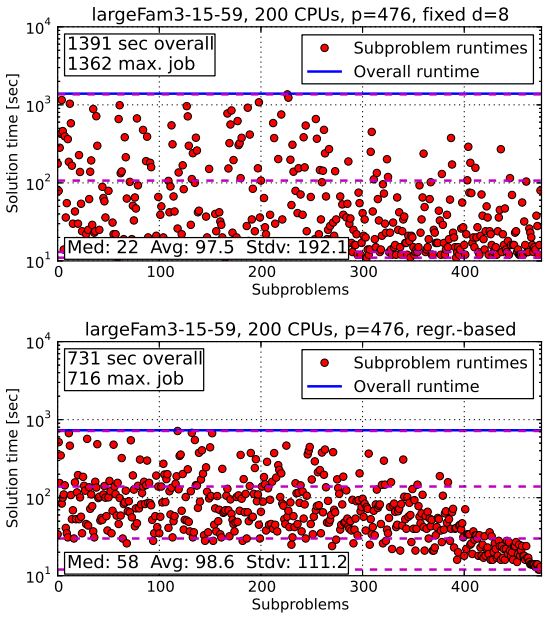


Figure 8: Subproblem statistics for fixed-depth (top) and regression-based (bottom) parallelization on *largeFam3-15-59* instance.

substantial set of 11,500 subproblem samples, the model can accommodate this most general level of learning across problem classes without a noticeable penalty, at least for the current collection of problem classes. A model learned across classes is therefore also the basis for the next section, where we demonstrate the benefit of robust complexity estimates in the context of distributed AOBB.

5 REGRESSION-BASED LOAD BALANCING IN PRACTICE

In this section we present selected experimental results that show the potential of the proposed regression models in guiding the parallelization process, as described in Section 2.2 – a comprehensive empirical evaluation of Distributed AOBB is beyond the scope of this paper. As noted above, the regression model used for experiments in this section was learned at the most general level, using all available problem classes as discussed in Section 4.3 (but always excluding the test problem instance).

5.1 IMPROVING LOAD BALANCING

To demonstrate the profound impact the complexity predictions can have on the load balancing of the parallel scheme, we revisit the two parallel experiments presented in Section 2.2, Figure 2. In both cases the overall performance was heavily dominated by very few long-running subproblems.

Figure 7 shows runtime statistics for parallel execution on

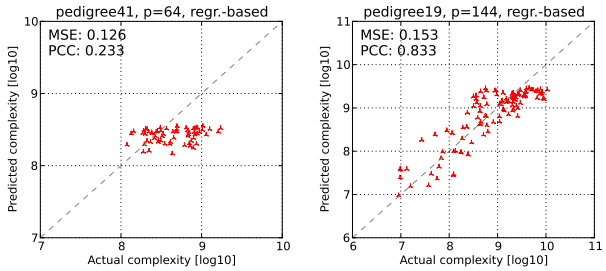


Figure 9: Actual vs. predicted subproblem complexity from the two parallel executions in Figure 7.

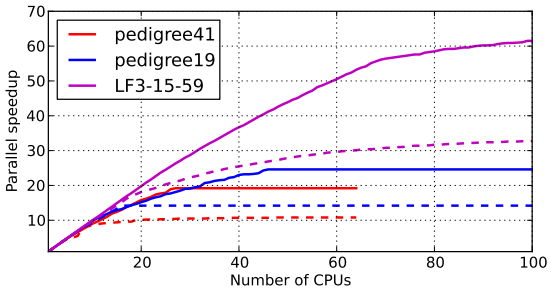


Figure 10: Parallel speedup for *pedigree41*, *pedigree19*, and *largeFam3-15-59* (cf. Figures 2, 7, and 8) as a function of the number of parallel CPUs. Dashed lines represent fixed-depth parallelization, solid lines correspond to parallel runs guided by our regression model.

these instances, using the regression model for load balancing. Figure 8 gives an additional example on a largeFam instance. In all cases we see that the max. subproblem runtime has been reduced greatly, close to 50% for *largeFam3-15-59* (1,362 to 716 seconds for "max. job" in Figure 8). We also note the drastically lower standard deviation in subproblem runtimes.

In addition, Figure 9 compares the complexity estimates obtained during the parallel execution with the actual recorded values. In both cases we observe good prediction error and *pedigree19* in particular also shows good PCC.

5.2 FACILITATING PARALLEL SPEEDUP

Figure 10 plots the parallel speedup for the three problem instances considered in Section 5.1 over the number of parallel CPUs. For each instance we show (as a dashed line) the speedup when using fixed-depth parallelization and (with a solid line) the speedup of the parallel execution guided by the regression model.

Comparing each instance's two entries reveals a clear advantage for the regression-based parallelization: it achieves higher speedups, roughly by a factor of two, and seems to plateau later, i.e. it is able to utilize a larger number of parallel resources.

On the other hand, most curves appear to level off well before their theoretical limit. This indicates that further improvements are possible, even though "perfect" speedup is unattainable in practice, since splitting a given subproblem often yields components of widely varying size and large jumps in complexity.

One way to mitigate these issues lies in increasing the subproblem granularity, i.e., setting the number of subproblems to match several times the number of parallel CPUs. However, this may add overhead in the general distributed context and redundancies in the particular graphical model context, which can negate potential gains in extreme cases. Indeed, finding the right balance in granularity is a central research issue in the field of distributed computing.

6 CONCLUSION & FUTURE WORK

We have presented a case study of complexity estimation in the context of parallelizing the state-of-the-art sequential optimization algorithm AND/OR Branch and Bound. The pruning power of the algorithm makes parallel load balancing very challenging, leading to inefficiencies in practice.

To address these symptoms we have proposed to employ statistical regression analysis in order to identify bottlenecks for parallel performance ahead of time. In particular, we developed a linear regression model that uses a variety of static as well as dynamic features to predict a subproblem's complexity, enabling us to detect and split problematic subproblems.

We identified three distinct levels of learning and evaluated our proposed model accordingly, using more than 11,000 subproblem samples from 31 problem instances and four problem classes. Results were good throughout, with generally low prediction error and high correlation coefficients.

In the context of our parallel scheme, we have shown how the regression model can enable more effective load balancing and improved parallel speedup. This last set of results, however, also outlined opportunities for further improvements and future research, including varying the parallel granularity.

Future work with respect to learning of complexity estimates will expand to more instances and additional problem classes. In that context we also plan to investigate how our learned models perform on instances from previously unseen problem classes. Furthermore, we are trying to devise more subproblem features, for instance extracted directly from the cost function tables within a subproblem.

ACKNOWLEDGEMENTS

Work supported in part by NSF grants IIS-0713118, IIS-1065618 and NIH grant 5R01HG004175-03.

References

[1] Geoffrey Chu, Christian Schulte, and Peter J. Stuckey. Confidence-based work stealing in parallel constraint programming. In *CP*, pages 226–241, 2009.

[2] Gérard Cornuéjols, Miroslav Karamanov, and Yanjun Li. Early estimates of the size of branch-and-bound trees. *INFORMS Journal on Computing*, 18(1):86–96, 2006.

[3] Rina Dechter and Robert Mateescu. AND/OR search spaces for graphical models. *Artif. Intell.*, 171(2-3):73–106, 2007.

[4] Rina Dechter and Irina Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM*, 50(2):107–153, 2003.

[5] Bernard Gendron and Teodor Gabriel Crainic. Parallel branch-and-bound algorithms: Survey and synthesis. *Operations Research*, 42(6):1042–1066, 1994.

[6] Ananth Grama, George Karypis, Vipin Kumar, and Anshul Gupta. *Introduction to Parallel Computing*. Addison Wesley, 2003.

[7] Ananth Grama and Vipin Kumar. State of the art in parallel search techniques for discrete optimization problems. *IEEE Trans. Knowl. Data Eng.*, 11(1):28–35, 1999.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 5th corrected edition, 2011.

[9] Philip Kilby, John Slaney, Sylvie Thiébaux, and Toby Walsh. Estimating search tree size. In *AAAI*, pages 1014–1019. AAAI Press, 2006.

[10] Donald E. Knuth. Estimating the efficiency of backtrack programs. *Mathematics of Computation*, 29(129):121–136, 1975.

[11] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM*, 56(4):1–52, 2009.

[12] Radu Marinescu and Rina Dechter. AND/OR Branch-and-Bound search for combinatorial optimization in graphical models. *Artif. Intell.*, 173(16-17):1457–1491, 2009.

[13] Lars Otten and Rina Dechter. Towards parallel search for optimization in graphical models. In *ISAIM*, 2010.

[14] Lars Otten and Rina Dechter. Finding most likely haplotypes in general pedigrees through parallel search with dynamic load balancing. In *PSB*, 2011.

[15] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

[16] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, June 2008.

[17] Chen Yanover and Yair Weiss. Approximate inference and protein-folding. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1457–1464. MIT Press, Cambridge, MA, 2003.

[18] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67(2):301–320, 2005.

A Spectral Algorithm for Latent Junction Trees

Ankur P. Parikh
Carnegie Mellon University
apparikh@cs.cmu.edu

Le Song
Georgia Tech
lsong@cc.gatech.edu

Mariya Ishteva
Georgia Tech
mariya.ishteva@cc.gatech.edu

Gabi Teodoru
Gatsby Unit, UCL
gabi.teodoru@gmail.com

Eric P. Xing
Carnegie Mellon University
epxing@cs.cmu.edu

Abstract

Latent variable models are an elegant framework for capturing rich probabilistic dependencies in many applications. However, current approaches typically parametrize these models using conditional probability tables, and learning relies predominantly on local search heuristics such as Expectation Maximization. Using tensor algebra, we propose an alternative parameterization of latent variable models (where the model structures are junction trees) that still allows for computation of marginals among observed variables. While this novel representation leads to a moderate increase in the number of parameters for junction trees of low treewidth, it lets us design a local-minimum-free algorithm for learning this parameterization. The main computation of the algorithm involves only tensor operations and SVDs which can be orders of magnitude faster than EM algorithms for large datasets. To our knowledge, this is the first provably consistent parameter learning technique for a large class of low-treewidth latent graphical models beyond trees. We demonstrate the advantages of our method on synthetic and real datasets.

1 Introduction

Latent variable models such as Hidden Markov Models (HMMs) (Rabiner & Juang, 1986), and Latent Dirichlet Allocation (Blei et al., 2003) have become a popular framework for modeling complex dependencies among variables. A latent variable can represent an abstract concept (such as topic or state), thus enriching the dependency structure among the observed variables while simultaneously allowing for a more tractable representation. Typically, a latent variable model is parameterized by a set of conditional probability tables (CPTs) each associated with an edge in the latent graph structure. For instance, an HMM can be parametrized compactly by a transition probability table and an observation probability table. By summing out the latent variables in the HMM, we obtain a fully connected graphical model for the observed variables.

Although the parametrization of latent variable models using CPTs is very compact, parameters in this representation can be difficult to learn. Compared to parameter learning in fully observed models which is either of closed form or convex (Koller & Friedman, 2009), most parameter learning algorithms for latent variable models resort to maximizing a non-convex objective via Expectation Maximization (EM) (Dempster et al., 1977). EM can get trapped in local optima and has slow convergence.

While EM explicitly learns the CPTs of a latent variable model, in many cases the goal of the model is primarily for prediction and thus the actual latent parameters are not needed. One example is determining splicing sites in DNA sequences (Asuncion & Newman, 2007). One can build a different latent variable model, such as an HMM, for each type of splice site from training data. A new sequence is then classified by determining which model it is most likely to have been generated by. Other examples include supervised topic modelling such as (Blei & McAuliffe, 2007; Lacoste-Julien et al., 2008; Zhu et al., 2009) and collaborative filtering (Su & Khoshgoftaar, 2009).

In these cases, it is natural to ask whether there exists an alternative representation/parameterization of a latent variable model where parameter learning can be done consistently and the representation remains tractable for inference among the observed variables. This question has been tackled recently by Hsu et al. (2009), Balle et al. (2011), and Parikh et al. (2011) who proposed spectral algorithms for local-minimum-free learning of HMMs, finite state transducers, and latent tree graphical models respectively. Unlike traditional parameter learning algorithms such as EM, spectral algorithms do not directly learn the CPTs of

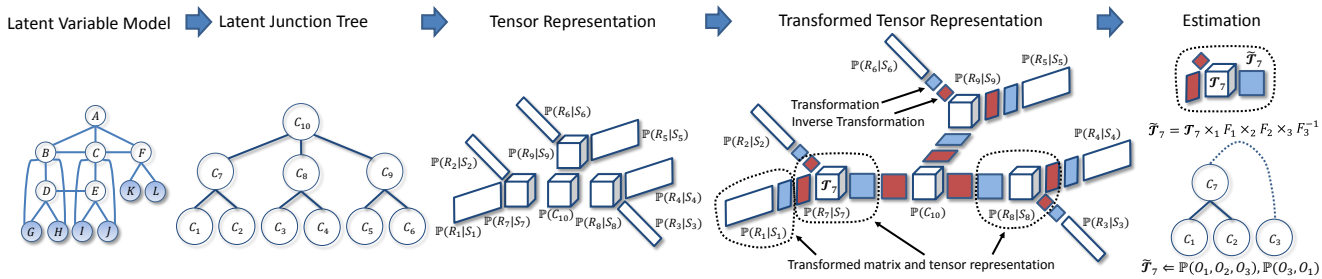


Figure 1: Our algorithm for local-minimum-free learning of latent variable models consist of four major steps. (1) First, we transform a model into a junction tree, such that each node in the junction tree corresponds to a maximal clique of variables in the triangulated graph of the original model. (2) Then we embed the clique potentials of the junction tree into higher order tensors and express the marginal distribution of the observed variables as a tensor-tensor/matrix multiplication according to the message passing algorithm. (3) Next we transform the tensor representation by inserting a pair of transformations between those tensor-tensor/matrix operations. Each pair of transformations is chosen so that they are inversions of each other. (4) Lastly, we show that each transformed representation is a function of only observed variables. Thus, we can estimate each individual transformed tensor quantity using samples from observed variables.

a latent variable model. Instead they learn an alternative parameterization (called the *observable representation*) which generally contains a larger number of parameters than the CPTs, but where computing observed marginals is still tractable. Moreover, these alternative parameters have the advantage that they only depend on observed variables and can therefore be directly estimated from data. Thus, parameter learning in the alternative representation is fast, local-minimum-free, and provably consistent. Furthermore, spectral algorithms can be generalized to nonparametric latent models (Song et al., 2010, 2011) where it is difficult to run EM.

However, existing spectral algorithms apply only to restricted latent structures (HMMs and latent trees), while latent structures beyond trees, such as higher order HMMs (Kundu et al., 1989), factorial HMMs (Ghahramani & Jordan, 1997) and Dynamic Bayesian Networks (Murphy, 2002), are needed and have been proven useful in many real world problems. The challenges for generalizing spectral algorithms to general latent structured models include the larger factors, more complicated conditional independence structures, and the need to sum out multiple variables simultaneously.

The goal of this paper is to develop a new representation for latent variable models with structures beyond trees, and design a spectral algorithm for learning this representation. We will focus on latent junction trees; thus the algorithm is suitable for both directed and undirected models which can be transformed into junction trees. Concurrently to our work, Cohen et al. (2012) proposed a spectral algorithm for Latent Probabilistic Context Free Grammars (PCFGs). Latent PCFGs are not trees, but have many tree-like properties, and so the representation Cohen et al. (2012) propose does not easily extend to other non-tree models such as higher order/factorial HMMs that we consider here. Our more general approach requires more complex tensor operations, such as multi-mode inversion, that are not used in the latent PCFG case.

The key idea of our approach is to embed the clique potentials of the junction tree into higher order tensors such that the computation of the marginal probability of observed variables can be carried out via tensor operations. While this novel representation leads only to a moderate increase in the number parameters for junction trees of low treewidth, it allows us to design an algorithm that can recover a transformed version of the tensor parameterization and ensure that the joint probability of observed variables are computed correctly and consistently. The main computation of the algorithm involves only tensor operations and singular value decompositions (hence the name "spectral") which can be orders of magnitude faster than EM algorithms in large datasets. To our knowledge, this is the first provably consistent parameter learning technique for a large class of low-treewidth latent graphical models beyond trees. In our experiments with large scale synthetic datasets, we show that our spectral algorithm can be almost 2 orders of magnitude faster than EM while at the same achieving considerably better accuracy. Our spectral algorithm also achieves comparable accuracy to EM on real data.

Organization of paper. A high level overview of our approach is given in Figure 1. We first provide some background on tensor algebra and latent junction trees. We then derive the spectral algorithm by representing junction tree message passing with tensor operations, and then transform this representation into one that only depends on observed variables. Finally, we analyze the sample complexity of our method and evaluate it on synthetic and real datasets.

2 Tensor Notation

We first give an introduction to the tensor notation tailored to this paper. An Nth order tensor is a multiway array with N "modes", *i.e.*, N indices

$\{i_1, i_2, \ldots, i_N\}$ are needed to access its entries. Subarrays of a tensor are formed when a subset of the indices is fixed, and we use a colon to denote all elements of a mode. For instance, $\boldsymbol{\mathcal{A}}(i_1, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N)$ are all elements in the nth mode of a tensor $\boldsymbol{\mathcal{A}}$ with indices from the other $N-1$ modes fixed to $\{i_1, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N\}$ respectively. Furthermore, we also use the shorthand $\boldsymbol{i}_{p:q} = \{i_p, i_{p+1}, \ldots, i_{q-1}, i_q\}$ for consecutive indices, *e.g.*, $\boldsymbol{\mathcal{A}}(i_1, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N) = \boldsymbol{\mathcal{A}}(\boldsymbol{i}_{1:n-1}, :, \boldsymbol{i}_{n+1:N})$.

Labeling tensor modes with variables. In contrast to the conventional tensor notation such as the one described in Kolda & Bader (2009), the ordering of the modes of a tensors will not be essential in this paper. We will use random variables to *label* the modes of a tensor: each mode will correspond to a random variable and what is important is to keep track of this correspondence. Therefore, we think two tensors are equivalent if they have the same set of labels and they can be obtained from each other by a permutation of the modes for which the labels are aligned.

In the matrix case this translates to $\boldsymbol{A}$ and $\boldsymbol{A}^\top$ being equivalent in the sense that $\boldsymbol{A}^\top$ carries the same information as $\boldsymbol{A}$, as long as we remember that the rows of $\boldsymbol{A}^\top$ are the columns of $\boldsymbol{A}$ and vice versa. We will use the following notation to denote this equivalence

$$\boldsymbol{A} \cong \boldsymbol{A}^\top \tag{1}$$

Under this notation, the dimension (or the size) of a mode labeled by variable X will be the same as the number of possible values for variable X. Furthermore, when we multiply two tensors together, we will always carry out the operation along (a set of) modes with matching labels.

Tensor multiplication with mode labels. Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be an Nth order tensor and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ be an Mth order tensor. If X is a common mode label for both $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ (w.l.o.g. we assume that this is the first mode, implying also that $I_1 = J_1$), multiplying along this mode will give

$$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_X \boldsymbol{\mathcal{B}} \quad \in \quad \mathbb{R}^{I_2 \times \cdots \times I_N \times J_2 \times \cdots \times J_M}, \tag{2}$$

where the entries of $\boldsymbol{\mathcal{C}}$ is defined as

$$\boldsymbol{\mathcal{C}}(\boldsymbol{i}_{2:N}, \boldsymbol{j}_{2:M}) = \sum\nolimits_{i=1}^{I_1} \boldsymbol{\mathcal{A}}(i, \boldsymbol{i}_{2:N})\, \boldsymbol{\mathcal{B}}(i, \boldsymbol{j}_{2:M})$$

Similarly, we can multiply two tensors along multiple modes. Let $\boldsymbol{\sigma} = \{X_1, \ldots, X_k\}$ be an arbitrary set of k modes (k variables) shared by $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ (w.l.o.g. we assume these labels correspond to the first k modes, and $I_1 = J_1, \ldots, I_k = J_k$ holds for the corresponding dimensions). Then multiplying $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ along $\boldsymbol{\sigma}$ results in

$$\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{A}} \times_{\boldsymbol{\sigma}} \boldsymbol{\mathcal{B}} \quad \in \quad \mathbb{R}^{I_{k+1} \times \ldots \times I_N \times J_{k+1} \times \ldots \times J_M}, \tag{3}$$

where the entries of $\boldsymbol{\mathcal{D}}$ are defined as

$$\boldsymbol{\mathcal{D}}(\boldsymbol{i}_{k+1:N}, \boldsymbol{j}_{k+1:M}) = \sum_{\boldsymbol{i}_{1:k}} \boldsymbol{\mathcal{A}}(\boldsymbol{i}_{1:k}, \boldsymbol{i}_{k+1:N}) \boldsymbol{\mathcal{B}}(\boldsymbol{i}_{1:k}, \boldsymbol{j}_{k+1:M}).$$

Multi-mode multiplication can also be interpreted as reshaping the $\boldsymbol{\sigma}$ modes of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ into a single mode and doing single-mode tensor multiplication. Furthermore, tensor multiplication with labels is symmetric in its arguments, *i.e.*, $\boldsymbol{\mathcal{A}} \times_{\boldsymbol{\sigma}} \boldsymbol{\mathcal{B}} \cong \boldsymbol{\mathcal{B}} \times_{\boldsymbol{\sigma}} \boldsymbol{\mathcal{A}}$.

Mode-specific identity tensor. We now define our notion of identity tensor with respect to a set of modes $\boldsymbol{\sigma} = \{X_1, \ldots, X_K\}$. Let $\boldsymbol{\mathcal{A}}$ be a tensor with mode labels containing $\boldsymbol{\sigma}$, and $\boldsymbol{\mathcal{I}}_{\boldsymbol{\sigma}}$ be a tensor with $2K$ modes with mode labels $\{X_1, \ldots, X_K, X_1, \ldots, X_K\}$. Then $\boldsymbol{\mathcal{I}}_{\boldsymbol{\sigma}}$ is an identity tensor with respect to modes $\boldsymbol{\sigma}$ if

$$\boldsymbol{\mathcal{A}} \times_{\boldsymbol{\sigma}} \boldsymbol{\mathcal{I}}_{\boldsymbol{\sigma}} \cong \boldsymbol{\mathcal{A}}. \tag{4}$$

One can also understand $\boldsymbol{\mathcal{I}}_{\boldsymbol{\sigma}}$ using its matrix representation: flattening $\boldsymbol{\mathcal{I}}_{\boldsymbol{\sigma}}$ with respect to $\boldsymbol{\sigma}$ (the first $\boldsymbol{\sigma}$ modes mapped to rows and the second $\boldsymbol{\sigma}$ modes mapped to columns) results in an identity matrix.

Mode-specific tensor inversion. Let $\boldsymbol{\mathcal{F}}, \boldsymbol{\mathcal{F}}^{-1} \in \mathbb{R}^{I_1 \times \cdots \times I_K \times I_{K+1} \times \cdots \times I_{K+K'}}$ be tensors of order $K + K'$, and both have two sets of mode labels $\boldsymbol{\sigma} = \{X_1, \ldots, X_K\}$ and $\boldsymbol{\omega}' = \{X_{K+1}, \ldots, X_{K+K'}\}$. Then $\boldsymbol{\mathcal{F}}^{-1}$ is the inverse of $\boldsymbol{\mathcal{F}}$ w.r.t. modes $\boldsymbol{\omega}$ if and only if

$$\boldsymbol{\mathcal{F}} \times_{\boldsymbol{\omega}} \boldsymbol{\mathcal{F}}^{-1} \cong \boldsymbol{\mathcal{I}}_{\boldsymbol{\sigma}}. \tag{5}$$

Multimode inversion can also be interpreted as reshaping $\boldsymbol{\mathcal{F}}$ with respect to $\boldsymbol{\omega}$ into a matrix of size $(I_1 \ldots I_K) \times (I_{K+1} \ldots I_{K+K'})$, taking the inverse, and then rearranging back into a tensor. Thus the existence and uniqueness of this inverse can be characterized by the rank of the matricized version of $\boldsymbol{\mathcal{F}}$.

Mode-specific diagonal tensors. We use $\boldsymbol{\delta}$ to denote an N-way relation: its entry $\boldsymbol{\delta}(\boldsymbol{i}_{1:N})$ at position $\boldsymbol{i}_{1:N}$ equals 1 when all indexes are the same ($i_1 = i_2 = \ldots = i_N$), and 0 otherwise. We will use $\oslash_d$ to denote repetition of an index d times. For instance, we use $\mathbb{P}(\oslash_d X)$ to denote a dth order tensor where its entries at $(\boldsymbol{i}_{1:d})$th position are specified by $\boldsymbol{\delta}(\boldsymbol{i}_{1:d})\mathbb{P}(X = x_{i_1})$. A diagonal matrix with its diagonal equal to $\mathbb{P}(X)$ is then denoted as $\mathbb{P}(\oslash_2 X)$. Similarly, we can define a $(d + d')$th order tensor $\mathbb{P}(\oslash_d X | \oslash_{d'} Y)$ where its $(\boldsymbol{i}_{1:d}\boldsymbol{j}_{1:d'})$th entry corresponds to $\boldsymbol{\delta}(\boldsymbol{i}_{1:d})\boldsymbol{\delta}(\boldsymbol{j}_{1:d'})\mathbb{P}(X = x_{i_1} | Y = y_{j_1})$.

3 Latent Junction Trees

In this paper, we will focus on discrete latent variable models where the number of states, k_h, for each hidden variable is much smaller than the number of states, k_o, for each observed variable. Uppercase letters denote random variables (*e.g.*, X_i) and lowercase letters their instantiations (*e.g.*, x_i). A latent variable model defines a joint probability distribution over a set of variables $\mathscr{X} = \mathscr{O} \cup \mathscr{H}$. Here, $\mathscr{O}$ denotes the set of observed variables, $\{X_1, \ldots, X_{|\mathscr{O}|}\}$. $\mathscr{H}$ denotes the set of hidden variables, $\{X_{|\mathscr{O}|+1}, \ldots, X_{|\mathscr{H}|+|\mathscr{O}|}\}$.

We will focus on latent variable models where the

structure of the model is a junction tree of low treewidth (Cowell et al., 1999). Each node C_i in a junction tree corresponds to a subset (clique) of variables from the original graphical model. We will also use C_i to denote the collection of variables contained in the node, *i.e.* $C_i \subset \mathscr{X}$. Let $\mathbb{C}$ denote the set of all clique nodes. The treewidth is then the size of a largest clique in a junction tree minus one, that is $t = \max_{C_i \in \mathbb{C}} |C_i| - 1$. Furthermore, we associate each edge in a junction tree with a separator set $S_{ij} := C_i \cap C_j$ which contains the common variables of the two cliques C_i and C_j it is connected to. If we condition on all variables in any S_{ij}, the variables on different sides of S_{ij} will become independent.

Without loss of generality, we assume that each internal clique node in the junction tree has exactly 3 neighbors.[1] Then we can pick a clique C_r as the root of the tree and reorient all edges away from the root to induce a topological ordering of the clique nodes. Given the ordering, the root node will have 3 children nodes, denoted as C_{r_1}, C_{r_2} and C_{r_3}. Each other internal node C_i will have a unique parent node, denoted as C_{i_0}, and 2 children nodes denoted as C_{i_1} and C_{i_2}. Each leaf node C_l is only connected with its unique parent node C_{l_0}. Furthermore, we can simplify the notation for the separator set between a node C_i and its parent C_{i_0} as $S_i = C_i \cap C_{i_0}$, omitting the index for the parent node. Then the remainder set of a node is defined as $R_i = C_i \setminus S_i$. We also assume w.l.o.g. that if C_i is a leaf in the junction tree, R_i consists of only observed variables. We will use r_i to denote an instantiation of the set of variables in R_i. See Figure 2 for an illustration of notation.

Given a root and a topological ordering of the nodes in a junction tree, the joint distribution of all variables $\mathscr{X}$ can be factorized according to

$$\mathbb{P}(\mathscr{X}) = \prod_{i=1}^{|\mathscr{X}|} \mathbb{P}(R_i|S_i), \tag{6}$$

where each CPT $\mathbb{P}(R_i|S_i)$, also called a clique potential, corresponds to a node C_i. The number of parameters needed to specify the model is $O(|\mathbb{C}|k_o^t)$, linear in the number of cliques but exponential in the tree width t. Then the marginal distribution of the observed variables can be obtained by summing over the latent variables,

$$\mathbb{P}(\mathscr{O}) = \sum_{X_{|\mathscr{O}|+1}} \cdots \sum_{X_{|\mathscr{O}|+|\mathscr{H}|}} \left[\prod_{i=1}^{|\mathscr{X}|} \mathbb{P}(R_i|S_i) \right], \tag{7}$$

where we use $\sum_X \phi(X)$ to denote summation over all possible instantiations of $\phi(x)$ w.r.t. variable X. Note that each (non-leaf) remainder set R_i contains a small subset of all latent variables. The presence of latent variables introduces complicated dependency between observed variables, while at the same time only a small number of parameters corresponding to the entries in the CPTs are needed to specify the model.

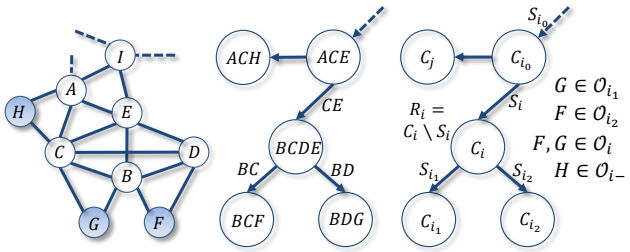


Figure 2: Example latent variable models with variables $\mathscr{X} = \{A, B, C, D, E, F, G, H, I, \ldots\}$, the observed variables are $\mathscr{O} = \{F, G, H, \ldots\}$ (only partially drawn). Its corresponding junction tree is shown in the middle panel. Corresponding to this junction tree, we also show the general notation for it in the rightmost panel.

The process of eliminating the latent variables in (7) can be carried out efficiently via message passing. More specifically, the summation can be broken up into local computation for each node in the junction tree. Each node only needs to sum out a small number of variables and then the intermediate result, called the message, is passed to its parent for further processing. In the end the root node incorporates all messages from its children and produces the final result $P(\mathscr{O})$. The local summation step, called the message update, can be generically written as[2]

$$\boldsymbol{\mathcal{M}}(S_i) = \sum_{R_i} \mathbb{P}(R_i|S_i)\boldsymbol{\mathcal{M}}(S_{i_1})\boldsymbol{\mathcal{M}}(S_{i_2}) \tag{8}$$

where we use $\boldsymbol{\mathcal{M}}(S_i)$ to denote the intermediate results of eliminating variables in the remainder set R_i. This message update is then carried out recursively according the reverse topological order of the junction tree until we reach the root node. The local summation step for the leaf nodes and root node can be viewed as special cases of (8). For a leaf node C_l, there is no incoming message from children nodes, and hence $\boldsymbol{\mathcal{M}}(S_l) = \mathbb{P}(r_l|S_l)$; for the root node C_r, $S_r = \emptyset$ and $R_i = C_i$, and hence $\mathbb{P}(\mathscr{O}) = \boldsymbol{\mathcal{M}}(\emptyset) = \sum_{\forall X \in C_r} \mathbb{P}(C_r)\boldsymbol{\mathcal{M}}(S_{r_1})\boldsymbol{\mathcal{M}}(S_{r_2})\boldsymbol{\mathcal{M}}(S_{r_3})$.

Example. The message update at the internal node C_{BCDE} in Figure 2 is
$\boldsymbol{\mathcal{M}}(\{C, E\}) = \sum_{B,D} \mathbb{P}(B, D|C, E)\mathbb{P}(f|B, C)\mathbb{P}(g|B, D)$.

4 Tensor Representation for Message Passing

Although the parametrization of latent junction trees using CPTs is very compact and inference (message passing) can be carried out efficiently, parameters in this representation can be difficult to learn. Since the likelihood of the observed data is no longer convex in the latent parameters, local search heuristics, such as

[1] If this is not the case, the derivation is similar but notationally much heavier.

[2] For simplicity of notation, assume $C_i = S_i \cup S_{i_1} \cup S_{i_2}$.

EM, are often employed to learning the parameters. Therefore, our goal is to design a new representation for latent junction trees, such that subsequent learning can be carried out in a local-minimum-free fashion.

In this section, we will develop a new representation for the message update in (8) by embedding each CPT $\mathbb{P}(R_i|S_i)$ into a higher order tensor $\mathcal{P}(C_i)$. As we will see, there will be two advantages to the tensor form. The first is that tensor multiplication can be used to compactly express the sum and product steps involved in message passing. As a very simplistic example, let $\boldsymbol{P}_{A|B} = \mathbb{P}(A|B)$ be a conditional probability matrix and $\boldsymbol{P}_B = \mathbb{P}(B)$ be a marginal probability vector. Then matrix-vector multiplication, $\boldsymbol{P}_{A|B}\boldsymbol{P}_B = P(A)$, sums out variable B. However, if we put the marginal probability of B on the diagonal of a matrix, then B will not be summed out: *e.g.*, if $\boldsymbol{P}_{\oslash_2 B} = \mathbb{P}(\oslash_2 B)$, then $\boldsymbol{P}_{A|B}\boldsymbol{P}_{\oslash_2 B} = \mathbb{P}(A, B)$ (but now B is no longer on the diagonal). We will leverage these facts to derive our tensor representation for message passing.

Moreover, we can then utilize tensor inversion to construct an alternate parameterization. In the very simplistic (matrix) example, note that $\mathbb{P}(A, B) = \boldsymbol{P}_{A|B}\boldsymbol{P}_{\oslash_2 B} = \boldsymbol{P}_{A|B}\boldsymbol{F}\boldsymbol{F}^{-1}\boldsymbol{P}_{\oslash_2 B}$. The invertible transformations $\boldsymbol{F}$ will give us an extra degree of freedom to allow us to design an alternate parameterization of the latent junction tree that is only a function of observed variables. This would not be possible in the traditional representation (Eq. 8).

4.1 Embed CPTs to higher order tensors

As we can see from (6), the joint probability distribution of all variables can be represented by a set of conditional distributions over just subsets of variables. Each one of this conditionals is a low order tensor. For example in Figure 2, the CPT corresponding to the clique node C_{BCDE} would be a 4th order tensor $\mathbb{P}(B, D|C, E)$ where each variable corresponds to a different mode of the tensor. However, this representation is not suitable for deriving the observable representation since message passing cannot be defined easily using the tensor multiplication/sum connection shown above. Instead we will embed these tensors into even higher order tensors to facilitate the computation. The key idea is to introduce duplicate indexes using the mode-specific identity tensors, such that the sum-product steps in message updates can be expressed as tensor multiplications.

More specifically, the number of times a mode of the tensor is duplicated will depend on how many times the corresponding variable in the clique C_i appears in the separator sets incident to C_i. We can define the count for a variable $X_j \in C_i$ as

$$d_{j,i} = \mathbb{I}[X_j \in S_i] + \mathbb{I}[X_j \in S_{i_1}] + \mathbb{I}[X_j \in S_{i_2}], \quad (9)$$

where $\mathbb{I}[\cdot]$ is an indicator function taking value 1 if its argument is true and 0 otherwise. Then the tensor representation of the node C_i is

$$\boldsymbol{\mathcal{P}}(C_i) := \mathbb{P}(\underbrace{\ldots, (\oslash_{d_{j,i}} X_j), \ldots}_{\forall X_j \in R_i} | \underbrace{\ldots, (\oslash_{d_{j',i}} X_{j'}), \ldots}_{\forall X_{j'} \in S_i}), \quad (10)$$

where the labels for the modes of the tensor are the combined labels of the separator sets, *i.e.*, $\{S_i, S_{i_1}, S_{i_2}\}$. The number of times a variable is repeated in the label set is exactly equal to $d_{j,i}$.

Essentially, tensor $\boldsymbol{\mathcal{P}}(C_i)$ contains exactly the same information as the original CPT $\mathbb{P}(R_i|S_i)$. Furthermore, $\boldsymbol{\mathcal{P}}(C_i)$ has a lot of zero entries, and the entries from $\mathbb{P}(R_i|S_i)$ are simply embedded in the higher order tensor $\boldsymbol{\mathcal{P}}(C_i)$. Suppose all variables in node C_i are latent variables each taking k_h values. Then the number of entries needed to specify $\mathbb{P}(R_i|S_i)$ is $k_h^{|C_i|}$, while the high order tensor $\boldsymbol{\mathcal{P}}(C_i)$ has $k_h^{d_i}$ entries where $d_i := \sum_{j:X_j \in C_i} d_{j,i}$ which is never smaller than $k_h^{|C_i|}$. In a sense, the parametrization using higher order tensor $\boldsymbol{\mathcal{P}}(C_i)$ is less compact than the parametrization using the original CPTs. However, constructing the tensor $\boldsymbol{\mathcal{P}}$ this way allows us to express the junction tree message update step in (8) as tensor multiplications (more details in the next section), and then we can leverage tools from tensor analysis to design a local-minimum-free learning algorithm.

The tensor representation for the leaf nodes and the root node are special cases of the representation in (10). The tensor representation at a leaf node C_l is simply equal to its CPT $\boldsymbol{\mathcal{P}}(C_l) = \mathbb{P}(R_l|S_l)$. The root node C_r has no parent, so $\boldsymbol{\mathcal{P}}(C_r) = \mathbb{P}(\ldots, (\oslash_{d_{j,r}} X_j), \ldots)$, $\forall X_j \in C_r$. Furthermore, since $d_{j,i}$ is simply a count of how many times a variable in C_i appears in each of the incident separators, the size of each tensor does not depend on which clique node was selected as the root.

Example. In Figure 2, node C_{BCDE} corresponds to CPT $\mathbb{P}(B, D|C, E)$. Its high order tensor representation is $\boldsymbol{\mathcal{P}}(C_{BCDE}) = \mathbb{P}(\oslash_2 B, D| \oslash_2 C, E)$, since both B and C occur twice in the separator sets incident to C_{BCDE}. Therefore the tensor $\boldsymbol{\mathcal{P}}(\{B, C, D, E\})$ is a 6th order tensor with mode labels $\{B, B, D, C, C, E\}$.

4.2 Tensor message passing

With the higher order tensor representation for clique potentials in the junction tree as in (10), we can express the message update step in (8) as tensor multiplications. Consequently, we can compute the marginal distribution of the observed variables $\mathscr{O}$ in equation (7) recursively using a sequence of tensor multiplications. More specifically the general message update equation

for a node in a junction tree can be expressed as

$$\boldsymbol{\mathcal{M}}(S_i) = \boldsymbol{\mathcal{P}}(C_i) \times_{S_{i_1}} \boldsymbol{\mathcal{M}}(S_{i_1}) \times_{S_{i_2}} \boldsymbol{\mathcal{M}}(S_{i_2}). \quad (11)$$

Here the modes of the tensor $\boldsymbol{\mathcal{P}}(C_i)$ are labeled by the variables, and the mode labels are used to carry out tensor multiplications as explained in Section 2. Essentially, multiplication with respect to the duplicated modes of the tensor $\boldsymbol{\mathcal{P}}(C_i)$ will implement some kind of element-wise multiplication for the incoming messages and then summation over the variables in the remainder set R_i.

The tensor message passing steps in leaf nodes and the root node are special cases of the tensor message update in equation (11). The outgoing message $\boldsymbol{\mathcal{M}}(S_l)$ at a leaf node C_l can be computed by simply setting all variables in R_l to the actual observed values r_l, *i.e.*,

$$\boldsymbol{\mathcal{M}}(S_l) = \mathcal{P}(C_l)_{R_l=r_l} = \mathbb{P}(R_l = r_l | S_l). \quad (12)$$

In this step, there is no difference between the augmented tensor representation and the standard message passing in junction tree. At the root, we arrive at the final results of the message passing algorithm, and we obtain the marginal probability of the observed variables by aggregating all incoming messages from its 3 children, *i.e.*,

$$\mathbb{P}(\mathscr{O}) = \quad (13)$$
$$\boldsymbol{\mathcal{P}}(C_r) \times_{S_{r_1}} \boldsymbol{\mathcal{M}}(S_{r_1}) \times_{S_{r_2}} \boldsymbol{\mathcal{M}}(S_{r_2}) \times_{S_{r_3}} \boldsymbol{\mathcal{M}}(S_{r_3}).$$

Example. For Figure 2, using the following tensors

$$\boldsymbol{\mathcal{P}}(\{B,C,D,E\}) = \mathbb{P}(\oslash_2 B, D \mid \oslash_2 C, E)$$
$$\boldsymbol{\mathcal{M}}(\{B,C\}) = \mathbb{P}(f|B,C)$$
$$\boldsymbol{\mathcal{M}}(\{B,D\}) = \mathbb{P}(g|B,D),$$

we can write the message update for node C_{BCDE} in the form of equation (11) as

$$\boldsymbol{\mathcal{M}}(\{C,E\}) = \boldsymbol{\mathcal{P}}(\{B,C,D,E\}) \times_{\{B,C\}} \boldsymbol{\mathcal{M}}(\{B,C\}) \times_{\{B,D\}} \boldsymbol{\mathcal{M}}(\{B,D\}).$$

Note how the tensor multiplication sums out B and D: $\boldsymbol{\mathcal{P}}(\{B,C,D,E\})$ has two B labels, and it appears in the subscripts of tensor multiplication twice; D appears once in the label and in the subscript of tensor multiplication respectively. Similarly, C is not summed out since there are two C labels but it appears only once in the subscript of tensor multiplication.

5 Transformed Representation

Explicitly learning the tensor representation in (10) is still an intractable problem. Our key observation is that we do not need to recover the tensor representation explicitly if our focus is to perform inference using the message passing algorithm as in (11)–(13). As long as we can recover the tensor representation up to some invertible transformation, we can still obtain the correct marginal probability $\mathbb{P}(\mathscr{O})$.

More specifically, we can insert a mode-specific identity tensor $\boldsymbol{\mathcal{I}}_{\boldsymbol{\sigma}}$ into the message update equation in (11) without changing the outgoing message. Subsequently, we can then replace the mode-specific identity tensor by a pair of tensors, $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{F}}^{-1}$, which are mode-specific inversions of each other ($\boldsymbol{\mathcal{F}} \times_{\boldsymbol{\omega}} \boldsymbol{\mathcal{F}}^{-1} \cong \boldsymbol{\mathcal{I}}_{\boldsymbol{\sigma}}$). Then we can group these inserted tensors with the representation $\boldsymbol{\mathcal{P}}(C)$ from (10), and obtain a transformed version $\widetilde{\boldsymbol{\mathcal{P}}}(C)$ (also see Figure 1). Furthermore, we have the freedom in choosing these collections of tensor inversion pairs. We will show that if we choose them systematically, we will be able to estimate each transformed tensor $\widetilde{\boldsymbol{\mathcal{P}}}(C)$ using the marginal probability of a small set of observed variables (observable representation). In this section, we will first explain the transformed tensor representation.

As an illustration, consider a sequence of matrix multiplications with two identity matrices $\boldsymbol{I}_1 = \boldsymbol{F}_1\boldsymbol{F}_1^{-1}$ and $\boldsymbol{I}_2 = \boldsymbol{F}_2\boldsymbol{F}_2^{-1}$ inserted

$$\begin{aligned}\boldsymbol{ABC} &= \boldsymbol{A}(\boldsymbol{F}_1\boldsymbol{F}_1^{-1})\boldsymbol{B}(\boldsymbol{F}_2\boldsymbol{F}_2^{-1})\boldsymbol{C} \\ &= \underbrace{(\boldsymbol{AF}_1)}_{\widetilde{\boldsymbol{A}}} \underbrace{(\boldsymbol{F}_1^{-1}\boldsymbol{BF}_2)}_{\widetilde{\boldsymbol{B}}} \underbrace{(\boldsymbol{F}_2^{-1}\boldsymbol{C})}_{\widetilde{\boldsymbol{C}}}.\end{aligned}$$

We see that we can equivalently compute $\boldsymbol{ABC}$ using their transformed versions, *i.e.*, $\boldsymbol{ABC} = \widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\widetilde{\boldsymbol{C}}$.

Moving to the tensor case, let us first consider a node C_i and its parent node C_{i_0}. Then the outgoing message of C_{i_0} can be computed recursively as

$$\boldsymbol{\mathcal{M}}(S_{i_0}) = \boldsymbol{\mathcal{P}}(C_{i_0}) \times_{S_i} \underbrace{\boldsymbol{\mathcal{M}}(S_i)}_{\boldsymbol{\mathcal{P}}(C_i)\times_{S_{i_1}}\boldsymbol{\mathcal{M}}(S_{i_1})\times_{S_{i_2}}\boldsymbol{\mathcal{M}}(S_{i_2})} \times \dots$$

Inserting a mode specific identity tensor $\boldsymbol{\mathcal{I}}_{S_i}$ with labels $\{S_i, S_i\}$ and similarly defined mode specific identity tensors $\boldsymbol{\mathcal{I}}_{S_{i_1}}$ and $\boldsymbol{\mathcal{I}}_{S_{i_2}}$ into the above two message updates, we obtain

$$\boldsymbol{\mathcal{M}}(S_{i_0}) = \boldsymbol{\mathcal{P}}(C_{i_0}) \times_{S_i} (\boldsymbol{\mathcal{I}}_{S_i} \times_{S_i} \underbrace{\boldsymbol{\mathcal{M}}(S_i)}_{\boldsymbol{\mathcal{P}}(C_i)\times_{S_{i_1}}(\boldsymbol{\mathcal{I}}_{S_{i_1}}\times_{S_{i_1}}\boldsymbol{\mathcal{M}}(S_{i_1}))\times_{S_{i_2}}(\boldsymbol{\mathcal{I}}_{S_{i_2}}\times_{S_{i_2}}\boldsymbol{\mathcal{M}}(S_{i_1}))}) \times \dots$$

Then we can further expand $\boldsymbol{\mathcal{I}}_{S_i}$ using tensor inversion pairs $\boldsymbol{\mathcal{F}}_i$, $\boldsymbol{\mathcal{F}}_i^{-1}$, *i.e.*, $\boldsymbol{\mathcal{I}}_{S_i} = \boldsymbol{\mathcal{F}}_i \times_{\boldsymbol{\omega}_i} \boldsymbol{\mathcal{F}}_i^{-1}$. Note that both $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{F}}^{-1}$ have two set of mode labels, S_i and another set $\boldsymbol{\omega}_i$ which is related to the observable representation and explained in the next section. Similarly, we expand $\boldsymbol{\mathcal{I}}_{S_{i_1}}$ and $\boldsymbol{\mathcal{I}}_{S_{i_2}}$ using their corresponding tensor inversion pairs.

After expanding tensor identities $\boldsymbol{\mathcal{I}}$, we can regroup terms, and at node C_i we have

$$\begin{aligned}\boldsymbol{\mathcal{M}}(S_i) =& (\boldsymbol{\mathcal{P}}(C_i) \times_{S_{i_1}} \boldsymbol{\mathcal{F}}_{i_1} \times_{S_{i_2}} \boldsymbol{\mathcal{F}}_{i_2}) \\ &\times_{\boldsymbol{\omega}_{i_1}} (\boldsymbol{\mathcal{F}}_{i_1}^{-1} \times_{S_{i_1}} \boldsymbol{\mathcal{M}}(S_{i_1})) \\ &\times_{\boldsymbol{\omega}_{i_2}} (\boldsymbol{\mathcal{F}}_{i_2}^{-1} \times_{S_{i_2}} \boldsymbol{\mathcal{M}}(S_{i_2}))\end{aligned} \quad (14)$$

and at the parent node C_{i_0} of C_i

$$\begin{aligned}\boldsymbol{\mathcal{M}}(S_{i_0}) =& (\boldsymbol{\mathcal{P}}(C_{i_0}) \times_{S_i} \boldsymbol{\mathcal{F}}_i \times \dots) \\ &\times_{\boldsymbol{\omega}_i} (\boldsymbol{\mathcal{F}}_i^{-1} \times_{S_i} \boldsymbol{\mathcal{M}}(S_i)) \times \dots\end{aligned} \quad (15)$$

Now we can define the transformed tensor representation for $\boldsymbol{\mathcal{P}}(C_i)$ as

$$\widetilde{\boldsymbol{\mathcal{P}}}(C_i) := \boldsymbol{\mathcal{P}}(C_i) \times_{S_{i_1}} \boldsymbol{\mathcal{F}}_{i_1} \times_{S_{i_2}} \boldsymbol{\mathcal{F}}_{i_2} \times_{S_i} \boldsymbol{\mathcal{F}}_i^{-1}, \quad (16)$$

where the two transformations $\boldsymbol{\mathcal{F}}_{i_1}$ and $\boldsymbol{\mathcal{F}}_{i_2}$ are ob-

tained from the children side and the transformation $\boldsymbol{\mathcal{F}}_i^{-1}$ is obtained from the parent side. Similarly, we can define the transformed representation for a leaf node and for the root node as

$$\widetilde{\boldsymbol{\mathcal{P}}}(C_l) = \boldsymbol{\mathcal{P}}(C_l) \times_{S_l} \boldsymbol{\mathcal{F}}_l^{-1} \tag{17}$$

$$\widetilde{\boldsymbol{\mathcal{P}}}(C_r) = \boldsymbol{\mathcal{P}}(C_r) \times_{S_{r_1}} \boldsymbol{\mathcal{F}}_{r_1} \times_{S_{r_2}} \boldsymbol{\mathcal{F}}_{r_2} \times_{S_{r_3}} \boldsymbol{\mathcal{F}}_{r_3} \tag{18}$$

Applying these definitions of the transformed representation recursively, we can perform message passing based purely on these transformed representations

$$\widetilde{\boldsymbol{\mathcal{M}}}(S_{i_0}) = \widetilde{\boldsymbol{\mathcal{P}}}(C_{i_0}) \times_{\boldsymbol{\omega}_i} \underbrace{\widetilde{\boldsymbol{\mathcal{M}}}(S_i)}_{\widetilde{\boldsymbol{\mathcal{P}}}(C_i) \times_{\boldsymbol{\omega}_{i_1}} \widetilde{\boldsymbol{\mathcal{M}}}(S_{i_1}) \times_{\boldsymbol{\omega}_{i_2}} \widetilde{\boldsymbol{\mathcal{M}}}(S_{i_2})} \times \dots \tag{19}$$

6 Observable Representation

In the transformed tensor representation in (16)-(18), we have the freedom of choosing the collection of tensor pairs $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{F}}^{-1}$. We will show that if we choose them systematically, we can recover each transformed tensor $\widetilde{\boldsymbol{\mathcal{P}}}(C)$ using the marginal probability of a small set of observed variables (observable representation).

We will focus on the transformed tensor representation in (16) for an internal node C_i (other cases follow as special cases). Due to the recursive way the transformed representation is defined, we only have the freedom of choosing $\boldsymbol{\mathcal{F}}_{i_1}$ and $\boldsymbol{\mathcal{F}}_{i_2}$ in this formula; the choice of $\boldsymbol{\mathcal{F}}_i$ will be fixed by the parent node of C_i. The idea is to choose

- $\boldsymbol{\mathcal{F}}_{i_1} = \mathbb{P}(\mathscr{O}_{i_1}|S_{i_1})$ as the conditional distribution of some set of observed variables $\mathscr{O}_{i_1} \subset \mathscr{O}$ in the subtree rooted at child node C_{i_1} of node C_i, conditioning on the corresponding separator set S_{i_1}.
- Similarly, we choose $\boldsymbol{\mathcal{F}}_{i_2} = \mathbb{P}(\mathscr{O}_{i_2}|S_{i_2})$ where $\mathscr{O}_{i_2} \subset \mathscr{O}$ and it lies in subtree rooted at C_{i_2}.
- Following this convention, $\boldsymbol{\mathcal{F}}_i$ is chosen by the parent node C_{i_0} and is fixed to $\mathbb{P}(\mathscr{O}_i|S_i)$.

Therefore, we have

$$\begin{aligned}\widetilde{\boldsymbol{\mathcal{P}}}(C_i) = \boldsymbol{\mathcal{P}}(C_i) &\times_{S_{i_1}} \mathbb{P}(\mathscr{O}_{i_1}|S_{i_1}) \times_{S_{i_2}} \mathbb{P}(\mathscr{O}_{i_2}|S_{i_2}) \\ &\times_{S_i} \mathbb{P}(\mathscr{O}_i|S_i)^{-1},\end{aligned} \tag{20}$$

where the first two tensor multiplications essentially eliminate the latent variables in S_{i_1} and S_{i_2}.[3] With these choices, we also fix the mode labels $\boldsymbol{\omega}_i$, $\boldsymbol{\omega}_{i_1}$ and $\boldsymbol{\omega}_{i_2}$ in (14) (15) and (19). That is $\boldsymbol{\omega}_i = \mathscr{O}_i$, $\boldsymbol{\omega}_{i_1} = \mathscr{O}_{i_1}$ and $\boldsymbol{\omega}_{i_2} = \mathscr{O}_{i_2}$.

To remove all dependencies on latent variables in $\widetilde{\boldsymbol{\mathcal{P}}}(C_i)$ and relate it to observed variables, we need to eliminate the latent variables in S_i and the tensor $\mathbb{P}(\mathscr{O}_i|S_i)^{-1}$. For this, we multiply the transformed tensor $\widetilde{\boldsymbol{\mathcal{P}}}(C_i)$ by $\mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-})$, where $\mathscr{O}_{i-}$ denotes some set of observed variables which do *not* belong to the subtree rooted at node C_i. Furthermore, $\mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-})$ can be re-expressed using the conditional distribution of $\mathscr{O}_i$ and $\mathscr{O}_{i-}$ respectively, conditioning on the separator set S_i, *i.e.*,

$$\mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-}) = \mathbb{P}(\mathscr{O}_i|S_i) \times_{S_i} \mathbb{P}(\oslash_2 S_i) \times_{S_i} \mathbb{P}(\mathscr{O}_{i-}|S_i).$$

Therefore, we have

$$\mathbb{P}(\mathscr{O}_i|S_i)^{-1} \times_{\mathscr{O}_i} \mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-}) = \mathbb{P}(\oslash_2 S_i) \times_{S_i} \mathbb{P}(\mathscr{O}_{i-}|S_i),$$

and plugging this into (20), we have

$$\begin{aligned}&\widetilde{\boldsymbol{\mathcal{P}}}(C_i) \times_{\mathscr{O}_i} \mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-}) \\ =&\boldsymbol{\mathcal{P}}(C_i) \times_{S_{i_1}} \mathbb{P}(\mathscr{O}_{i_1}|S_{i_1}) \times_{S_{i_2}} \mathbb{P}(\mathscr{O}_{i_2}|S_{i_2}) \\ &\times_{S_i} \mathbb{P}(\oslash_2 S_i) \times_{S_i} \mathbb{P}(\mathscr{O}_{i-}|S_i) \\ =&\mathbb{P}(\mathscr{O}_{i_1}, \mathscr{O}_{i_2}, \mathscr{O}_{i-}),\end{aligned} \tag{21}$$

where $\widetilde{\boldsymbol{\mathcal{P}}}(C_i)$ is now related to only marginal probabilities of observed variables. From the equivalent relation, we can inverting $\mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-})$, and obtain the observable representation for $\widetilde{\boldsymbol{\mathcal{P}}}(C_i)$

$$\widetilde{\boldsymbol{\mathcal{P}}}(C_i) = \mathbb{P}(\mathscr{O}_{i_1}, \mathscr{O}_{i_2}, \mathscr{O}_{i-}) \times_{\mathscr{O}_{i-}} \mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-})^{-1}. \tag{22}$$

Example. For node C_{BCDE} in Figure 2, the choices of $\mathscr{O}_i, \mathscr{O}_{i_1}, \mathscr{O}_{i_2}$ and $\mathscr{O}_{i-}$ are $\{F, G\}$, G, F and H respectively.

There are many valid choices of $\mathscr{O}_{i-}$. In the supplementary, we describe how these different choices can be combined via a linear system using Eq. 21. This can substantially increase performance.

For the leaf nodes and the root node, the derivation for their observable representations can be viewed as special cases of that for the internal nodes. We provide the results for their observable representation below:

$$\widetilde{\boldsymbol{\mathcal{P}}}(C_r) = \mathbb{P}(\mathscr{O}_{r_1}, \mathscr{O}_{r_2}, \mathscr{O}_{r_3}), \tag{23}$$

$$\widetilde{\boldsymbol{\mathcal{P}}}(C_l) = \mathbb{P}(\mathscr{O}_l, \mathscr{O}_{l-}) \times_{\mathscr{O}_{l-}} \mathbb{P}(\mathscr{O}_l, \mathscr{O}_{l-})^{-1}. \tag{24}$$

If $\mathbb{P}(\mathscr{O}_l, \mathscr{O}_{l-})$ is invertible, then $\widetilde{\boldsymbol{\mathcal{P}}}(C_l) = \boldsymbol{\mathcal{I}}_{\mathscr{O}_l}$. Otherwise we need to project $\mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-})$ using a tensor $\boldsymbol{\mathcal{U}}_i$ to make it invertible, as discussed in the next section. The overall algorithm is given in Algorithm 1. Given N *i.i.d.* samples of the observed nodes, we simply replace $\mathbb{P}(\cdot)$ by the empirical estimate $\widehat{\mathbb{P}}(\cdot)$.

Algorithm 1 Spectral algorithm for latent junction tree

In: Junction tree topology and N *i.i.d.* samples $\{x_1^s, \dots, x_{|\mathscr{O}|}^s\}_{s=1}^N$

Out: Estimated marginal $\widehat{\mathbb{P}}(\mathscr{O})$

1: Estimate $\widehat{\boldsymbol{\mathcal{P}}}(C_i)$ for the root, leaf and internal nodes

$$\widehat{\boldsymbol{\mathcal{P}}}(C_r) = \widehat{\mathbb{P}}(\mathscr{O}_{r_1}, \mathscr{O}_{r_2}, \mathscr{O}_{r_3}) \times_{\mathscr{O}_{r_1}} \boldsymbol{\mathcal{U}}_{r_1} \times_{\mathscr{O}_{r_2}} \boldsymbol{\mathcal{U}}_{r_2} \times_{\mathscr{O}_{r_3}} \boldsymbol{\mathcal{U}}_{r_3}$$

$$\widehat{\boldsymbol{\mathcal{P}}}(C_l) = \widehat{\mathbb{P}}(\mathscr{O}_l, \mathscr{O}_{l-}) \times_{\mathscr{O}_{l-}} (\widehat{\mathbb{P}}(\mathscr{O}_l, \mathscr{O}_{l-}) \times_{\mathscr{O}_l} \boldsymbol{\mathcal{U}}_l)^{-1}$$

$$\widehat{\boldsymbol{\mathcal{P}}}(C_i) = \widehat{\mathbb{P}}(\mathscr{O}_{i_1}, \mathscr{O}_{i_2}, \mathscr{O}_{i-}) \times_{\mathscr{O}_{i_1}} \boldsymbol{\mathcal{U}}_{i_1} \times_{\mathscr{O}_{i_2}} \boldsymbol{\mathcal{U}}_{i_2} \times_{\mathscr{O}_{i-}} (\widehat{\mathbb{P}}(\mathscr{O}_i, \mathscr{O}_{i-}) \times_{\mathscr{O}_i} \boldsymbol{\mathcal{U}}_i)^{-1}$$

2: In reverse topological order, leaf and internal nodes send messages

$$\widehat{\boldsymbol{\mathcal{M}}}(S_l) = \widehat{\boldsymbol{\mathcal{P}}}(C_l)_{\mathscr{O}_l = o_l}$$

$$\widehat{\boldsymbol{\mathcal{M}}}(S_i) = \widehat{\boldsymbol{\mathcal{P}}}(C_i) \times_{\mathscr{O}_{i_1}} \widehat{\boldsymbol{\mathcal{M}}}(S_{i_1}) \times_{\mathscr{O}_{i_2}} \widehat{\boldsymbol{\mathcal{M}}}(S_{i_2})$$

3: At the root, obtain $\widehat{\mathbb{P}}(\mathscr{O})$ by

$$\widehat{\boldsymbol{\mathcal{P}}}(C_r) \times_{\mathscr{O}_{r_1}} \widehat{\boldsymbol{\mathcal{M}}}(S_{r_1}) \times_{\mathscr{O}_{r_2}} \widehat{\boldsymbol{\mathcal{M}}}(S_{r_2}) \times_{\mathscr{O}_{r_3}} \widehat{\boldsymbol{\mathcal{M}}}(S_{r_3})$$

[3] If a latent variable in $S_{i_1} \cup S_{i_2}$ is also in S_i, it is not eliminated in this step but in another step.

7 Discussion

The observable representation exists only if there exist tensor inversion pairs $\boldsymbol{\mathcal{F}}_i = \mathbb{P}(\mathscr{O}_i|S_i)$, and $\boldsymbol{\mathcal{F}}_i^{-1}$. This is equivalent to requiring that the rank of the matricized version of $\boldsymbol{\mathcal{F}}_i$ (rows corresponds to modes $\mathscr{O}_i$ and column to modes S_i) has rank $\tau_i := k_h \times |S_i|$. Similarly. the matricized version of $\mathbb{P}(\mathscr{O}_{-i}|S_i)$ also needs to have rank τ_i, so that the matricized version of $\mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-})$ has rank τ_i and is invertible. Thus, it is required that $\#\text{states}(\mathscr{O}_i) \geq \#\text{states}(S_i)$. This can be achieved by either making $\mathscr{O}_i$ consist of a few high dimensional observations, or of many smaller dimensional ones. In the case when $\#\text{states}(\mathscr{O}_i) > \#\text{states}(S_i)$, we need to project $\boldsymbol{\mathcal{F}}_i$ to a lower dimensional space using a tensor $\boldsymbol{\mathcal{U}}_i$ so that it can be inverted. In this case, we define $\boldsymbol{\mathcal{F}}_i := \mathbb{P}(\mathscr{O}_i|S_i) \times_{\mathscr{O}_i} \boldsymbol{\mathcal{U}}_i$. For example, following this through the computation for the leaf gives us that $\widetilde{\boldsymbol{\mathcal{P}}}(C_l) = \mathbb{P}(\mathscr{O}_l, \mathscr{O}_{l-}) \times_{\mathscr{O}_{l-}} (\mathbb{P}(\mathscr{O}_l, \mathscr{O}_{l-}) \times_{\mathscr{O}_l} \boldsymbol{\mathcal{U}}_l)^{-1}$. A good choice of $\boldsymbol{\mathcal{U}}_i$ can be obtained by performing a singular value decomposition of the matricized version of $\mathbb{P}(\mathscr{O}_i, \mathscr{O}_{i-})$ (variables in $\mathscr{O}_i$ are arranged to rows and those in $\mathscr{O}_{i-}$ to columns).

For HMMs and latent trees, this rank condition can be expressed simply as requiring the conditional probability tables of the underlying model to not be rank-deficient. However, junction trees encode more complex latent structures that introduce subtle considerations. A general characterization of the existence condition for observable representation with respect to the graph topology will be our future work. In the appendix, we give some intuition using a couple of examples where observable representations do not exist.

8 Sample Complexity

We analyze the sample complexity of Algorithm 1 and show that it depends on the junction tree topology and the spectral properties of the true model. Let d_i be the order of $\boldsymbol{\mathcal{P}}(C_i)$ and e_i be the number of modes of $\boldsymbol{\mathcal{P}}(C_i)$ that correspond to observed variables.

Theorem 1 *Let $\tau_i = k_h \times |S_i|$, $d_{\max} = \max_i d_i$, and $e_{\max} = \max_i e_i$. Then, for any $\epsilon > 0, 0 < \delta < 1$, if*

$$N \geq O\left(\left(\frac{4k_h^2}{3\beta^2}\right)^{d_{\max}} \frac{k_o^{e_{\max}} \ln \frac{|\mathbb{C}|}{\delta} |\mathbb{C}|^2}{\epsilon^2 \alpha^4}\right)$$

where $\sigma_\tau(*)$ returns the τ^{th} largest singular value and

$$\alpha = \min_i \ \sigma_{\tau_i}(\mathbb{P}(\mathscr{O}_i, \mathscr{O}_{-i})), \quad \beta = \min_i \ \sigma_{\tau_i}(\boldsymbol{\mathcal{F}}_i)$$

Then with probability $1 - \delta$,

$$\sum_{x_1,\ldots,x_{|\mathscr{O}|}} \left|\widehat{\mathbb{P}}(x_1, \ldots, x_{|\mathscr{O}|}) - \mathbb{P}(x_1, \ldots, x_{|\mathscr{O}|})\right| \leq \epsilon \ .$$

See the supplementary for a proof. The result implies that the estimation problem depends exponentially on d_{max} and e_{max}, but note that $e_{max} \leq d_{max}$. Furthermore, d_{max} is always greater than or equal to the treewidth. Note the dependence on the singular values of certain probability tensors. In fully observed models, the accuracy of the learned parameters depends only on how close the empirical estimates of the factors are to the true factors. However, our spectral algorithm also depends on how close the inverses of these empirical estimates are to the true inverses, which depends on the spectral properties of the matrices (Stewart & Sun, 1990).

9 Experiments

We now evaluate our method on synthetic and real data and compare it with both standard EM (Dempster et al., 1977) and stepwise online EM (Liang & Klein, 2009). All methods were implemented in C++, and the matrix library Eigen (Guennebaud et al., 2010) was used for computing SVDs and solving linear systems. For all experiments, standard EM is given 5 random restarts. Online EM tends to be sensitive to the learning rate, so it is given one restart for each of 5 choices of the learning rate $\{0.6, 0.7, 0.8, 0.9, 1\}$ (the one with highest likelihood is selected). Convergence is determined by measuring the change in the log likelihood at iteration t (denoted by $f(t)$) over the average: $\frac{|f(t)-f(t-1)|}{\text{avg}(f(t),f(t-1))} \leq 10^{-4}$ (the same precision as used in Murphy (2005)).

For large sample sizes our method is almost two orders of magnitude faster than both EM and online EM. This is because EM is iterative and every iteration requires inference over all the training examples which can become expensive. On the other hand, the computational cost of our method is dominated by the SVD/linear system. Thus, it is primarily dependent only on the number of observed states and maximum tensor order, and can easily scale to larger sample sizes.

In terms of accuracy, we generally observe 3 distinct regions, low-sample size, mid-sample size, and large sample size. In the low sample size region, EM/online EM tend to overfit to the training data and our spectral algorithm usually performs better. In the mid-sample size region EM/online EM tend to perform better since they benefit from a smaller number of parameters. However, once a certain sample size is reached (the large sample size region), our spectral algorithm consistently outperforms EM/online EM which suffer from local minima and convergence issues.

9.1 Synthetic Evaluation

We first perform a synthetic evaluation. 4 different latent structures are used (see Figure 3): a second order nonhomogenous (NH) HMM, a third order NH HMM, a 2 level NH factorial HMM, and a complicated synthetic junction tree. The second/third order HMMs have $k_h = 2$ and $k_o = 4$, while the factorial HMM and synthetic junction tree have $k_h = 2$, and $k_o = 16$. For each latent structure, we generate 10 sets of model parameters, and then sample N training points and 1000

test points from each set, where N is varied from 100 to 100,000. For evaluation, we measure the accuracy of joint estimation using $error = \frac{|\widehat{\mathbb{P}}(x_1,\ldots,x_O) - \mathbb{P}(x_1,\ldots,x_O)|}{\mathbb{P}(x_1,\ldots,x_O)}$. We also measure the training time of both methods.

Figure 3 shows the results. As discussed earlier, our algorithm is between one and two orders of magnitude faster than both EM and online EM for all the latent structures. EM is actually slower for very small sample sizes than for mid-range sample sizes because of overfitting. Also, in all cases, the spectral algorithm has the lowest error for large sample sizes. Moreover, critical sample size at which spectral overtakes EM/online EM is largely dependent on the number of parameters in the observable representation compared to that in the original parameterization of the model. In higher order/factorial HMM models, this increase is small, while in the synthetic junction tree it is larger.

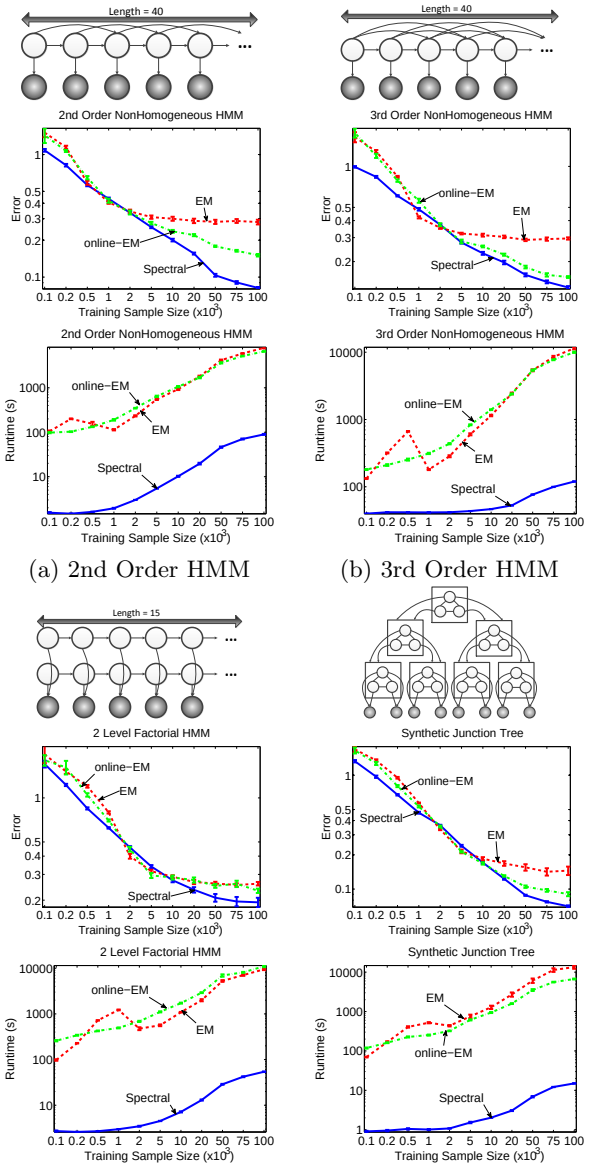
9.2 Splice dataset

We next consider the task of determining splicing sites in DNA sequences (Asuncion & Newman, 2007). Each example consists of a DNA sequence of length 60, where each position in the sequence is either an A, T, C, or G. The goal is to classify whether the sequence is an Intron/Exon site, Exon/Intron site, or neither. During training, for each class a different second order nonhomogeneous HMM with $k_h = 2$ and $k_o = 4$ is trained. At test, the probability of the test sequence is computed for each model, and the one with the highest probability is selected (which we found to perform better than a homogeneous one).

Figure 4, shows our results, which are consistent with our synthetic evaluation. Spectral performs the best in low sample sizes, while EM/online EM perform a little better in the mid-sample size range. The dataset is not large enough to explore the large sample size regime. Moreover, we note that spectral algorithm is much faster for all the sample sizes.

10 Conclusion

We have developed an alternative parameterization that allows fast, local minima free, and consistent parameter learning of latent junction trees. Our approach generalizes spectral algorithms to a much wider range of structures such as higher order, factorial, and semi-hidden Markov models. Unlike traditional non-convex optimization formulations, spectral algorithms allow us to theoretically explore latent variable models in more depth. The spectral algorithm depends not only on the junction tree topology but also on the spectral properties of the parameters. Thus, two models with the same structure may pose different degrees of difficulty based on the underlying singular values. This is very different from learning fully observed junction trees, which is primarily dependent on only the topology/treewidth. Future directions include learning discriminative models and structure learning.



(a) 2nd Order HMM (b) 3rd Order HMM

(c) 2 Level Factorial HMM (d) Synthetic Junction Tree

Figure 3: Comparison of our spectral algorithm (blue) to EM (red) and online EM (green) for various latent structures. Both errors and runtimes in log scale.

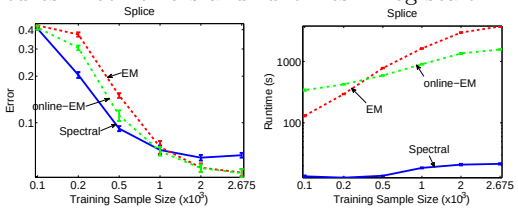


Figure 4: Results on Splice dataset

Acknowledgements: This work is supported by an NSF Graduate Fellowship (Grant No. 0750271) to APP, Georgia Tech Startup Funding to LS, NIH 1R01GM093156, and The Gatsby Charitable Foundation. We thank Byron Boots for valuable discussion.

References

Asuncion, A. and Newman, D.J. UCI machine learning repository, 2007.

Balle, B., Quattoni, A., and Carreras, X. A spectral learning algorithm for finite state transducers. *Machine Learning and Knowledge Discovery in Databases*, pp. 156–171, 2011.

Blei, David and McAuliffe, Jon. Supervised topic models. In *Advances in Neural Information Processing Systems 20*, pp. 121–128. 2007.

Blei, D.M., Ng, A.Y., and Jordan, M.I. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

Cohen, S.B., Stratos, K., Collins, M., Foster, D.P., and Ungar, L. Spectral learning of latent-variable pcfgs. In *Association of Computational Linguistics (ACL)*, volume 50, 2012.

Cowell, R., Dawid, A., Lauritzen, S., and Spiegelhalter, D. *Probabilistic Networks and Expert Sytems*. Springer, New York, 1999.

Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39 (1):1–22, 1977.

Ghahramani, Z. and Jordan, M.I. Factorial hidden Markov models. *Machine learning*, 29(2):245–273, 1997.

Guennebaud, G., Jacob, B., et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

Hsu, D., Kakade, S., and Zhang, T. A spectral algorithm for learning hidden Markov models. In *Proc. Annual Conf. Computational Learning Theory*, 2009.

Kolda, T. and Bader, B. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

Kundu, A., He, Y., and Bahl, P. Recognition of handwritten word: first and second order hidden Markov model based approach. *Pattern recognition*, 22(3): 283–297, 1989.

Lacoste-Julien, S., Sha, F., and Jordan, M.I. Disclda: Discriminative learning for dimensionality reduction and classification. volume 21, pp. 897–904. 2008.

Liang, P. and Klein, D. Online em for unsupervised models. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pp. 611–619. Association for Computational Linguistics, 2009.

Murphy, K. Hidden Markov model (HMM) toolbox for matlab http://www.cs.ubc.ca/murphyk/software/. 2005.

Murphy, K.P. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, 2002.

Parikh, A.P., Song, L., and Xing, E.P. A spectral algorithm for latent tree graphical models. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 1065–1072. ACM, 2011.

Rabiner, L. R. and Juang, B. H. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3 (1):4–16, 1986.

Song, L., Boots, B., Siddiqi, S., Gordon, G., and Smola, A. Hilbert space embeddings of hidden Markov models. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 991–998. ACM, 2010.

Song, L., Parikh, A.P., and Xing, E.P. Kernel embeddings of latent tree graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pp. 2708–2716. 2011.

Stewart, GW and Sun, J. *Matrix Perturbation Theory*. Academic Press, 1990.

Su, X. and Khoshgoftaar, T.M. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4, 2009.

Zhu, J., Ahmed, A., and Xing, E.P. Medlda: maximum margin supervised topic models for regression and classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1257–1264. ACM, 2009.

A Model-Based Approach to Rounding in Spectral Clustering

Leonard K. M. Poon **April H. Liu** **Tengfei Liu** **Nevin L. Zhang**
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong, China
{lkmpoon, aprillh, liutf, lzhang}@cse.ust.hk

Abstract

In spectral clustering, one defines a similarity matrix for a collection of data points, transforms the matrix to get the Laplacian matrix, finds the eigenvectors of the Laplacian matrix, and obtains a partition of the data using the leading eigenvectors. The last step is sometimes referred to as *rounding*, where one needs to decide how many leading eigenvectors to use, to determine the number of clusters, and to partition the data points. In this paper, we propose a novel method for rounding. The method differs from previous methods in three ways. First, we relax the assumption that the number of clusters equals the number of eigenvectors used. Second, when deciding the number of leading eigenvectors to use, we not only rely on information contained in the leading eigenvectors themselves, but also use subsequent eigenvectors. Third, our method is model-based and solves all the three subproblems of rounding using a class of graphical models called *latent tree models*. We evaluate our method on both synthetic and real-world data. The results show that our method works correctly in the ideal case where between-clusters similarity is 0, and degrades gracefully as one moves away from the ideal case.

1 Introduction

Clustering is a data analysis task where one assigns similar data points to the same cluster and dissimilar data points to different clusters. There are many different clustering algorithms, amongst which *spectral clustering* [14] is one approach that has gained prominence in recent years. The idea is to convert clustering into a graph cut problem. More specifically, one first builds a *similarity graph* over the data points using data similarity as edge weights and then partitions the graph by cutting some of the edges. Each connected component in the resulting graph is a cluster. The cut is done so as to simultaneously minimize the cost of cut and balance the sizes of the resulting clusters [12]. The graph cut problem is NP-hard and is hence relaxed. The solution is given by the leading eigenvectors of the so-called Laplacian matrix, which is a simple transformation of the original data similarity matrix. In a post-processing step called rounding [2], a partition of the data points is obtained from those real-valued eigenvectors.

In this paper we focus on rounding. In general, rounding is considered an open problem. There are three subproblems: (1) decide which leading eigenvectors to use; (2) determine the number of clusters; and (3) determine the members in each cluster. Previous rounding methods fall into two groups depending on whether they assume the number of clusters is given. When the number of clusters is known to be k, rounding is usually done based on the k leading eigenvectors. The data points are projected onto the subspace spanned by those eigenvectors and then the K-means algorithm is run on that space to get k clusters [14]. Bach and Jordan [2] approximate the subspace using a space spanned by k piecewise constant vectors and then run K-means on the latter space. Zhang and Jordan [19] observe a link between rounding and the orthogonal Procrustes problem and iteratively use an analytical solution for the latter problem to build a method for rounding. Rebagliati and Verri [10] ask the user to provide a number K that is larger than k and obtain k clusters based on the K leading eigenvectors using a randomized algorithm.

When the number of clusters is not given, one needs to estimate it. A common method is to manually examine the difference between every two consecutive eigenvalues starting from the first two. If a big gap appears for the first time between the k-th and $(k+1)$-

th eigenvalues, then one uses k as an estimate of the number of clusters. Zelnik-Manor and Perona [17] propose an automatic method. For a particular integer k, it tries to rotate the k leading eigenvectors so as to align them with the canonical coordinate system for the eigenspace spanned by those vectors. A cost function is defined in terms of how well the alignment can be achieved. The k with the lowest cost is chosen as an estimate for the number of clusters. Xiang and Gong [16] and Zhao et al. [20] relax the assumption that clustering should be based on a continuous block of eigenvectors at the beginning of the eigenvector spectrum. They use heuristics to choose eigenvectors which do not necessarily form a continuous block, and use Gaussian mixture models to determine the clusters. Socher et al. [13] assume the number of leading eigenvectors to use is given. Based on those leading eigenvectors, they determine the number of clusters and the membership of each cluster using a non-parametric Bayesian clustering method.

In this paper, we propose a model-based approach to rounding. The novelty lies in three aspects. First, we relax the assumption that the number of clusters equals the number of eigenvectors used for rounding. While the assumption is reasonable in the *ideal case* where between-cluster similarity is 0, it is not appropriate in non-ideal cases. Our method allows the number of clusters to differ from the number of eigenvectors. Second, we choose a continuous block of leading eigenvectors for rounding just as [17]. The difference is that, when deciding the appropriateness of the k leading eigenvectors, Zelnik-Manor and Perona [17] use only information contained in those eigenvectors, whereas we also use information in subsequent eigenvectors. So our method uses more information and hence the choice is expected to be more robust. Third, we solve all the three subproblems of rounding within one class of models, namely latent tree models [18]. In contrast, most previous methods either assume that the first two subproblems are solved and focus only on the third subproblem, or do not solve all the subproblems within one class of models [17, 20].

The remaining paper is organized as follows. In Section 2 we review the basics of spectral clustering and point out two key properties of the eigenvectors of the Laplacian matrix in the ideal case. In Section 3 we describe a straightforward method for rounding that takes advantage of the two properties. This method is fragile and breaks down as soon as we move away from the ideal case. In Sections 4 and 5 we propose a model-based method that exploits the same two properties. In Section 6 we evaluate the method on both synthetic and real-world data sets and also on real-world image segmentation tasks. The paper concludes in Section 7.

2 Background

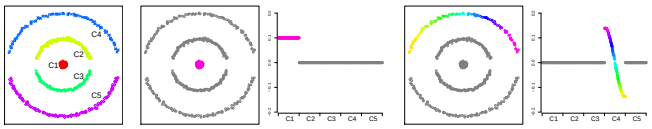
To cluster a set of n data points $X = \{x_1, \ldots, x_n\}$, one starts by defining a non-negative similarity measure s_{ij} for each pair x_i and x_j of data points. In this paper we consider two measures. The *k-NN similarity measure* sets s_{ij} to 1 if x_i is one of the k nearest neighbors of x_j, or vice verse, and 0 otherwise. The *Gaussian similarity measure* sets $s_{ij} = \exp(\frac{-\|x_i - x_j\|^2}{\sigma^2})$, where σ controls the width of neighborhood of each data point. The $n \times n$ matrix $S = [s_{ij}]$ is called the *similarity matrix*. The *similarity graph* is an undirected graph over the data points where the edge is added between x_i and x_j, with weight s_{ij}, if and only if $s_{ij} > 0$.

In spectral clustering, one transforms the similarity matrix to get the *Laplacian matrix*. There are several Laplacian matrices to choose from [14]. We use the *normalized Laplacian matrix* $L_{rw} = I - D^{-1}S$, where I is the identity matrix, and $D = (d_{ij})$ is the diagonal *degree matrix* given by $d_{ii} = \sum_{j=1}^{n} s_{ij}$. Next one computes the eigenvectors of L_{rw} and sorts them in ascending order of their corresponding eigenvalues. Denote those eigenvectors as $e_1, \ldots, e_n$. To obtain k clusters, one forms an $n \times k$ matrix U using the k leading eigenvectors $e_1, \ldots, e_k$ as columns. Each row y_i of U is a point in $\mathbb{R}^k$ and corresponds to an original data point x_i. One partitions the points $\{y_1, \ldots, y_n\}$ into k clusters using the K-means algorithm and obtains k clusters of the original data points accordingly.

Suppose the data consist of k_t true clusters $C_1, \ldots, C_{k_t}$. In the *ideal case*, the similarity graph has k_t connected components, where each of them corresponds to a true cluster. Assume that the data points are ordered based on their cluster memberships. Then the Laplacian matrix L_{rw} has a block diagonal form, with each block corresponding to one true cluster [14]. Moreover, L_{rw} has exactly k_t eigenvalues that equal 0, and the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\vec{1}_{C_1}, \ldots, \vec{1}_{C_{k_t}}$. Create an $n \times k_t$ matrix U_c using those indicator vectors as columns. It is known that, when $k = k_t$, $U = U_c R$ for some orthogonal matrix $R \in \mathbb{R}^{k \times k}$ [8]. For simplicity, assume the non-zero eigenvalues of L_{rw} are distinct. Call those eigenvectors for eigenvalue 0 the *primary eigenvectors* and the others the *secondary eigenvectors*. The facts reviewed in this paragraph imply the following:

Proposition 1. *In the ideal case, eigenvectors of L_{rw} have the following properties:*

1. *The primary eigenvectors are piecewise constant, with each value identifying one true cluster or the union of several true clusters.*
2. *The support (non-zero elements) of each secondary eigenvector is contained by one of the true clusters.*



(a) Data (b) Eigenvector e_1 (c) Eigenvector e_6

Figure 1: Example eigenvectors: There are five true clusters $C_1, \ldots, C_5$ in the data set shown in (a). The 10-NN similarity measure is used to produce the ideal case. Two eigenvectors are shown in (b) and (c). Each eigenvector is depicted in two diagrams. In the first diagram, the values of the eigenvector are indicated by different colors, with grey meaning 0. In the second diagram, the x-axis indexes the data points and the y-axis shows the values of the eigenvector.

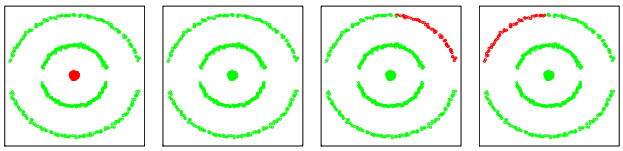
The two properties are illustrated in Fig 1. The primary eigenvector e_1 has two different values, 0.1 and 0. The value 0.1 identifies the cluster C_1, whereas 0 corresponds to the union of the other clusters. The eigenvector e_6 is a secondary eigenvector. Its support is contained by the true cluster C_4.

3 A Naive Method for Rounding

In the original graph cut formulation of spectral clustering [12], an eigenvector has two different values. It partitions the data points into two clusters. In the relaxed problem, an eigenvector can have many different values (see Fig 1(c)). The first step of our method is to obtain, from each eigenvector e_i, two clusters using a confidence parameter δ that is between 0 and 1. Let e_{ij} be the value of e_i at the j-th data point. One of the clusters consists of the data points x_j that satisfy $e_{ij} > 0$ and $e_{ij} > \delta \max_j e_{ij}$, while the other cluster consists of the data points x_j that satisfy $e_{ij} < 0$ and $e_{ij} < \delta \min_j e_{ij}$. The indicator vectors of those two clusters are denoted as e_i^+ and e_i^- respectively. We refer to the process of obtaining e_i^+ and e_i^- from e_i as *binarization*. Applying binarization to the eigenvectors e_1 and e_6 of Fig 1 results in the binary vectors shown in Fig 2. Note that e_1^- is a degenerated binary vector in the sense that it is all 0. We still refer to it as a binary vector for convenience. The following proposition follows readily from Proposition 1.

Proposition 2. *Let e^b be a vector obtained from an eigenvector e of the Laplacian matrix L_{rw} via binarization. In the ideal case:*

1. *If e is a primary eigenvector, then the support of e^b is either a true cluster or the union of several true clusters.*
2. *If e is a secondary eigenvector, then the support of e^b is contained by one of the true clusters.*



(a) e_1^+ (b) e_1^- (c) e_6^+ (d) e_6^-

Figure 2: Binary vectors obtained from eigenvectors e_1 and e_6 through binarization with $\delta = 0.1$. Data points have values 1 (red) or 0 (green).

Each binary vector gives a partition of the data points, with one cluster comprising points with value 1 and another comprising points with value 0. Consider two partitions where one divides the data into two clusters C_1 and C_2 and the other into P_1 and P_2. *Overlaying* the two partitions results in a new partition consisting of clusters $C_1 \cap P_1$, $C_1 \cap P_2$, $C_2 \cap P_1$, and $C_2 \cap P_2$. Note that there are not necessarily exactly 4 clusters in the new partition as some of the 4 intersections might be empty. It is evident how to overlay multiple partitions.

Consider the case when the number q of leading eigenvectors to use is given. Here is a straightforward method for rounding:

NAIVE-ROUNDING1(q)

1. Compute the q leading eigenvectors of L_{rw}.
2. Binarize the q eigenvectors.
3. Obtain a partition of the data using each binary vector from the previous step.
4. Overlay all the partitions to get the final partition.

Note that the number of clusters obtained by NAIVE-ROUNDING1 is determined by q, but it is not necessarily the same as q. The following proposition is an easy corollary of Proposition 2.

Proposition 3. *In the ideal case and when q is smaller than or equal to the number of true clusters, the clusters obtained by* NAIVE-ROUNDING1 *are either true clusters or unions of true clusters.*

Now consider how to determine the number q of leading eigenvectors to use. We do so by making use of Proposition 2. The idea is to gradually increase q in NAIVE-ROUNDING1 and test the partition P_q obtained for each q to see whether it satisfies the condition of Proposition 2 (2).

Suppose K is a sufficiently large integer. Denote a subroutine that tests the partition P_q by `cTest`(P_q, q, K). To perform this test, we use the binary vectors obtained from the eigenvectors from the range $[q+1, K]$. If the support of every such binary vector is contained by some cluster in P_q, we say that P_q satisfies the *containment condition* and `cTest` returns `true`. Oth-

erwise, P_q violates the condition and `cTest` returns `false`.

Let k_t be the number of true clusters. When $q = k_t$, $\texttt{cTest}(P_q, q, K)$ passes because of Proposition 2. When $q = k_t + 1$, P_q is likely to be finer than the true partition. Consequently, $\texttt{cTest}(P_q, q, K)$ may fail. The probability of this happening increases with the number of binary vectors used in the test. To make the probability high, we pick K such that $K/2$ is safely larger than k_t and let q run from 2 to $\lfloor K/2 \rfloor$. When $q < k_t$, $\texttt{cTest}(P_q, q, K)$ usually passes.

The above discussions suggest that if $\texttt{cTest}(P_q, q, K)$, for the first time, passes for some $q = k$ and fails for $q = k + 1$, we can use k as an estimate of k_t. Consequently, we can use the k leading eigenvectors for clustering and return P_k as the final clustering result. This leads to the following algorithm:

NAIVE-ROUNDING2(K)

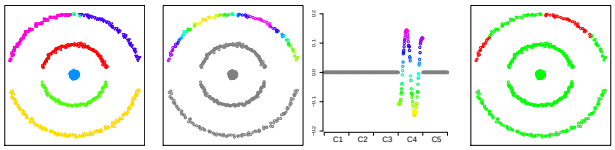
1. For $q = 2$ to $\lfloor K/2 \rfloor$,
   (a) $P \leftarrow$ NAIVE-ROUNDING1(q).
   (b) $P' \leftarrow$ NAIVE-ROUNDING1($q + 1$).
   (c) If $\texttt{cTest}(P, q, K) = \texttt{true}$ and $\texttt{cTest}(P', q + 1, K) = \texttt{false}$, return P.
2. Return P.

Suppose an integer K is given such that $K/2$ is safely larger than the number of true clusters. The algorithm NAIVE-ROUNDING2 automatically decides how many leading eigenvectors to use and automatically determines the number of clusters. Consider running it on the data set shown in Fig 1(a) with $K = 40$. It loops q through 2 to 4 without terminating because both the two tests at Step 1(c) return `true`. When $q = 5$, P is the true partition and P' is shown in Fig 3(a). At Step 1(c), the first test for P passes. However, the second test for P' fails, because the support of the binary vector e_{16}^+ is not contained by any cluster in P' (Fig 3). So, NAIVE-ROUNDING2 terminates at Step 1(c) and returns P (the true partition) as the final result.

4 Latent Class Models for Rounding

NAIVE-ROUNDING2 is fragile. It can break down as soon as we move away from the ideal case. In this and the next section, we describe a model-based method that also exploits Proposition 2 and is more robust.

In the model-based method, the binary vectors are regarded as features and the problem is to cluster the data points based on those features. So what we face is a *problem of clustering discrete data*. As in the previous section, there is a question of how many leading eigenvectors to use. In this section, we assume the number q of leading eigenvectors to use is given. In the next section, we will discuss how to determine q.



(a) 7 clusters (b) e_{16} (c) e_{16}^+

Figure 3: Illustration of NAIVE-ROUNDING2: (a) shows the 7 clusters given by the first 6 pairs of binary vectors. (b) and (c) show eigenvector e_{16} and one of the binary vectors obtained from e_{16}. The support of e_{16}^+ (the red region) is not contained by any of the clusters in (a).

The problem we address in this section is how to cluster the data points based on the first q pairs of binary vectors e_1^+, e_1^-, ..., e_q^+, e_q^-. We solve the problem using latent class models (LCMs) [3]. LCMs are commonly used to cluster discrete data, just as Gaussian mixture models are used to cluster continuous data. Technically they are the same as the Naive Bayes model except that the class variable is not observed.

The LCM for our problem is shown in Fig 4(a). So far we have been using the notations e_s^+ and e_s^- to denote vectors of n elements or functions over the data points. In this and the next sections, we overload the notations to denote random variables that take different values at different data points. The latent variable Y represents the partition to find and each state of Y represents a cluster. So the number of states of Y, often called the *cardinality* of Y, is the number of clusters. To learn an LCM from data means to determine: (1) the cardinality of Y, i.e., the number of clusters; and (2) the probability distributions $P(Y)$, $P(e_s^+|Y)$, and $P(e_s^-|Y)$, i.e., the characterization of the clusters.

After an LCM is learned, one can compute the posterior distribution of Y for each data point. This gives a *soft partition* of the data. To get a *hard partition*, one can assign each data point to the state of Y that has the maximum posterior probability. This is called *hard assignment*.

There are two cases with the LCM learning problem, depending on whether the number of clusters is known. When that number is known, we only need to determine the probability distributions $P(Y)$, $P(e_s^+|Y)$, and $P(e_s^-|Y)$. This is done using the EM algorithm [5].

Proposition 4. *It is possible to set the probability parameter values of the LCM in Fig 4(a) in such a way that it gives the same partition as* NAIVE-ROUDING1. *Moreover, those parameter values maximize the likelihood of the model.*

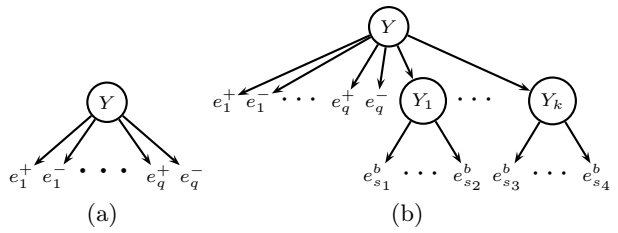


Figure 4: (a) Latent class model and (b) latent tree model for rounding: The binary vectors e_1^+, e_1^-, ..., e_q^+, e_q^- are regarded as random variables. The discrete latent variable Y represents the partition to find.

The proof is omitted due to space limit. It is well known that the EM algorithm aims at finding the maximum likelihood estimate (MLE) of the parameters. So the LCM method for rounding actually tries to find the same partition as NAIVE-ROUNDING1.

Now consider the case when the number of clusters is not known. We determine it using the BIC score [11]. Given a model m, the score is defined as $BIC(m) = \log P(D|m,\theta) - \frac{d}{2}\log n$, where D is the data, θ is the MLE of the parameters, and d is the number of free parameters in the model. We start by setting the number k of clusters to 2 and increase it gradually. We stop the process as soon as the BIC score of the model starts to decrease, and use the k with the maximum BIC score as the estimate of the number of clusters. Note that the number of clusters determined using the BIC score might not be the same as the number of clusters found by NAIVE-ROUNDING1.

5 Latent Tree Models for Rounding

In this section we present a method for determining the number q of leading eigenvectors to use. Consider an integer q between 2 and $K/2$. We first build an LCM using the first q pairs of binary vectors and obtain a hard partition of the data using the LCM. Suppose k clusters C_1, ..., C_k are obtained. Each cluster C_r corresponds to a state r of the latent variable Y.

We extend the LCM model to obtain the model shown in Fig 4(b). We do this in three steps. First, we introduce k new latent variables Y_1, ..., Y_k into the model and connect them to Y. Each Y_r is a binary variable and its conditional distribution is set as follows:

$$P(Y_r = 1|Y = r') = \begin{cases} 1 & \text{if } r' = r, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

So the state $Y_r = 1$ means the cluster C_r and $Y_r = 0$ means the union of all the other clusters.

Next, we add binary vectors from the range $[q+1, K]$ to the model by connecting them to the new latent variables. For convenience we call those vectors *secondary binary vectors*. This is not to be confused with the secondary eigenvectors mentioned in Proposition 1. For each secondary binary vector e_s^b, let D_s be its support. When determining to which Y_r to connect e_s^b, we consider how well the cluster C_r covers D_s. We connect e_s^b to the Y_r such that C_r covers D_s the best, in the sense that the quantity $|D_s \cap C_r|$ is maximized, where $|.|$ stands for the number of data points in a set. We break ties arbitrarily.

Finally, we set the conditional distribution $P(e_s^b|Y_r)$ as follows:

$$P(e_s^b = 1|Y_r = 1) = \frac{|D_s \cap C_r|}{|C_r|} \quad (2)$$

$$P(e_s^b = 1|Y_r = 0) = \frac{|D_s - C_r|}{n - |C_r|} \quad (3)$$

What we get after the extension is a tree structured probabilistic graphical model, in which the variables at the leaf nodes are observed and the variables at the internal nodes are latent. Such models are known as *latent tree models (LTMs)* [4, 15], sometimes also called hierarchical latent class models [18]. The LCM part of the model is called its *primary part*, while the newly added part is called the *secondary part*. The parameter values for the primary part is determined during LCM learning, while those for the secondary part are set manually by Equations (1–3).

To determine the number q of leading eigenvectors to use, we examine all integers in the range $[2, K/2]$. For each such integer q, we build an LTM as described above and compute its BIC score. We pick the q with the maximum BIC score as the answer. After q is determined, we use the primary part of the LTM to partition the data. In other words, the secondary part is used only to determine q. We call this method *LTM-Rounding*.

Here are the intuitions behind the LTM method. If the support D_s of e_s^b is contained in cluster C_r, it fits the situation to connect e_s^b to Y_r. The model construction no longer fits the data well if D_s is not contained in any of the clusters. The worst case is when two different clusters C_r and $C_{r'}$ cover D_s equally well and better than other clusters. In this case, e_s^b can be either connected to Y_r or to $Y_{r'}$. Different choices here lead to different models. As such, neither choice is 'ideal'. Even when there is only one cluster C_r that covers D_s the best, connecting e_s^b to Y_r is still intuitively not 'perfect' as long as C_r does not cover D_s completely.

So when the support of every secondary binary vector is contained by one of the clusters C_1, ..., C_k, the

LTM we build would fit the data well. However, when the supports of some secondary binary vectors are not completely covered by any of the clusters, the LTM would not fit the data well. In the ideal case, the fit would be good if q is the number k_t of eigenvectors for eigenvalue 0 according to Proposition 2. The fit would not be as good otherwise. This is why the likelihood of the LTM contains information that can be used to choose q.

We now summarize the LTM method for rounding:

LTM-ROUNDING(K,δ)

1. Compute the K leading eigenvectors of L_{rw}.
2. Binarize the eigenvectors with confidence parameter δ as in Section 3.
3. $S^* \leftarrow -\infty$.
4. For $q = 2$ to $\lfloor K/2 \rfloor$,
   (a) $m_{lcm} \leftarrow$ the LCM learnt using the first q pairs of binary vectors as shown in Section 4.
   (b) $P \leftarrow$ hard partition obtained using m_{lcm}.
   (c) $m_{ltm} \leftarrow$ the LTM extended from m_{lcm} as explained in Section 5.
   (d) $S \leftarrow$ the BIC score of m_{ltm}.
   (e) If $S > S^*$, then $P^* \leftarrow P$ and $S^* \leftarrow S$.
5. Return P^*.

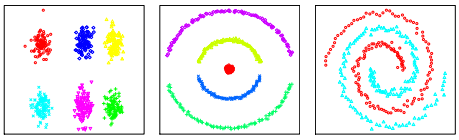
An implementation of LTM-ROUNDING can be obtained from `http://www.cse.ust.hk/~lzhang/ltm/index.htm`.

The choice of K should be such that $K/2$ is safely larger than the number of true clusters. We do not allow q to be larger than $K/2$, so that there is sufficient information in the secondary part of the LTM to determine the appropriateness of using the q leading eigenvectors for rounding. The parameter δ should be picked from the range $(0, 1)$. We recommend to set δ at 0.1. The sensitivity of LTM-ROUNDING with respect to those parameters will be investigated in Section 6.3.

6 Empirical Evaluations

Our empirical investigations are designed to: (1) show that LTM-ROUNDING works perfectly in the ideal case and its performance degrades gracefully as we move away from the ideal case, and (2) compare LTM-ROUNDING with alternative methods. Synthetic data are used for the first purpose, while both synthetic and real-world data are used for the second purpose.

LTM-ROUNDING has two parameters δ and K. We set $\delta = 0.1$ and $K = 40$ in all our experiments except in sensitivity analysis (Section 6.3). For each set of synthetic data, 10 repetitions were run.



(a) 6 clusters (b) 5 clusters (c) 2 clusters

Figure 5: Synthetic data set for the ideal case. Each color and shape of the data points identifies a cluster. Clusters were recovered by LTM-ROUNDING correctly.

6.1 Performance in the Ideal Case

Three data sets were used for the ideal case (Fig 5). They vary in the number and the shape of clusters. Intuitively the first data set is the easiest, while the third one is the hardest. To produce the ideal case, we used the 10-NN similarity measure for the first two data sets. For the third one, the same measure gave a similarity graph with one single connected component. So we used the 3-NN similarity measure instead.

LTM-ROUNDING produced the same results on all 10 runs. The results are shown in Fig 5 using colors and shapes of data points. Each color identifies a cluster. LTM-ROUNDING correctly determined the numbers of clusters and recovered the true clusters perfectly.

6.2 Graceful Degrading of Performance

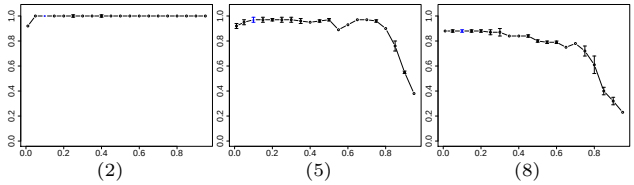
To demonstrate that the performance of LTM-ROUNDING degrades gracefully as we move away from the ideal case, we generated 8 new data sets by adding different levels of noise to the second data set in Fig 5. The Gaussian similarity measure was adopted with $\sigma = 0.2$. So the similarity graphs for all the data sets are complete.

We evaluated the quality of an obtained partition by comparing it with the true partition using Rand index (RI) [9] and variation of information (VI) [7]. Note that higher RI values indicate better performance, while the opposite is true for VI. The performance statistics of LTM-ROUNDING are shown in Table 1. We see that RI is 1 for the first three data sets. This means that the true clusters have been perfectly recovered. The index starts to drop from the 4th data set onwards in general. It falls gracefully with the increase in the level of noise in data. Similar trend can also be observed on VI.

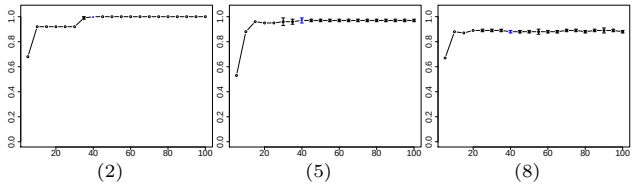
The partitions produced by LTM-ROUNDING at the best run (in terms of BIC score) are shown in Fig 7(a) using colors and shapes of the data points. We see that on the first four data sets, LTM-ROUNDING correctly determined the number of clusters and the members of each cluster. On the next three data sets, it also

Table 1: Performances of various methods on the 8 synthetic data sets in terms of Rand index (RI) and variation of information (VI). Higher values of RI or lower values of VI indicate better performance. K-MEANS and GMM require extra information for rounding and should not be compared with the other two methods directly.

	Data	1	2	3	4	5	6	7	8
RI	LTM	**1.0±.00**	**1.0±.00**	**1.0±.00**	**.99±.01**	**.97±.02**	**.98±.01**	**.94±.01**	.88±.01
	ROT	.92±.00	.92±.00	**1.0±.00**	.98±.00	.52±.00	.52±.00	.88±.00	**.90±.00**
	K-MEANS	1.0±.00	1.0±.00	1.0±.00	1.0±.00	.85±.00	.72±.00	.71±.00	.75±.00
	GMM	1.0±.00	1.0±.00	1.0±.00	1.0±.00	.94±.00	.88±.00	.91±.00	.88±.00
VI	LTM	**.00±.00**	**.00±.00**	**.00±.00**	**.06±.09**	**.29±.19**	**.28±.14**	**.79±.12**	1.64±.10
	ROT	.40±.00	.40±.00	**.00±.00**	.20±.00	1.60±.00	1.60±.00	1.85±.00	**1.42±.00**
	K-MEANS	.00±.00	.00±.00	.00±.00	.00±.00	1.04±.00	1.41±.00	1.52±.00	1.97±.00
	GMM	.00±.00	.00±.00	.00±.00	.00±.00	.85±.00	.91±.00	1.25±.00	1.74±.00

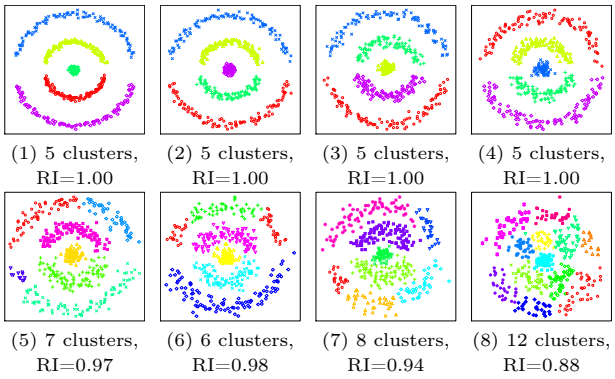


(a) $K = 40$ with varying δ (x-axis)

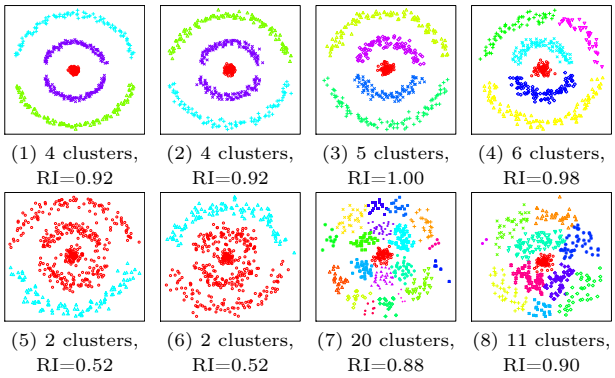


(b) $\delta = 0.1$ with varying K (x-axis)

Figure 6: Sensitivity analysis on parameters δ and K in LTM-ROUNDING. The y-axis denotes Rand index. We recommend $\delta = 0.1$ and $K = 40$ in general.



(a) Partitions produced by LTM-ROUNDING



(b) Partitions produced by ROT-ROUNDING

Figure 7: The 8 synthetic data sets.

correctly recovered the true clusters except that the top and the bottom crescent clusters might have been broken into two or three parts. Given the gaps between the different parts, the results are probably the best one can hope for. The result on the last data set is less ideal but still reasonable.

Putting together, the above discussions indicate that the performance of LTM-ROUNDING degrades gracefully as we move away from the ideal case.

6.3 Sensitivity Analysis

We now study the sensitivity of the parameters δ and K in LTM-ROUNDING. Experiments were conducted on the 2nd, 5th, and 8th data sets in Fig 7. Those data sets contain different levels of noise and hence are at different distances from the ideal case.

To determine the sensitivity of LTM-ROUNDING with respect to δ, we fix $K = 40$ and let δ vary between 0.01 and 0.95. The RI statistics are shown in Fig 6(a). We see that on data set (2) the performance of LTM-ROUNDING is insensitive to δ except when $\delta = 0.01$. On data set (5), the performance of LTM-ROUNDING is more or less robust when $0.1 \leq \delta \leq 0.3$. Its performance gets particularly worse when $\delta \geq 0.8$. Similar behavior can be observed on data set (8). Those results suggest that the performance of LTM-ROUNDING is robust with respect to δ in situations close to the ideal case and it becomes sensitive in situations that are far away from the ideal case. In general, we recommend $\delta = 0.1$.

To determine the sensitivity of LTM-ROUNDING with respect to K, we fix $\delta = 0.1$ and let K vary between 5 and 100. The RI statistics are shown in Fig 6(b). It is clear that the performance of LTM-ROUNDING is

robust with respect to K as long as it is not too small.

6.4 Running Time

LTM-ROUNDING deals with tree-structured models. It is deterministic everywhere except at Step 4(a), where the EM algorithm is called to estimate the parameters of LCM. EM is an iterative algorithm and is computationally expensive in general. However, it is efficient on LCMs. So the running time of LTM-ROUNDING is relatively short. To process one of the data sets in Fig 7, it took around 10 minutes on a laptop computer.

6.5 Alternative Methods for Rounding

To compare with LTM-ROUNDING, we included the method by Zelnik-Manor and Perona [17] in our experiments. The latter method determines the appropriateness of using the q leading eigenvectors by checking how well they can be aligned with the canonical coordinates through rotation. So we name it ROT-ROUNDING. Both LTM-ROUNDING and ROT-ROUNDING can determine the number q of eigenvectors and the number k of clusters automatically. Therefore, they are directly comparable. The method by Xiang and Gong [16] is also related to our work. However, its implementation is not available to us and hence it is excluded from comparison.

K-MEANS and GMM can also be used for rounding. However, both methods cannot determine q, and K-MEANS cannot even determine k. We therefore gave the number k_t of true clusters for K-MEANS and used $q = k_t$ for both methods. Since the two methods required additional information, they are included in our experiments only for reference. They should not be compared with LTM-ROUNDING directly.

ROT-ROUNDING and GMM require the maximum allowable number of clusters. We set that number to 20.

6.6 Comparison on Synthetic Data

Table 1 shows the performance statistics of the three other methods on the 8 synthetic data sets. LTM-ROUNDING performed better than ROT-ROUNDING on all but two data sets. The differences were substantial on 5 of those data sets.

The partitions produced by ROT-ROUNDING at one run are shown in Fig 7(b).[1] We see that on the first two data sets, ROT-ROUNDING underestimated the number of clusters and merged the two smaller crescent clusters. This is serious under-performance given the easiness of those data sets. On the 3rd data set, it recovered the true clusters correctly. On the 4th data set, it broke the top crescent cluster into two. On the 5th data set, it recovered the bottom cluster correctly but merged all the other clusters incorrectly. This leads to a much lower RI. The story on the 6th data set is similar. On the 7th data set, it produced many more clusters than the number of true clusters. Therefore, its RI is also lower. On the last data set, the clustering obtained by ROT-ROUNDING is not visually better than the one produced by LTM-ROUNDING, even though RI suggests otherwise. Given all the evidence presented, we conclude that the performance of LTM-ROUNDING is significantly better than that of ROT-ROUNDING.

[1]Same results were obtained for all 10 repetitions.

Table 2: Comparison on MNIST digits data.

Method	#clusters	RI	VI
LTM	**9.3±.82**	.91±.00	**2.17±.07**
ROT	19.0±.00	**.92±.00**	2.40±.00
K-MEANS	(10)	.90±.00	1.83±.00
GMM	16.0±.00	.91±.00	2.14±.00
SD-CRP	9.3±.96	.89±.00	2.72±.08

Like LTM-ROUNDING, both K-MEANS and GMM recovered clusterings correctly on the first three data sets (Table 1). They performed slightly better than LTM-ROUNDING on the 4th data set. However, their performances were considerably worse on the next three data sets. They were also worse on the last one in terms of VI. It happened even though K-MEANS and GMM were given additional information for rounding. This shows the superior performance of LTM-ROUNDING.

6.7 MNIST Digits Data

In the next two experiments, we used real-world data to compare the rounding methods. The MNIST digits data were used in this subsection. The data consist of 1000 samples of handwritten digits from 0 to 9. They were preprocessed by the deep belief network as described in [6] using their accompanying code.[2] Table 2 shows the results averaged over 10 runs.

We see that the number of clusters estimated by LTM-ROUNDING is close to the ground truth (10), but that by ROT-ROUNDING is considerably larger than 10. In terms of quality of clusterings, the results are inconclusive. RI suggests that ROT-ROUNDING performed slightly better, but VI suggests that LTM-ROUNDING performed significantly better.

Compared with LTM-ROUNDING, K-MEANS obtained

[2]We thank Richard Socher for sharing the preprocessed data with us. The original data can be found at `http://yann.lecun.com/exdb/mnist/`.

a better clustering (in terms of VI), whereas GMM obtained one with similar quality. However, K-MEANS was given the number of true clusters and GMM the number of eigenvectors. GMM also overestimated the number of clusters even with the extra information.

A non-parametric Bayesian clustering method, called SD-CRP, has recently been proposed for rounding [13]. Although we obtained the same data from their authors, we could not get a working implementation for their method. Therefore, we simply copied their reported performance to Table 2. Note that SD-CPR can determine the number clusters automatically. However, it requires the number of eigenvectors as input, and the number was set to 10 in this experiment. Hence, SD-CPR requires more information than LTM-ROUNDING. Table 2 shows that it estimated a similar number of clusters as LTM-ROUNDING. However, its clustering was significantly worse in terms of RI or VI. This shows that LTM-ROUNDING performed better than SD-CPR even with less given information.

6.8 Image Segmentation

The last part of our empirical evaluation was conducted on real-world image segmentation tasks. Five images from the Berkeley Segmentation Data Set (BSDS500) were used. They are shown in the first column of Fig 8. The similarity matrices were built using the method proposed by Arbeláez et al. [1].

The segmentation results obtained by ROT-ROUNDING and LTM-ROUNDING are shown in the second and third columns of Fig 8 respectively. On the first two images, ROT-ROUNDING did not identify any meaningful segments. In contrast, LTM-ROUNDING identified the polar bear and detected the boundaries of the river on the first image. It identified the bottle, the glass and a lobster on the second image.

An obvious undesirable aspect of the results is that some uniform regions are broken up. Examples include the river bank and the river itself in the first image, and the background and the table in the second image. This is a known problem of spectral clustering when applied to image segmentation and can be dealt with using image analysis techniques [1]. We do not deal with the problem in this paper.

On the third image the performance of LTM-ROUNDING was better because the lizard was identified and the segmentation lines follow leaf edges more closely. On the fourth image LTM-ROUNDING did a better job at detecting the edges around the lady's hands and skirt and on the left end of the silk scarf. However, ROT-ROUNDING did a better job at the last image because it produced a cleaner segmentation.

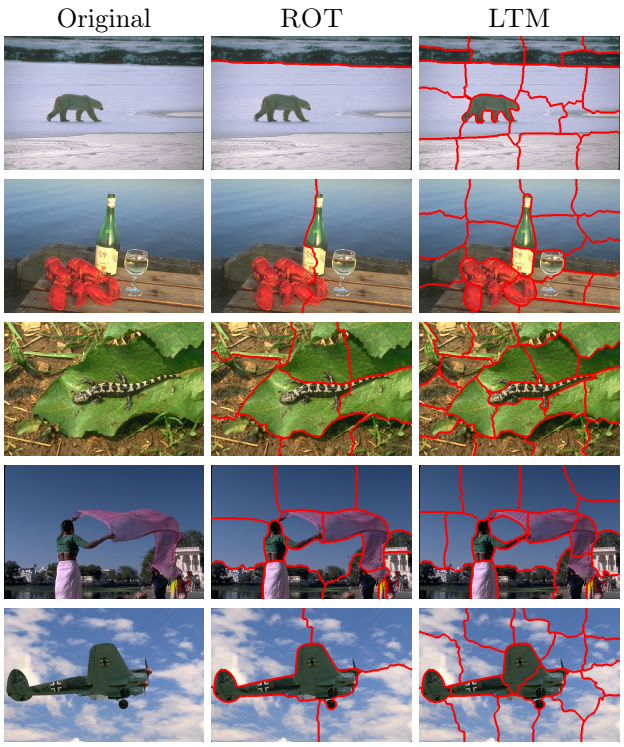


Figure 8: Image segmentation results.

Overall, the performance of LTM-ROUNDING was better than that of ROT-ROUNDING. The objective of this section has been to compare LTM-ROUNDING and ROT-ROUNDING on the same collection of eigenvectors. The conclusion is meaningful even if the final segmentation results are not as good as the best that can be achieved by image analysis techniques.

7 Conclusion

Rounding is an important step of spectral clustering that has not received sufficient attention. Not many papers have been published on the topic, especially on the issues of determining the number of leading eigenvectors to use. In this paper, we have proposed a novel method for the task. The method is based on latent tree models. It can automatically select an appropriate number of eigenvectors to use, determine the number of clusters, and finally assign data points to clusters. We have shown that the method works correctly in the ideal case and its performance degrades gracefully as we move away from the ideal case.

Acknowledgements

Research on this paper was supported by China National Basic Research 973 Program project No. 2011CB505101 and Guangzhou HKUST Fok Ying Tung Research Institute.

References

[1] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 33 (5):898–916, 2011.

[2] Francis R. Bach and Michael I. Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7:1963–2001, 2006.

[3] David J. Bartholomew and Martin Knott. *Latent Variable Models and Factor Analysis*. Arnold, 2nd edition, 1999.

[4] Tao Chen, Nevin L. Zhang, Tengfei Liu, Kin Man Poon, and Yi Wang. Model-based multidimensional clustering of categorical data. *Artificial Intelligence*, 176:2246–2269, 2012.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1): 1–38, 1977.

[6] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[7] Marina Meilă. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98:873–895, 2007.

[8] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clusterings: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2002.

[9] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[10] Nicola Rebagliati and Alessandro Verri. Spectral clustering with more than k eigenvectors. *Neurocomputing*, 74:1391–1401, 2011.

[11] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[12] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 731–737, 1997.

[13] Richard Socher, Andrew Maas, and Christopher D. Manning. Spectral Chinese restaurant processes: Nonparametric clustering based on similarities. In *14th International Conference on Artificial Intelligence and Statistics*, 2011.

[14] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

[15] Yi Wang, Nevin L. Zhang, and Tao Chen. Latent tree models and approximate inference in Bayesian networks. *Journal of Artificial Intelligence Research*, 32:879–900, 2008.

[16] Tao Xiang and Shaogang Gong. Spectral clustering with eigenvector selection. *Pattern Recognition*, 41(3):1012–1029, 2008.

[17] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems*, 2005.

[18] Nevin L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.

[19] Zhihua Zhang and Michael I. Jordan. Multiway spectral clustering: A margin-based perspective. *Statistical Science*, 23(3):383–403, 2008.

[20] Feng Zhao, Licheng Jiao, Hanqiang Liu, Xinbo Gao, and Maoguo Gong. Spectral clustering with eigenvector selection based on entropy ranking. *Neurocomputing*, 73:1704–1717, 2010.

A Maximum Likelihood Approach For Selecting Sets of Alternatives

Ariel D. Procaccia
Computer Science Department
Carnegie Mellon University
arielpro@cs.cmu.edu

Sashank J. Reddi
Machine Learning Department
Carnegie Mellon University
sjakkamr@cs.cmu.edu

Nisarg Shah
Computer Science Department
Carnegie Mellon University
nkshah@cs.cmu.edu

Abstract

We consider the problem of selecting a subset of alternatives given noisy evaluations of the relative strength of different alternatives. We wish to select a k-subset (for a given k) that provides a maximum likelihood estimate for one of several objectives, e.g., containing the strongest alternative. Although this problem is $\mathcal{NP}$-hard, we show that when the noise level is sufficiently high, intuitive methods provide the optimal solution. We thus generalize classical results about singling out one alternative and identifying the hidden ranking of alternatives by strength. Extensive experiments show that our methods perform well in practical settings.

1 Introduction

The initial motivation for this paper stemmed from discussions with the inventors of EteRNA (http://eterna.cmu.edu). Similarly to Foldit [9], EteRNA is a scientific discovery game where the goal is to design RNA molecules that fold into stable structures. Thousands of human players propose RNA designs weekly, but only a relatively small number k of them can be synthesized in a laboratory to discover whether they actually fold well. To choose k designs to synthesize, players vote over the proposed designs using a voting rule known as *k-approval*: each player reports up to k favorite designs, each of which is awarded one point. The k designs that receive the most points are synthesized, and the best design is identified.

The aggregation of opinions via k-approval has several shortcomings. For example, players typically do not consider all proposed designs, and in particular designs that receive votes early in the voting process gain visibility and therefore usually accumulate more votes, leading to a snowball effect. One alternative is to elicit players' opinions about pairs of designs, making sure that each proposed design is compared with at least several others. Regardless of how players' opinions are elicited, one would need to answer the question: *what is our objective in aggregating opinions?* For EteRNA, the answer is simple; since biochemists are seeking a stable design for a specific molecule, but will only use a single proposed design, we would like to select a set of k designs that includes at least one great design. Including one great design is sufficient, as it will be singled out when the k selected designs are synthesized.

Of course, this setup is not confined to the realm of human computation; it also arises naturally, for example, in the process of *new product development (NPD)*. A product can have many different potential designs. A k-subset of these designs is selected based on a market survey, and (possibly costly) prototypes are manufactured for the selected designs. The prototypes are then evaluated by a focus group that definitively singles out the best design. Again, the goal is to include at least one great design among the prototypes that are evaluated.

A natural approach views elicited opinions over alternatives (designs, in the examples) as noisy estimates of a true, hidden ranking of the alternatives in terms of quality. Given such a noise model, the examples presented above call for the selection of the subset of alternatives that is most likely to contain the top alternative of the true ranking, that is, the truly strongest alternative. However, other settings may require the selection of a subset of alternatives that is most likely to possess a different property. In this paper we study this maximum likelihood estimation (MLE) framework under several objectives and noise models.

1.1 Our model and results

Our model consists of two components: the noise model and the objective function. We are interested

in two basic noise models (see, e.g, [4]) that govern how the dataset is obtained given a hidden true ranking. The first model—the *noisy comparisons* model—corresponds to independent evaluations of pairs of alternatives, where each evaluation is consistent with the true ranking with a fixed probability $p \in (1/2, 1)$. For the second model—the *noisy orders* model (also known as the *Mallows* model [14])—we imagine voters submitting complete rankings over the alternatives. The probability of observing a ranking is exponentially small in its Kendall Tau distance to the true ranking, i.e., the number of pairs of alternatives on which the two rankings disagree. The second model is consistent with EteRNA's current voting method, where in theory players are expected to consider all designs, but it is much more natural when the number of alternatives (e.g., product designs) is small. Our positive results hold with respect to a general noise model—the *noisy choice model*—that includes both basic models as special cases.

For the second component, we focus on three objective functions. Objective 1 is the one discussed above: select a subset that maximizes the probability of including the top alternative of the true ranking. Objective 2 aims to select a k-subset of alternatives that coincides with the top k alternatives of the true ranking; this objective is natural, for example, when choosing a team of workers to carry out a task that requires multiple workers with identical skills (here the alternatives are the workers). Objective 3 seeks to select an ordered tuple of alternatives of length k that maximizes the probability of coinciding with the k-prefix of the true ranking. In other words, we are trying to single out the k top alternatives as before, but in the correct order; this objective is closely aligned with web search applications. Note that the three objectives coincide when $k = 1$. We consider Objective 1 to be the most natural and important among the three, and indeed our exposition concentrates on this objective.

We prove that computing the optimal solution under all three objectives is $\mathcal{NP}$-hard for any nontrivial value of k (for Objectives 1 and 2, the case of $k = m$, where m is the number of alternatives, is trivial). However, our analytical results indicate that the optimal solutions for special cases correspond to intuitive (and sometimes tractable) methods. In particular, our analytical results focus on the case where the level of noise is very high. There are two compelling reasons for considering such noisy settings. First, if the level of noise is not very high, any reasonable method would be able to single out the true ranking with relatively little data and high confidence, and therefore maximizing our objectives becomes a nonissue. Second, as we discuss below, our experiments clearly indicate that methods that perform well in theory with respect to a very high noise level also perform well in practice.

For Objective 1, we introduce the *extended scoring method* to single out a k-subset of alternatives. We prove that under the noisy choice model, when the noise level is sufficiently high, any optimal solution is in the solution space provided by the extended scoring method.[1] Interestingly, under noisy orders the extended scoring method reduces to a well-known voting rule (which maps a vector of rankings submitted by voters to a selected alternative) called *Borda count*, and its special case for noisy comparisons also provides a highly intuitive and tractable method. To our surprise, it turns out that the extended scoring method also yields the optimal solution for Objective 2 when the noise level is sufficiently high.

For Objective 3 we present the *scored tuples* method, and prove that it gives the optimal solution when the noise level is high. Interestingly, under the noisy orders model, this method coincides with Borda count when $k = 1$, and with another famous rule known as the *Kemeny* rule when $k = m$. Intermediate values of k give a sequence of optimal voting rules that connects Borda count with Kemeny; we believe that this insight is of independent interest to social choice theory.

Finally, we conduct extensive experiments that compare the extended scoring method (for Objectives 1 and 2) and the scored tuples method (for Objective 3) with other methods. Our experiments indicate that the proposed methods, which are theoretically optimal under high noise, also outperform other methods under practical noise levels. Moreover, in cases where we are able to compute the MLE (i.e., the optimal solution), its performance and that of the proposed methods are almost indistinguishable.

1.2 Related work

Young [18] studied maximum likelihood estimators under (a variation of) the noisy orders model and two objectives: identifying the top alternative of the true ranking (which coincides with our objectives for $k = 1$), and identifying the MLE ranking. In fact, Young's paper is based on work done by the marquis de Condorcet roughly two centuries earlier [10]. Young found that when the noise level is sufficiently high, the winner according to Borda count is the MLE for the top alternative, and regardless of the noise level the Kemeny rule is an MLE for the true ranking. Our main results for Objectives 1 and 2 generalize Young's results for Borda count by extending them to a more powerful noise model and (more importantly) to dif-

[1] Subtleties with respect to tie breaking are discussed in detail in Section 3.

ferent values of k. Our result for Objective 3 connects Young's results for Borda ($k = 1$) and Kemeny ($k = m$) via a sequence of optimal voting rules for intermediate values of k, while again generalizing the noise model.

A series of relatively recent AI papers deal with voting rules as MLEs (see, e.g., [6, 8, 17, 16]). In particular, in a UAI 2005 paper, Conitzer and Sandholm [6] reverse Young's question by asking for which common voting rules there exist (constrained) noise models such that the voting rules are MLEs for the top alternative or the true ranking. One section of the paper of Xia and Conitzer [16] studies a setting where a set of alternatives must be selected, under a noise model that is different from ours, where there is a set of winners of size k but no underlying ranking. They provide a single result for this setting: an evaluation problem that is related to identifying the set of winners is $\mathcal{NP}$-hard.

Under standard noise models such as noisy orders and noisy comparisons, computing the MLE true ranking is also $\mathcal{NP}$-hard [3, 5], but there is a significant amount of work on this problem (see, e.g., [1, 2, 7, 12, 4]). For example, Braverman and Mossel [4] provide polynomial time algorithms that compute the MLE ranking with high probability. However, we see below that selecting the top k elements of the MLE ranking is not the optimal solution to our objectives (with the obvious exception of Objective 3 for $k = m$) and moreover our experiments show that this method provides poor performance with respect to our objectives in practice.

2 The Model

We denote $[k] = \{1, \ldots, k\}$. In addition, let $\arg\max^k_{s \in S} H(s)$ be the set of k-subsets of S (let $|S| = t$) with largest values under the given function H. In other words, for each order $(s_1, \ldots, s_t)$ of the elements of S such that $H(s_i) \geq H(s_{i+1})$ for all $i \in [t-1]$, $\arg\max^k_{s \in S} H(s)$ includes the set $\{s_1, \ldots, s_k\}$.

We consider a set of alternatives A; denote $|A| = m$. We will use small letters to denote specific alternatives. Let $L(A)$ be the set of permutations (which we think of as linear orders or rankings) on A, where each permutation is a bijection $\sigma : A \to \{1, 2, \ldots, m\}$. Hence, $\sigma(a)$ denotes the position of alternative a in σ, and $\sigma^{-1}(i)$ denotes the alternative at the ith position, i.e., the ith most preferred alternative when σ is viewed as a ranking over the alternatives. In particular, $\sigma(a) < \sigma(b)$ denotes that a is preferred to b under σ. We let $\sigma^{-1}([k]) = (\sigma^{-1}(1), \ldots, \sigma^{-1}(k))$ denote the ordered tuple consisting of the k-prefix of σ.

We assume that there exists a true hidden order $\sigma^* \in L(A)$ over the alternatives, which reflects their true strengths. We also make the standard assumption that σ^* is selected using a uniform prior over $L(A)$. Let $a^* = (\sigma^*)^{-1}(1)$ denote the best alternative under σ^*.

Our objective is to find a "good" set of alternatives given noisy observations. We consider two standard noise models (see, e.g., [4]).

2.1 Noisy comparisons and tournaments

In the *noisy comparisons* model, a pairwise preference $a \succ b$ denotes that alternative a is preferred to alternative b. We imagine that each pair of alternatives is presented to n voters (with possibly different sets of voters for each pair). The preferences returned by the voters are independently consistent with the true ranking σ^* with a fixed probability $1/2 < p < 1$. Hence, the dataset D is a set of comparisons where each pair of alternatives appears exactly n times, for a fixed value of n. Note that the case of $n = 1$ with relatively high value of p can also represent the aggregate opinion of many voters.

We think of the dataset D as corresponding to a slightly nonstandard *weighted tournament*. Denote by n_{ab} the number of $a \succ b$ votes. The tournament T_D is a directed graph where the vertex set is the set of alternatives, there are edges between each pair of alternatives in both directions, and the weight of the edge $e = (a, b)$ is $w_e = n_{ab}$.[2]

2.2 Noisy orders and ranked voting

In our second model, a fixed set of n voters provide rankings over all alternatives. In this *noisy orders model* (also known as the *Condorcet noise model [10]*), each ranking is generated independently by drawing pairwise preferences similarly to the noisy comparisons model, except that the process is restarted if the generated vote has a cycle (e.g. $a \succ b \succ c \succ a$). Concisely, the probability of drawing each ranking σ_i given the true order $\sigma^* = \sigma$ is proportional to $p^{\binom{m}{2} - d_K(\sigma_i, \sigma)} \cdot (1-p)^{d_K(\sigma_i, \sigma)}$. The distance $d_K(\cdot, \cdot)$ is the *Kendall tau* distance between two rankings, which counts their number of disagreements on pairs of alternatives. Probabilities are normalized using a normalization constant which can be shown to be independent of the true ranking σ^* (see, e.g., [13]). Note that this model is equivalent to the *Mallows* model [14], which is widely used in machine learning and statistics.

A *voting rule* (also known as a *social choice function*) is a function $f : L(A)^n \to A$ that accepts n rankings $(\sigma_1, \sigma_2, \cdots, \sigma_n) \in L(A)^n$ as input and outputs a selected alternative. We will informally use the same

[2] Typically tournaments have exactly one directed edge between each pair of vertices.

term to refer to functions that output a ranking of the alternative, i.e., an element of $L(A)$; such functions are also known as *social welfare functions*.

Below we consider a number of well-known voting rules; we begin with two that will play a special role. Under *Borda count* each voter i awards $m - \sigma_i(a)$ points to each alternative $a \in A$, and an alternative with most points overall is selected. The *Kemeny* rule selects a ranking $\pi \in L(A)$ that minimizes $\sum_{i=1}^{n} d_K(\pi, \sigma_i)$. Informally, the Kemeny ranking minimizes the number of disagreements with voters over pairs of alternatives.

In Section 6 we consider some additional voting rules as benchmarks. Under *Maximin*, the score of an alternative a is $\min_{a' \in A \setminus \{a\}} |\{i \in [n] : \sigma_i(a) < \sigma_i(a')\}|$. Under *plurality*, each voter i awards one point to $\sigma_i^{-1}(1)$, and under *k-approval*, one point to each of $\sigma_i^{-1}(1), \ldots, \sigma_i^{-1}(k)$.

2.3 A generalization: The noisy choice model

We next present the *noisy choice model*, a general framework that unifies both models; to the best of our knowledge this model is novel. The model is characterized by two properties. First, a notion of n_{ab} for all alternatives $a \in A$ and $b \in A \setminus \{b\}$ that denotes the degree to which a is preferred over b in the input. For a fixed n and all $a \in A$ and $b \in A \setminus \{a\}$, $n_{ab} + n_{ba} = n$.

Second, a likelihood of a dataset D given that the true order is $\sigma^* = \sigma$,

$$\Pr[D|\sigma^* = \sigma] = \frac{\gamma^{d(\sigma,D)}}{Z_\gamma}, \tag{1}$$

where the normalizing constant Z_γ is independent of σ (although it might depend on the parameters of the model), and the distance function is defined as $d(\sigma, D) = \sum_{a,b \in A:\, \sigma(a) < \sigma(b)} n_{ba}$. The distance intuitively measures the amount of disagreement between the ranking and the dataset on pairs of alternatives. Crucially, the likelihood of the dataset diminishes exponentially as its distance from the true order increases.

The noisy comparisons model and noisy orders model both fall under this more general framework. Indeed, the notion of n_{ab} is clear in both models: it is simply the number of votes that prefer a to b. It is also easy to verify that $\gamma = (1-p)/p$ for both models.

In addition to noisy comparisons and noisy orders, the noisy choice model can capture other practical models of inputs. For example, consider the model where inputs are noisy partial orders of a fixed length l generated as follows. For each voter, l alternatives are chosen uniformly at random from the set of all alternatives. Then a partial order is generated according to the Mallows Model over the chosen l alternatives. Alternatively, we can consider a model where each voter reports an unweighted tournament over the alternatives, i.e., individual preferences may be "irrational".

Note that γ is a measure of the level of noise; $\gamma \approx 0$ induces a distribution that is highly centered around the true order and $\gamma \approx 1$ induces a distribution that is close to the uniform distribution. We also remark that the distance function $d(\cdot, \cdot)$ under the noisy orders model is just the sum of Kendall tau distances of a particular ranking from the input orders.

3 Including the Top Alternative

We first consider the case where we want to select a k-subset of alternatives that is most likely to contain a^*, the top alternative of the true ranking σ^*.

Objective 1 *Given k, find a k-subset of alternatives that maximizes the probability of containing the best alternative, i.e., a subset in*

$$\arg\max_{S \subseteq A,\, |S|=k} \Pr[a^* \in S|D].$$

A crucial observation regarding Objective 1 is that

$$\Pr[a^* \in S|D] = \sum_{a \in S} \Pr[a^* = a|D] \propto \sum_{a \in S} \Pr[D|a^* = a],$$

where the last step follows from Bayes' rule and the assumption of uniform prior over rankings. The following observation is immediately implied.

Observation 3.1 *The optimal solution to Objective 1 under the noisy choice model is any subset in*

$$\arg\max^k_{a \in A} \Pr[a^* = a|D].$$

In words, we choose the k most likely alternatives according to their probabilities of coinciding with the top alternative of the true ranking. Equivalently, one could select any subset in $\arg\max^k_{a \in A} \Pr[D|a^* = a]$ since we have assumed a uniform prior over rankings. Despite Observation 3.1, finding an optimal solution to Objective 1 is computationally hard.

Theorem 3.2 *For any $k \in [m-1]$, computing an optimal solution to Objective 1 is $\mathcal{NP}$-hard under noisy orders and noisy comparisons.*

The theorem's proof appears in the full version of the paper.[3] Note that the theorem also proves $\mathcal{NP}$-hardness under the noisy choice model because noisy

[3]The full version of the paper is available from: `http://www.cs.cmu.edu/~arielpro/papers.html`.

orders and noisy comparisons are just special cases. The intuition behind the proof is as follows. Young [18] demonstrated (via an example) that when p close to 1, under a specific noise model that is very similar to the noisy comparisons model, the optimal solution with respect to Objective 1 with $k = 1$ coincides with the first element of an MLE of the true ranking σ^*. We show that this result holds under both noisy comparisons and noisy orders and also prove that computing the first element of an MLE ranking is $\mathcal{NP}$-hard by utilizing the $\mathcal{NP}$-hardness of computing the MLE ranking itself. Here, an MLE ranking is a minimum feedback ranking under noisy comparisons (see, e.g., [4]), and a Kemeny ranking under noisy orders [18], which are both $\mathcal{NP}$-hard to compute [5, 3]. Finally, we leverage the case of $k = 1$ to extend the $\mathcal{NP}$-hardness to other values of k.

Because of these computational connections, it is natural to wonder whether the optimal solution to Objective 1 is given by taking the top k elements of the MLE ranking for any k and any value of p. However, Young [18] also showed that this is not the case when p is close to 1/2 even for $k = 1$ (in fact he did not consider the case of $k > 1$). Indeed, he showed that when p is close to 1/2, in his example the MLE ranking σ is the only ranking that puts the alternative $\sigma(1)$ first and has significant probability, whereas a different alternative (the winner under Borda count) appears first in rankings that individually have smaller probability than the MLE ranking, but combined have a larger overall probability. Below we extend this result by showing that when p is sufficiently close to 1/2, the problem is indeed tractable for any value of k under our general noisy choice model, and in fact the solution coincides with intuitive methods. Although the theoretical guarantees are for the case where p is close to 1/2 (i.e., very noisy settings, $\gamma \approx 1$ in Equation (1)), the experiments in Section 6 show that the methods developed here also work well when the noise level is lower.

Our general method, which we refer to as the *extended scoring method*,[4] is well defined for input data generated according to the noisy choice model. For an alternative $a \in A$, let the score of a be

$$\mathrm{sc}(a) = \sum_{b \in A \setminus \{a\}} n_{ab}. \tag{2}$$

We choose the top k alternatives according to their score, i.e., we return a set in $\arg\max_a^k \mathrm{sc}(a)$.

Theorem 3.3 *For every n and m there exists $\gamma' < 1$ such that for all $\gamma \geq \gamma'$, the optimal solutions to Objective 1 under the noisy choice model are in $\arg\max_a^k sc(a)$.*

[4]We use the term "extended" to avoid confusion with generalized scoring rules [15].

The theorem's formulation leaves open the possibility that some sets in $\arg\max_a^k \mathrm{sc}(a)$ are far from being optimal. However, the theorem's proof in fact shows that for any $\delta > 0$ there is sufficiently large γ such that *every* $S \in \arg\max_a^k \mathrm{sc}(a)$ is optimal up to δ, i.e., for any $S' \subseteq A$ such that $|S'| = k$,

$$\Pr[a^* \in S|D] \geq \Pr[a^* \in S'|D] - \delta.$$

Proof of Theorem 3.3 Let $\gamma = 1 - \epsilon$, $\epsilon > 0$. By Observation 3.1, we know that the optimal solution is given by $\arg\max_a^k \Pr[D|a^* = a]$. In our model

$$\begin{aligned}\Pr[D|a^* = a] &= \sum_{\sigma \in L(A), \sigma^{-1}(1)=a} \Pr[D|\sigma^* = \sigma] \\ &= \sum_{\sigma \in L(A), \sigma^{-1}(1)=a} \frac{\gamma^{d(\sigma,D)}}{Z_\gamma}.\end{aligned}$$

Let $L_a(A) = \{\sigma \in L(A) | \, \sigma^{-1}(1) = a\}$. Thus $|L_a(A)| = (m-1)!$. Define an objective function $f(a) = Z_\gamma \cdot \Pr[D|a^* = a]$. Since Z_γ is a constant, the optimal solution is also given by $\arg\max_a^k f(a)$. Using $\gamma = 1-\epsilon$ and the fact that $(1-\epsilon)^t \geq 1 - t \cdot \epsilon$ for any $t \in \mathbb{N}$, we obtain the following lower bound on our objective function f:

$$f(a) \geq \hat{f}(a) = \sum_{\sigma \in L_a(A)} (1 - \epsilon \cdot d(\sigma, D)). \tag{3}$$

In addition, the gap between f and $\hat{f}$ can be upper bounded using

$$|(1-\epsilon)^t - (1 - t \cdot \epsilon)| \leq \sum_{i=2}^{t} \binom{t}{i} \cdot \epsilon^i \leq 2^t \cdot \epsilon^2$$

for $\epsilon < 1$. It follows that for every $a \in A$,

$$\begin{aligned} f(a) - \hat{f}(a) \quad &\leq \sum_{\sigma \in L_a(A)} 2^{d(\sigma,D)} \cdot \epsilon^2 \\ &\leq \epsilon^2 \cdot (m-1)! \cdot 2^{n \cdot \binom{m}{2}}. \end{aligned} \tag{4}$$

The theorem will now follow by proving two statements: $\arg\max_a^k f(a) \subseteq \arg\max_a^k \hat{f}(a)$, and $\arg\max_a^k \hat{f}(a) = \arg\max_a^k \mathrm{sc}(a)$. We use the following claim.

Claim 3.4 *For every $a \in A$,*

$$\hat{f}(a) = C_\epsilon + \epsilon \cdot (m-1)! \cdot sc(a)$$

where C_ϵ depends only on ϵ (and not on a).

Proof First, we simplify $\hat{f}$ by summing the individual terms in Equation (3).

$$\hat{f}(a) = (m-1)! - \epsilon \cdot \sum_{\sigma \in L_a(A)} d(\sigma, D). \tag{5}$$

Furthermore, note that $\sum_{\sigma \in L_a(A)} d(\sigma, D)$ equals

$$\begin{aligned}
&\sum_{\sigma \in L_a(A)} \sum_{x \in A, y \in A \setminus \{x\}} n_{yx} \cdot \mathbb{1}[\sigma(x) < \sigma(y)] \\
&= \sum_{\sigma \in L_a(A)} \sum_{x \in A, y \in A \setminus \{x\}} (n - n_{xy}) \cdot \mathbb{1}[\sigma(x) < \sigma(y)] \\
&= n \cdot \sum_{\sigma \in L_a(A)} \sum_{x \in A, y \in A \setminus \{x\}} \mathbb{1}[\sigma(x) < \sigma(y)] \\
&\quad - \sum_{x \in A, y \in A \setminus \{x\}} n_{xy} \cdot \sum_{\sigma \in L_a(A)} \mathbb{1}[\sigma(x) < \sigma(y)]
\end{aligned}$$

The symbol $\mathbb{1}$ represents the indicator function. For the first term it holds that

$$\begin{aligned}
&n \cdot \sum_{\sigma \in L_a(A)} \sum_{x \in A, y \in A \setminus \{x\}} \mathbb{1}[\sigma(x) < \sigma(y)] \\
&= n \cdot \sum_{\sigma \in L_a(A)} \binom{m}{2} = n \cdot (m-1)! \cdot \binom{m}{2}.
\end{aligned}$$

To analyze the second term we consider three cases separately: (i) $x = a$, $y \in A \setminus \{a\}$, (ii) $x \in A \setminus \{a\}$, $y = a$, and (iii) $x \in A \setminus \{a\}$, $y \in A \setminus \{a, x\}$. If $x = a$, then for every $y \in A \setminus \{a\}$ and every $\sigma \in L_a(A)$ it holds that $a \succ y$, and hence n_{ay} is multiplied by $(m-1)!$. Similarly, if $y = a$, then for every $x \in A \setminus \{a\}$, n_{xa} is multiplied by 0. If $x \in A \setminus \{a\}$ and $y \in A \setminus \{a, x\}$, then n_{xy} is multiplied by $(m-1)!/2$ because $x \succ y$ in exactly half of the rankings in the summation. Thus the second terms equals

$$\begin{aligned}
&(m-1)! \cdot \left(\sum_{y \in A \setminus \{a\}} n_{ay} + \frac{1}{2} \cdot \sum_{x \in A \setminus \{a\}, y \in A \setminus \{a,x\}} n_{xy} \right) \\
&= (m-1)! \cdot \mathrm{sc}(a) + \frac{(m-1)!}{2} \cdot n \cdot \binom{m-1}{2},
\end{aligned}$$

since $n_{xy} + n_{yx} = n$ for every $x \neq y$. Combining both terms and substituting back,

$$\begin{aligned}
&\sum_{\sigma \in L_a(A)} d(\sigma, D) \\
&= n \cdot (m-1)! \cdot \binom{m}{2} \\
&\quad - \frac{(m-1)!}{2} \cdot n \cdot \binom{m-1}{2} - (m-1)! \cdot \mathrm{sc}(a) \\
&= C'_\epsilon - (m-1)! \cdot \mathrm{sc}(a),
\end{aligned}$$

for some constant C'_ϵ independent of a. Plugging this into Equation (5) we get

$$\hat{f}(a) = C_\epsilon + \epsilon \cdot (m-1)! \cdot \mathrm{sc}(a)$$

as required. ■(Claim 3.4)

Claim 3.4 directly implies that $\arg\max_a^k \hat{f}(a) = \arg\max_a^k \mathrm{sc}(a)$. It therefore remains to prove that $\arg\max_a^k f(a) \subseteq \arg\max_a^k \hat{f}(a)$. For this purpose it is sufficient to show that for every $a, a' \in A$, $\hat{f}(a) > \hat{f}(a')$ implies $f(a) > f(a')$.

Note that $\hat{f}(a) > \hat{f}(a')$ implies $\mathrm{sc}(a) \geq \mathrm{sc}(a') + 1$ (since scores are integers) and using Claim 3.4 this implies $\hat{f}(a) \geq \hat{f}(a') + \epsilon \cdot (m-1)!$. Therefore

$$\begin{aligned}
f(a) &\geq \hat{f}(a) \geq \hat{f}(a') + \epsilon \cdot (m-1)! \\
&\geq f(a') - \epsilon^2 \cdot (m-1)! \cdot 2^{n \cdot \binom{m}{2}} + \epsilon \cdot (m-1)!.
\end{aligned}$$

The third transition follows from Equation (4). Setting $\epsilon < 2^{-n \cdot \binom{m}{2}}$, i.e., $\gamma > \gamma' = 1 - 2^{-n \cdot \binom{m}{2}}$, we have that $f(a) > f(a')$ as required. ■

The notion of score underlying the extended scoring method (Equation (2)) provides an intuitive reflection of the quality of an alternative. In the special case of the noisy comparisons model, $\mathrm{sc}(a) = \sum_{x \in A \setminus \{a\}} n_{ax}$ is just the sum of weights of the outgoing edges from a in the weighted tournament defined in Section 2. Hence, the extended scoring method reduces to picking k alternatives with highest weighted outdegrees.

In the special case of the noisy orders model,

$$\begin{aligned}
\mathrm{sc}(a) &= \sum_{x \in A \setminus \{a\}} n_{ax} = \sum_{x \in A \setminus \{a\}} \sum_{i=1}^{n} \mathbb{1}[\sigma_i(a) < \sigma_i(x)] \\
&= \sum_{i=1}^{n} \sum_{x \in A \setminus \{a\}} \mathbb{1}[\sigma_i(a) < \sigma_i(x)] = \sum_{i=1}^{n} (m - \sigma_i(a)),
\end{aligned}$$

which is exactly the Borda score of alternative a. Hence, the extended scoring method reduces to picking k alternatives with maximum Borda scores. Theorem 3.3 thus extends Young's result for Borda count [18] from the noisy orders model to the noisy choice model and from $k = 1$ to any value of k under Objective 1.

4 Identifying the Top Subset

Under Objective 1 we choose k alternatives, each of which is likely to be the top alternative a^*. However, in principle it may be the case that each of these alternatives is either the top-ranked alternative or among the bottom-ranked alternatives. In this section we seek

to identify the set of top k alternatives, but we will see that the solution in fact coincides with the solution to Objective 1.

Objective 2 *Given k, find the k-subset of alternatives that maximizes the probability of coinciding with the top k alternatives of the true hidden order, i.e., a subset in*

$$\arg\max_{S\subseteq A,|S|=k} \Pr[S = \{(\sigma^*)^{-1}(i)\}_{i\in[k]}|D].$$

It is easy to see that this objective coincides with Objective 1 for $k = 1$, and hence it is $\mathcal{NP}$-hard for $k = 1$ under noisy orders and noisy comparisons. As before, we can extend this observation to any $k \in [m-1]$ for all three models (the proof appears in the full version of the paper).

Next, we show that the extended scoring method is again optimal when $\gamma \approx 1$.

Theorem 4.1 *For every n and m there exists $\gamma' < 1$ such that for all $\gamma \geq \gamma'$, the optimal solutions to Objective 2 under the noisy choice model are in* $\arg\max_a^k sc(a)$.

Theorem 4.1 (whose proof is given in the full version of the paper) suggests that the alternatives selected by the extended scoring method are not only good candidates for the best alternative individually, but as a whole they also make a good "team", i.e, when put together they are the most likely to coincide with the top k alternatives. Theorems 4.1 and 3.3 together provide an argument in favor of the extended scoring method and, in particular, they strongly advocate Borda count when inputs are noisy orders.

5 Identifying the Top Tuple

In this section we study an objective that is even more ambitious than Objective 2: not only do we want to correctly identify the top k alternatives of σ^*, we seek to identify them *in the correct order*.

Objective 3 *Given k, find the ordered k-tuple that maximizes the probability of coinciding with the k-prefix of the true hidden order, i.e., a tuple in*

$$\arg\max_{(a_1,a_2,\ldots,a_k)\in A^k} \Pr[(\sigma^*)^{-1}([k]) = (a_1, a_2, \ldots, a_k)|D].$$

Objective 3 coincides with its predecessors when $k = 1$, and reduces to finding an MLE for the true ranking when $k = m$. In fact we are able to prove that computing the optimal solution to Objective 3 is $\mathcal{NP}$-hard for any $k \in [m]$ under noisy orders and noisy comparisons (the proof appears in the full version of the paper).

To tackle Objective 3 we propose a new method, which we call the *scored tuples* method. Similarly to the extended scoring method, it maximizes a lower bound of the stated objective function and provides optimality guarantees when $\gamma \approx 1$. We first extend the definition of the noisy choice model's distance function to compute the distance between a k-tuple $(a_1, a_2, \ldots, a_k)$ and a dataset D; this distance is defined as

$$d((a_1, a_2, \ldots, a_k), D) = \sum_{1\leq i<j\leq k} n_{a_j a_i}.$$

Next, for a k-tuple $(a_1, a_2, \ldots, a_k)$, define the score of the tuple as

$$\mathrm{sc}(a_1, a_2, \ldots, a_k) = \sum_{i=1}^{k} \mathrm{sc}(a_i) - d((a_1, a_2, \ldots, a_k), D), \tag{6}$$

where $\mathrm{sc}(a_i)$ is defined as in Equation (2). We select a k-tuple in $\arg\max_{(a_1,a_2,\ldots,a_k)\in A^k} \mathrm{sc}(a_1, a_2, \ldots, a_k)$.

It may seem that overloading the notation of $\mathrm{sc}(a)$ as we have (via Equation (2) and Equation (6) with a tuple of length 1) may create inconsistencies, but in fact the $k = 1$ case of Equation (6) reduces to Equation (2).

It turns out that the above method provides guarantees with respect to Objective 3 that are equivalent to those that we were able to obtain for previous objectives.

Theorem 5.1 *For every n and m there exists $\gamma' < 1$ such that for all $\gamma \geq \gamma'$, the optimal solutions to Objective 3 under the noisy choice model are in* $\arg\max_{(a_1,a_2,\ldots,a_k)\in A^k} sc(a_1, a_2, \ldots, a_k)$.

The proof of Theorem 5.1 is given in the full version of the paper; let us consider its implications with respect to the noisy orders model. Since maximizing $\mathrm{sc}(a_1, a_2, \ldots, a_k)$ is optimal when $\gamma \approx 1$, it must be the case (and indeed it is easy to verify) that this solution reduces to finding the Borda winner when $k = 1$ and reduces to finding the Kemeny ranking when $k = m$. Thus the optimal solution for $k = 1, \ldots, m$ induces a range of voting rules where Borda count lies on one extreme, and the Kemeny rule lies on the other. We find it intriguing (and of independent interest to social choice theory) that Borda count and Kemeny are connected via a sequence of "optimal" voting rules.

We remark that the Kemeny rule has been suggested as a method of aggregating the search results given by different search engines; in particular it can be formally argued that spam websites will appear at the bottom of the aggregate ranking [11]. However, since

users are usually only interested in the top 50 results or so, our results suggest that the scored tuples method with a relatively small k may outperform the Kemeny rule (same method with $k = m$) in this context.

Finally, an important note is that although computing $\arg\max_{(a_1,a_2,\ldots,a_k)\in A^k} \mathrm{sc}(a_1, a_2, \ldots, a_k)$ is $\mathcal{NP}$-hard in general, it can be computed in polynomial time for constant k. In some practical settings one would only need to select a constant number of alternatives (consider, e.g., the search engine example given above) and the rule can therefore be easily applied.

6 Experiments

We performed simulations under the noisy comparisons model and the noisy orders model. We experimented with various values of the accuracy parameter p (recall that $\gamma = (1-p)/p$). Although the theoretical guarantees hold only for p close to 1/2 (which corresponds to γ close to 1), we observed that for various values of p in the practical range the methods suggested in this paper significantly outperform various other known methods.

The graphs present results for particular values of p for which the probabilities are visible; the probabilities quickly go to 1 for higher values of p and go to 0 for lower values of p. The results are also verified empirically for various values of m (number of alternatives) and n (number of voters) and p (accuracy parameter) under both models. Error bars correspond to 95% confidence intervals and thus verify that the suggested methods show statistically significant improvement over other methods. In all the graphs shown below each point is computed by taking the average over 10000 iterations.

We remark that the simulations are symmetric with respect to the true order, i.e., it does not matter which true order we begin with as long as the methods do not use any information about it. If we perform simulations on a fixed true order, it becomes necessary to break ties uniformly at random, which is what we do.

For noisy orders we compared Borda count for Objectives 1 and 2, and the scored tuples method for Objective 3, against the Kemeny rule (which is the MLE ranking), k-approval, plurality, maximin, and maximizing unweighted outdegree in the tournament where there is an edge from a to b if a majority of voters rank a above b (a.k.a. *Copeland*). For noisy comparisons we compared weighted outdegree for Objectives 1 and 2, and the scored tuples method for Objective 3, against the Minimum Feedback ranking (which is the MLE ranking), unweighted outdegree, and Maximin.

For Objective 1, we were also able to estimate the actual optimal solution using Observation 3.1. For this we needed to sample rankings according to $\Pr[\sigma^* = \sigma|D]$. One obstacle is that it can be shown that even computing $\Pr[\sigma^* = \sigma|D]$ is $\mathcal{NP}$-hard. We used the fact that $\Pr[\sigma^* = \sigma|D] \propto \Pr[D|\sigma^* = \sigma] \propto \gamma^{d(\sigma,D)}$ and sampled rankings using the Metropolis-Hastings algorithm. Note though that for larger values of the parameters the optimal solution is very difficult to estimate, whereas finding a Borda winner is a trivial computational task.

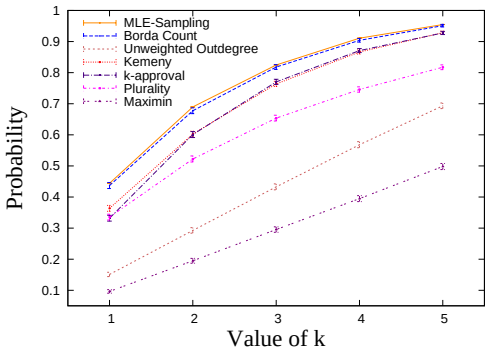


Fig. 1: Objective 1, Noisy Orders, m=10, n=10, p=0.55.

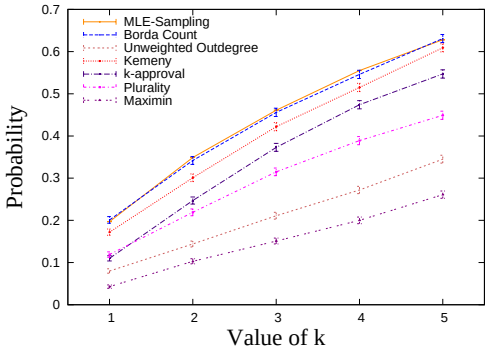


Fig. 2: Objective 1, Noisy Orders, m=20, n=100, p=0.505.

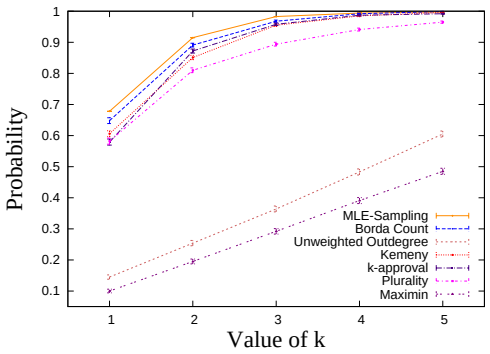


Fig. 3: Objective 1, Noisy Orders, m=10, n=10, p=0.6.

Figure 1 shows simulations for 10 alternatives, 10 voters and $p = 0.55$ under noisy orders. Clearly Borda

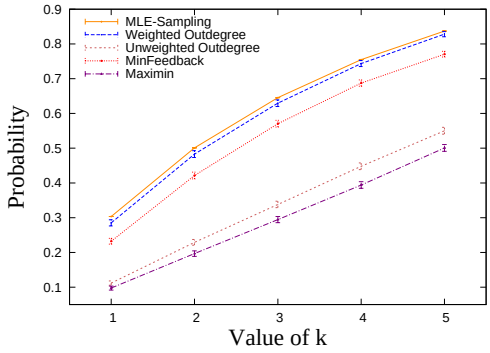


Fig. 4: Objective 1, Noisy Comparisons, m=10, n=10, p=0.55.

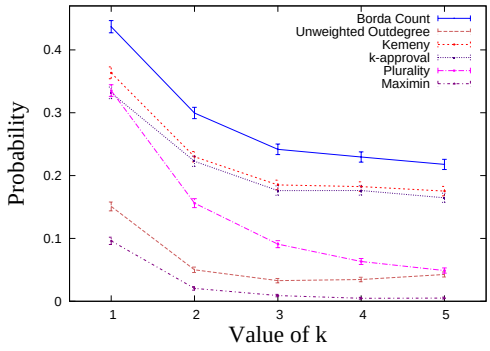


Fig. 5: Objective 2, Noisy Orders, m=10, n=10, p=0.55.

count outperforms other methods by a large margin, and in fact it is almost indistinguishable from the actual optimal solution. Extensive simulations show that similar results hold for a large number of alternatives and/or a large number of voters as well. An example with 20 alternatives, 100 voters and $p = 0.505$ under noisy orders is shown in Figure 2.

As mentioned above, the value of p is chosen such that the graphs span the probability spectrum (probabilities are not constantly 1 or 0) but similar results hold for other values of p as well. For example, Figure 4 shows that Borda count dominates other methods when $p = 0.6$. However, the margin is smaller. Indeed, for this relatively high value of p, most methods under consideration achieve excellent accuracy with only ten voters.

We also performed detailed simulations under the noisy comparisons model and observed similar results. Figure 4 shows a sample simulation for 10 alternatives, 10 voters and $p = 0.55$. In this case weighted outdegree (which is the special case of the extended scoring method) outperforms other methods by a statistically significant margin.

Under Objective 2, the extended scoring method again outperforms other methods as shown in Figure 5 (with 10 alternatives, 10 voters and $p = 0.55$ under noisy orders). As expected, under Objective 3 the scored tuples method outperforms other methods. Figure 6 shows simulations with 10 alternatives, 10 voters and $p = 0.55$ under noisy orders. We do not provide additional graphs for noisy comparisons due to lack of space, but the results are similar: the methods that are theoretically optimal for p close to $1/2$ outperform other methods for practical values of p.

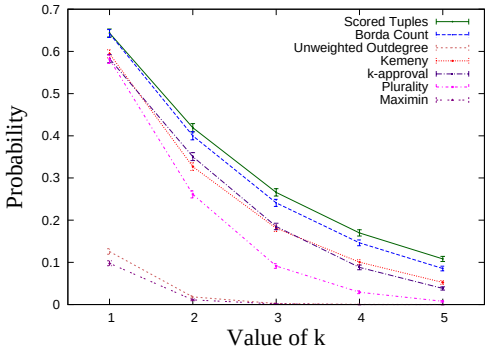


Fig. 6: Objective 3, Noisy Orders, m=10, n=10, p=0.55.

7 Discussion

We began our exposition with the human computation angle, and we would like to revisit it now that we have gained some new insights. Voting is a natural and almost ubiquitous tool in human computation systems. However, the designers of these systems usually employ simplistic voting rules such as plurality or k-approval (as EteRNA does). Our results suggest that the choice of voting rule can significantly affect performance. For example, given that we are indeed interested in singling out one great design, switching from k-approval to Borda count in EteRNA can provide significant benefits. Of course we cannot expect players to rank all the proposed designs, but we can work with partial rankings or pairwise comparisons (as described in Section 2). We find it exciting that social choice theory can help improve human computation systems. Indeed, it is difficult to apply the principles of social choice (e.g., voting rules as MLEs) to political elections, because it is almost impossible to switch voting rules. In contrast, human computation provides a perfect testbed for these principles.

References

[1] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 684–693, 2005.

[2] N. Alon. Ranking tournaments. *SIAM Journal of Discrete Mathematics*, 20(1–2):137–142, 2006.

[3] J. Bartholdi, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

[4] M. Braverman and E. Mossel. Noisy sorting without resampling. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 268–276, 2008.

[5] P. Charbit, S. Thomassé, and A. Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Combinatorics, Probability & Computing*, 16(1):1–4, 2007.

[6] V. Conitzer and T. Sandholm. Common voting rules as maximum likelihood estimators. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 145–152, 2005.

[7] V. Conitzer, A. Davenport, and H. Kalagnanam. Improved bounds for computing Kemeny rankings. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI)*, pages 620–626, 2006.

[8] V. Conitzer, M. Rognlie, and L. Xia. Preference functions that score rankings and maximum likelihood estimation. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 109–115, 2009.

[9] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, and Z. Popović. Predicting protein structures with a multiplayer online game. *Nature*, 466:756–760, 2010.

[10] M. de Condorcet. Essai sur l'application de l'analyse à la probabilité de décisions rendues à la pluralité de voix. Imprimerie Royal, 1785. Facsimile published in 1972 by Chelsea Publishing Company, New York.

[11] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference (WWW)*, pages 613–622, 2001.

[12] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 95–103, 2007.

[13] T. Lu and C. Boutilier. Learning Mallows models with pairwise preferences. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 145–152, 2011.

[14] C. L. Mallows. Non-null ranking models. *Biometrika*, 44:114–130, 1957.

[15] L. Xia and V. Conitzer. Generalized scoring rules and the frequency of coalitional manipulability. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC)*, pages 109–118, 2008.

[16] L. Xia and V. Conitzer. A maximum likelihood approach towards aggregating partial orders. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 446–451, 2011.

[17] L. Xia, V. Conitzer, and J. Lang. Aggregating preferences in multi-issue domains by using maximum likelihood estimators. In *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 399–408, 2010.

[18] H. P. Young. Condorcet's theory of voting. *The American Political Science Review*, 82(4):1231–1244, 1988.

New Advances and Theoretical Insights into EDML

Khaled S. Refaat and **Arthur Choi** and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
{krefaat,aychoi,darwiche}@cs.ucla.edu

Abstract

EDML is a recently proposed algorithm for learning MAP parameters in Bayesian networks. In this paper, we present a number of new advances and insights on the EDML algorithm. First, we provide the multivalued extension of EDML, originally proposed for Bayesian networks over binary variables. Next, we identify a simplified characterization of EDML that further implies a simple fixed-point algorithm for the convex optimization problem that underlies it. This characterization further reveals a connection between EDML and EM: a fixed point of EDML is a fixed point of EM, and vice versa. We thus identify also a new characterization of EM fixed points, but in the semantics of EDML. Finally, we propose a hybrid EDML/EM algorithm that takes advantage of the improved empirical convergence behavior of EDML, while maintaining the monotonic improvement property of EM.

1 INTRODUCTION

EDML is a recently proposed algorithm for learning MAP parameters of a Bayesian network from incomplete data (Choi, Refaat, & Darwiche, 2011). EDML is procedurally very similar to Expectation Maximization (EM), yet EDML was shown to have certain advantages, both theoretically and practically. Theoretically, EDML can in certain specialized cases provably converge in one iteration, whereas EM may require many iterations to solve the same learning problem. Empirically, a preliminary experimental evaluation suggested that EDML could find better parameter estimates than EM, in fewer iterations.

In this paper, we present a number of new results and insights on the EDML algorithm. First, we provide a simple extension of EDML to Bayesian networks over multivalued variables, whereas EDML was initially proposed for Bayesian networks over binary variables. We also show that the convex optimization problem that underlies the binary version of EDML, remains convex for the multivalued case.

Next, we identify a new and simplified characterization of EDML, which facilitates a number of theoretical observations about EDML. For example, this new characterization implies a simple, fixed-point iterative algorithm for solving the convex optimization problems underlying EDML. Moreover, we show that this fixed-point algorithm monotonically improves the solutions of these convex optimization problems, which correspond to an approximate factorization of the posterior over network parameters.

Armed with this new characterization of EDML, we go on to identify a surprising connection between EDML and EM (considering their theoretical and practical differences). In particular, we show that a fixed point of EDML is a fixed point of EM, and vice versa. This observation has a number of implications. First, it provides a new perspective on EM fixed points, based on the semantics of EDML, which was originally inspired by an approximate inference algorithm for Bayesian networks that subsumed the influential loopy belief propagation algorithm as a degenerate case (Pearl, 1988; Choi & Darwiche, 2006). Second, it suggests a hybrid EDML/EM algorithm that seeks to take advantage of the desirable properties from each: the improved convergence behavior of EDML, and the monotonic improvement property of EM.

2 TECHNICAL PRELIMINARIES

We use upper case letters (X) to denote variables and lower case letters (x) to denote their values. Variable sets are denoted by bold-face upper case letters ($\mathbf{X}$) and their instantiations by bold-face lower case letters ($\mathbf{x}$). Generally, we will use X to denote a variable

in a Bayesian network and $\mathbf{U}$ to denote its parents. A network parameter will therefore have the general form $\theta_{x|\mathbf{u}}$, representing the probability $Pr(X=x|\mathbf{U}=\mathbf{u})$.

Each variable X in a Bayesian network can be thought of as inducing a number of conditional random variables, denoted by $X|\mathbf{u}$, where the values of variable $X|\mathbf{u}$ are drawn based on the conditional distribution $Pr(X|\mathbf{u})$. Parameter estimation in Bayesian networks can be thought of as a process of estimating the distributions of these conditional random variables.

We will use θ to denote the set of all network parameters. Given a network structure G, our goal is to learn its parameters from an incomplete dataset, such as:

example	E	B	A	C
1	e_1	b_1	a_1	?
2	?	b_2	a_2	?
3	e_1	b_2	a_2	c_1

We use $\mathcal{D}$ to denote a dataset, and $\mathbf{d}_i$ to denote an example. The dataset above has three examples, with example $\mathbf{d}_2$ being $B = b_2$, and $A = a_2$.

2.1 LEARNING PARAMETERS

A commonly used measure for the quality of parameter estimates θ is their likelihood, defined as:

$$L(\theta|\mathcal{D}) = \prod_{i=1}^{N} Pr_\theta(\mathbf{d}_i),$$

where Pr_θ is the distribution induced by network structure G and parameters θ. In the case of complete data (each example fixes the value of each variable), the maximum likelihood (ML) parameters are unique and easily obtainable. Learning ML parameters is harder when the data is incomplete and the EM algorithm (Dempster, Laird, & Rubin, 1977; Lauritzen, 1995) is typically employed. EM starts with some initial parameters θ^0, called a *seed,* and successively improves on them via iteration. EM uses the update equation:

$$\theta^{k+1}_{x|\mathbf{u}} = \frac{\sum_{i=1}^{N} Pr_{\theta^k}(x\mathbf{u}|\mathbf{d}_i)}{\sum_{i=1}^{N} Pr_{\theta^k}(\mathbf{u}|\mathbf{d}_i)},$$

which requires inference on a Bayesian network parameterized by θ^k, in order to compute $Pr_{\theta^k}(x\mathbf{u}|\mathbf{d}_i)$ and $Pr_{\theta^k}(\mathbf{u}|\mathbf{d}_i)$. It is known that one run of the jointree algorithm on each example is sufficient to implement an iteration of EM, which is guaranteed to never decrease the likelihood of its estimates across iterations. EM also converges to every local maxima, given that it starts with an appropriate seed. It is common to run EM with multiple seeds, keeping the best local maxima it finds. See (Darwiche, 2009; Koller & Friedman, 2009) for recent treatments on parameter learning in Bayesian networks via EM and related methods.

EM can also be used to find Maximum a Posteriori (MAP) parameters given Dirichlet priors on network parameters. The Dirichlet prior for the parameters of a random variable $X|\mathbf{u}$ is specified by a set of exponents, $\psi_{x|\mathbf{u}}$, leading to a density $\propto \prod_x [\theta_{x|\mathbf{u}}]^{\psi_{x|\mathbf{u}}-1}$. It is common to assume that exponents are > 1, which guarantees a unimodal density. For MAP parameters, EM uses the update (see, e.g., Darwiche (2009)):

$$\theta^{k+1}_{x|\mathbf{u}} = \frac{\psi_{x|\mathbf{u}} - 1 + \sum_{i=1}^{N} Pr_{\theta^k}(x\mathbf{u}|\mathbf{d}_i)}{\psi_{X|\mathbf{u}} - |X| + \sum_{i=1}^{N} Pr_{\theta^k}(\mathbf{u}|\mathbf{d}_i)}, \quad (1)$$

where $\psi_{X|\mathbf{u}} = \sum_x \psi_{x|\mathbf{u}}$. When $\psi_{x|\mathbf{u}} = 1$, the equation reduces to the one for computing ML parameters. Moreover, using $\psi_{x|\mathbf{u}} = 2$ leads to ML parameters with Laplace smoothing. This is a common technique to deal with the problem of insufficient counts (i.e., instantiations that never appear in the dataset, leading to zero probabilities and division by zero). We will use Laplace smoothing in our experiments.

2.2 SOFT EVIDENCE

EDML makes heavy use of soft evidence (i.e., evidence that changes the distribution of a variable without necessarily fixing its value). In this section, we give an introduction to the semantics of soft evidence.

We follow the treatment of (Chan & Darwiche, 2005) for soft evidence, which models soft evidence as hard evidence on a virtual event η. In particular, soft evidence on some variable X with k values is quantified by a vector $\lambda_{x_1}, \ldots, \lambda_{x_k}$ with $\lambda_{x_i} \in [0, \infty)$. The semantics is that $\lambda_{x_1} : \cdots : \lambda_{x_k} = Pr(\eta|x_1) : \cdots : Pr(\eta|x_k)$. The soft evidence on variable X is then emulated by asserting the hard evidence η. That is, the new distribution on variable X after having asserted the soft evidence is modeled by $Pr(X|\eta)$. Note that $Pr(X|\eta)$ depends only on the ratios $\lambda_{x_1} : \cdots : \lambda_{x_k}$, not on their absolute values. Hard evidence of the form $X = x_j$ can be modeled using $\lambda_{x_i} = 0$ for all $i \neq j$, and $\lambda_{x_j} = 1$. Moreover, neutral evidence can be modeled using $\lambda_{x_i} = 1$ for all i. The reader is referred to (Chan & Darwiche, 2005) for more details.

3 BINARY EDML

EDML is a recent method for learning Bayesian network parameters from incomplete data (Choi et al., 2011). It is based on Bayesian learning in which one formulates estimation in terms of computing posterior distributions on network parameters. That is, given a Bayesian network, one constructs a corresponding *meta network* in which parameters are explicated as variables, and on which the given dataset $\mathcal{D}$ can be asserted as evidence; see Figure 1. One then estimates

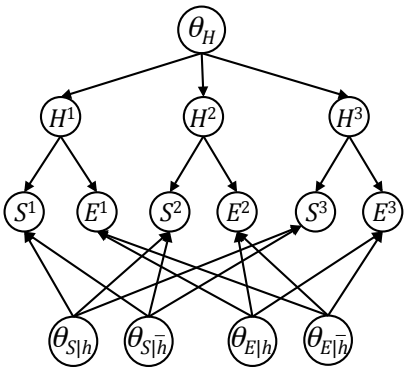


Figure 1: A meta network induced from a base network $S \longleftarrow H \longrightarrow E$. The CPTs here are based on standard semantics; see, e.g., (Darwiche, 2009, Ch. 18).

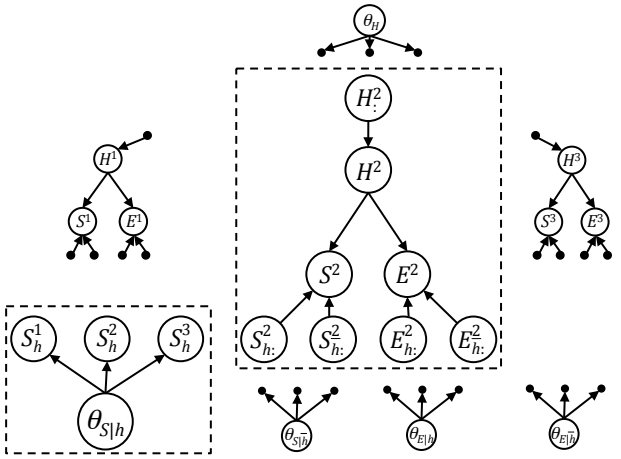


Figure 2: An edge-deleted network obtained from the meta network in Figure 1. Highlighted are the island for example $\mathbf{d}_2$ and the island for parameter set $\theta_{S|h}$.

parameters by considering the posterior distribution obtained from conditioning the meta network on the given dataset $\mathcal{D}$. Suppose for example that the meta network induces distribution $\mathcal{P}$ and let θ denote an instantiation of variables that represent parameters in the meta network. One can then obtain MAP parameter estimates by computing $\text{argmax}_\theta \mathcal{P}(\theta|\mathcal{D})$ using inference on the meta network; see (Darwiche, 2009) for an example treatment of Bayesian learning.

It is known that meta networks tend to be too complex for exact inference algorithms, especially when the dataset is large enough. The basic insight behind EDML was to adapt a specific approximate inference scheme to meta networks with the goal of computing MAP parameter estimates. In particular, the original derivation of EDML adapted the approximate inference algorithm proposed by (Choi & Darwiche, 2006), in which edges are deleted from a Bayesian network

Algorithm 1 Binary EDML

input:

G: A Bayesian network structure
$\mathcal{D}$: An incomplete dataset $\mathbf{d}_1, \ldots, \mathbf{d}_N$
θ: An initial parameterization of structure G
$\alpha_{X|\mathbf{u}}, \beta_{X|\mathbf{u}}$: Beta prior for each random variable $X|\mathbf{u}$

1: **while** not converged **do**
2: $Pr \leftarrow$ distribution induced by θ and G
3: **Compute** Bayes factors:

$$\kappa^i_{x|\mathbf{u}} \leftarrow \frac{Pr(x\mathbf{u}|\mathbf{d}_i)/Pr(x|\mathbf{u}) - Pr(\mathbf{u}|\mathbf{d}_i) + 1}{Pr(\bar{x}\mathbf{u}|\mathbf{d}_i)/Pr(\bar{x}|\mathbf{u}) - Pr(\mathbf{u}|\mathbf{d}_i) + 1}$$

for each family instantiation $x\mathbf{u}$ and example $\mathbf{d}_i$
4: **Update** parameters:

$$\theta_{x|\mathbf{u}} \leftarrow \underset{p}{\text{argmax}}\ [p]^{\alpha_{X|\mathbf{u}}-1}[1-p]^{\beta_{X|\mathbf{u}}-1} \prod_{i=1}^{N} [\kappa^i_{x|\mathbf{u}} \cdot p - p + 1]$$

5: **return** parameterization θ

to make it sparse enough for exact inference, followed by a compensation scheme that attempts to improve the quality of the approximations obtained from the edge-deleted network. The adaptation of this inference method to meta networks is shown in Figure 2. The two specific techniques employed here were to augment each edge $\theta_{X|\mathbf{u}} \longrightarrow X^i$ by an auxiliary variable $X^i_{\mathbf{u}}$, leading to $\theta_{X|\mathbf{u}} \longrightarrow X^i_{\mathbf{u}} \longrightarrow X^i$, where $X^i_{\mathbf{u}} \longrightarrow X^i$ is an equivalence edge. This is followed by deleting the equivalence edge. This technique yielded a disconnected meta network with two classes of subnetworks, called *parameter islands* and *network islands.*

Deleting edges, as proposed by (Choi & Darwiche, 2006), leads to introducing two auxiliary nodes in the Bayesian network for each deleted edge. Moreover, approximate inference by edge deletion follows the deletion process by a compensation scheme that searches for appropriate CPTs of these auxiliary nodes. As it turns out, the search for these CPTs, which is done iteratively, was amenable to a very intuitive interpretation as shown in (Choi et al., 2011).

In particular, one set of CPTs corresponded to soft evidence on network parameters, where each network island contributes one piece of soft evidence for each network parameter. The second set of CPTs corresponded to updated parameter estimates, where each parameter island contributes an estimate of its underlying parameter set. This interpretation was the basis for the form of EDML shown in Algorithm 1. This particular version of EDML, introduced in (Choi et al., 2011), assumes that all network variables are binary. Binary EDML, as we shall call it, iterates just

like EM does, producing new estimates after each iteration. However, EDML iterations can be viewed as having two phases. In the first phase, each example in the data set is used to compute a piece of soft evidence on each parameter set (Line 3 of Algorithm 1). In the second phase, the pieces of soft evidence pertaining to each parameter set are used to compute a new estimate of that set (by solving the convex optimization problem on Line 4 of Algorithm 1). The process repeats until some convergence criteria is met. Aside from this optimization task, EM and EDML have the same computational complexity.

4 MULTIVALUED EDML

One contribution of this paper is the extension of binary EDML so that it handles multivalued variables as well. In principle, the extension turns out to be straightforward and is depicted in Algorithm 3. However, two issues require further discussion. The first concerns the specification of soft evidence for multivalued variables. The second is confirming that the optimization problem corresponding to a parameter island (on Line 4 of Algorithm 3) remains strictly concave, therefore, admitting unique solutions. We will consider both issues next.

4.1 EXAMPLES AS SOFT EVIDENCE

The first key concept of EDML is to interpret a data example $\mathbf{d}_i$ in the dataset as soft evidence on a conditional random variable $X|\mathbf{u}$. As mentioned earlier, soft evidence on a variable is modeled using a vector of parameters, one for each value of the variable. We will therefore use $\lambda_{x|\mathbf{u}}$ to denote the parameter pertaining to value x of variable $X|\mathbf{u}$.

EDML uses Equation 2 in Algorithm 3 to compute soft evidence. In particular, example $\mathbf{d}_i$ is viewed as soft evidence on conditional random variable $X|\mathbf{u}$ that is quantified as follows:

$$\lambda^i_{x|\mathbf{u}} \leftarrow Pr(x\mathbf{u}|\mathbf{d}_i)/Pr(x|\mathbf{u}) - Pr(\mathbf{u}|\mathbf{d}_i) + 1$$

We will not derive this equation here as it resembles the one for binary EDML. We will, however, discuss some of its key properties in this and further sections.

Consider the case when the example $\mathbf{d}_i$ is inconsistent with the parent instantiation $\mathbf{u}$. In this case, the example should be irrelevant to variable $X|\mathbf{u}$. Equation 2 does the right thing here as it reduces to 1 for all values of x, which amounts to neutral evidence.

Another special case is when example $\mathbf{d}_i$ is complete; that is, it has no missing values. In this case, one can verify that if $\mathbf{d}_i$ is consistent with $\mathbf{u}$ (relevant), then $\lambda^i_{x|\mathbf{u}} = 0$ for all values x except the value $x^\star$ consistent with $\mathbf{d}_i$. This is equivalent to hard evidence in favor of $x^\star$. On the other hand, if $\mathbf{d}_i$ is inconsistent with $\mathbf{u}$ (irrelevant), then $\lambda^i_{x|\mathbf{u}} = 1$ for all values x, providing neutral evidence. In a nutshell, a complete example provides either hard evidence, if it is relevant; or neutral evidence, if it is irrelevant.

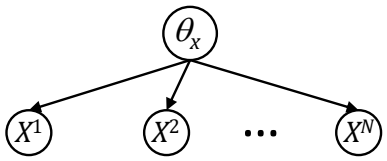


Figure 3: Learning from independent, hard observations $X^1, \ldots, X^N$. The distribution of variable X is specified by parameter set θ_X.

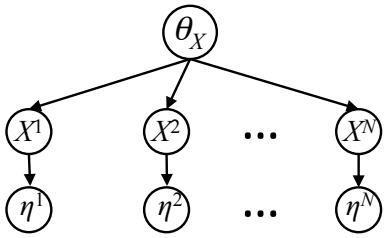


Figure 4: Learning from independent, soft observations $\eta_1, \ldots, \eta_N$. The distribution of variable X is specified by parameter set θ_X.

4.2 LEARNING FROM SOFT EVIDENCE

Consider the standard learning problem depicted in Figure 3. Here, we have a variable X that takes k values $x_1, \ldots, x_k$ and has a distribution specified by a parameter set $\theta_X : \theta_{x_1}, \ldots, \theta_{x_k}$. That is, each parameter θ_{x_i} represents the corresponding probability $Pr(X = x_i)$. Suppose further that we have a Dirichlet prior $\rho(\theta_X)$ on the parameter set with exponents ψ_{x_i} greater than one. A standard learning problem here is to compute the MAP estimates of parameter set θ_X given N independent observations on the variable X. MAP estimates are known to be unique in this case and have a corresponding closed form. In particular, it is known that the posterior $\rho(\theta_X|X^1, \ldots, X^N)$ is a unimodal Dirichlet and, hence, has a unique maximum; see for example (Darwiche, 2009).

EDML is based on a variant of this learning problem in which we are given N *soft observations* on variable X instead of hard observations. This variant is shown in Figure 4, where each observation X^i has a child η^i that is used to emulate soft evidence on X^i. That is, to represent soft evidence $\lambda^i_{x_1}, \ldots, \lambda^i_{x_k}$, we simply choose the CPT for η^i so that $Pr(\eta^i|x_1) : \cdots : Pr(\eta^i|x_k) =$

Algorithm 2 EM

input:
G: A Bayesian network structure
$\mathcal{D}$: An incomplete dataset $\mathbf{d}_1, \ldots, \mathbf{d}_N$
θ: An initial parameterization of structure G
ψ: A Dirichlet prior for each parameter set $\theta_{X|\mathbf{u}}$

1: **while** not converged **do**
2: $Pr \leftarrow$ distribution induced by θ and G
3: **Compute** probabilities:

$$Pr(x\mathbf{u}|\mathbf{d}_i) \quad \text{and} \quad Pr(\mathbf{u}|\mathbf{d}_i)$$

for each family instantiation $x\mathbf{u}$ and example $\mathbf{d}_i$
4: **Update** parameters:

$$\theta_{x|\mathbf{u}} \leftarrow \frac{\psi_{x|\mathbf{u}} - 1 + \sum_{i=1}^{N} Pr(x\mathbf{u}|\mathbf{d}_i)}{\psi_{X|\mathbf{u}} - |X| + \sum_{i=1}^{N} Pr(\mathbf{u}|\mathbf{d}_i)}$$

5: **return** parameterization θ

Algorithm 3 Multivalued EDML

input:
G: A Bayesian network structure
$\mathcal{D}$: An incomplete dataset $\mathbf{d}_1, \ldots, \mathbf{d}_N$
θ: An initial parameterization of structure G
ψ: A Dirichlet prior for each parameter set $\theta_{X|\mathbf{u}}$

1: **while** not converged **do**
2: $Pr \leftarrow$ distribution induced by θ and G
3: **Compute** soft evidence parameters:

$$\lambda^i_{x|\mathbf{u}} \leftarrow Pr(x\mathbf{u}|\mathbf{d}_i)/Pr(x|\mathbf{u}) - Pr(\mathbf{u}|\mathbf{d}_i) + 1 \qquad (2)$$

for each family instantiation $x\mathbf{u}$ and example $\mathbf{d}_i$
4: **Update** parameters:

$$\theta_{X|\mathbf{u}} \leftarrow \operatorname*{argmax}_{\hat{\theta}_{X|\mathbf{u}}} \prod_x [\hat{\theta}_{x|\mathbf{u}}]^{\psi_{x|\mathbf{u}}-1} \prod_{i=1}^{N} \sum_x \lambda^i_{x|\mathbf{u}} \hat{\theta}_{x|\mathbf{u}} \qquad (3)$$

5: **return** parameterization θ

$\lambda^i_{x_1} : \cdots : \lambda^i_{x_k}$.

Note here that the posterior density $\rho(\theta_X|\eta^1 \ldots \eta^N)$ is no longer Dirichlet, but takes the more complex form given by Equation 3 of Algorithm 3. Yet, we have the following result.

Theorem 1 *Given N soft observations η^i on a variable X, and a Dirichlet prior on its parameters θ_X, with Dirichlet exponents $\psi_x > 1$, the posterior density $\rho(\theta_X|\eta^1 \ldots \eta^N)$ is strictly log concave.*

Therefore, the posterior density, $\rho(\theta_X|\eta^1 \ldots \eta^N)$, has a unique maximum. We next provide a simple iterative method for obtaining the maximum in this case.

5 SIMPLE EDML

Multivalued EDML, as given in Algorithm 3, works as follows. We start with some initial parameter estimates, just like EM. We then iterate, while performing two steps in each iteration. In the first step, each example $\mathbf{d}_i$ in the dataset is used to compute soft evidence on each variable $X|\mathbf{u}$, as in Equation 2 of Algorithm 3. Using this soft evidence, a learning problem is set up for the parameter set $\theta_{X|\mathbf{u}}$ as given in Figure 4, which is a learning *sub*-problem in the context of Equation 3 of Algorithm 3. The solution to this learning sub-problem provides the next estimate for parameter set $\theta_{X|\mathbf{u}}$. The process repeats. In principle, any appropriate optimization algorithm could be used to solve each of these learning sub-problems.

We will next derive a simpler description of EDML that has two key components. First, we will provide a convergent update equation that iteratively solves the optimization problem in Equation 3 of Algorithm 3. Second, we will use this update equation to provide a characterization of EDML's fixed points.

First, consider the likelihood:

$$Pr(\eta^1, \ldots, \eta^N \mid \theta_X) = \prod_{i=1}^{N} Pr(\eta^i \mid \theta_X)$$
$$= \prod_{i=1}^{N} \sum_x Pr(\eta^i \mid x) Pr(x \mid \theta_X) = \prod_{i=1}^{N} \sum_x \lambda^i_x \theta_x$$

Next, we have the posterior:

$$\rho(\theta_X \mid \eta^1, \ldots, \eta^N) \propto \rho(\theta_X) Pr(\eta^1, \ldots, \eta^N \mid \theta_X)$$
$$= \rho(\theta_X) \prod_{i=1}^{N} \sum_x \lambda^i_x \theta_x$$

Assuming a Dirichlet prior over parameter set θ_X, with Dirichlet exponents ψ_x, the log of the posterior is:

$$\log \rho(\theta_X \mid \eta^1 \ldots \eta^N)$$
$$= \sum_x (\psi_x - 1) \log \theta_x + \sum_{i=1}^{N} \log \sum_x \lambda^i_x \theta_x + \gamma$$

where γ is a constant that is independent of θ_X, which we can ignore. By Theorem 1 we know that the log of the posterior is strictly concave when $\psi_x > 1$.

To get the unique maximum of the log posterior, we solve the optimization problem:

$$\begin{aligned} &\text{minimize} && -\log \rho(\theta_X \mid \eta^1, \ldots, \eta^N) \\ &\text{subject to} && \sum_x \theta_x = 1 \end{aligned}$$

In order to characterize the unique maximum of the log posterior, we start by taking the Lagrangian:

$$L(\theta_X, \nu) = -\log \rho(\theta_X \mid \eta^1, \ldots, \eta^N) + \nu \cdot (\sum_x \theta_x - 1)$$

where ν is a Lagrange multiplier. We get the following condition for the optimal parameter estimates, by setting the gradient of $L(\theta_X, \nu)$ with respect to θ_X to zero:

$$\theta_x = \frac{\psi_x - 1 + \sum_{i=1}^{N} \frac{\lambda^i_x \theta_x}{\sum_{x^*} \lambda^i_{x^*} \theta_{x^*}}}{\psi_X - |X| + N}$$

where $\psi_X = \sum_x \psi_x$, and where we used the constraint $\sum_x \theta_x = 1$ to identify that $\nu = \psi_X - |X| + N$.

The above equation leads to a much stronger result.

Theorem 2 *The following update equation monotonically increases the posterior $\rho(\theta_{X|\mathbf{u}} \mid \eta^1, \ldots, \eta^N)$:*

$$\theta^t_{x|\mathbf{u}} = \frac{\psi_{x|\mathbf{u}} - 1 + \sum_{i=1}^{N} \frac{\lambda^i_{x|\mathbf{u}} \theta^{t-1}_{x|\mathbf{u}}}{\sum_{x^*} \lambda^i_{x^*|\mathbf{u}} \theta^{t-1}_{x^*|\mathbf{u}}}}{\psi_{X|\mathbf{u}} - |X| + N} \qquad (4)$$

This theorem suggests a convergent iterative algorithm for solving the convex optimization problem of Equation 3 in Algorithm 3. First, we start with some initial parameter estimates $\theta^0_{x|\mathbf{u}}$ at iteration $t = 0$. For iteration $t > 0$, we use the above update to compute parameters $\theta^t_{x|\mathbf{u}}$ given the parameters $\theta^{t-1}_{x|\mathbf{u}}$ from the previous iteration. If at some point, the parameters of one iteration do not change in the next (in practice, up to some limit), we say that the iterations have converged to a fixed point. The above theorem, together with Theorem 1, shows that these updates are convergent to the unique maximum of the posterior.

Given Equation 4, one can think of two types of *iterations* in EDML: local and global. A *global* iteration corresponds to executing Lines 2–4 of Algorithm 3 and is similar to an EM iteration. Within each global iteration, we have *local* iterations which correspond to the evaluations of Equation 4. Note that each parameter set $\theta_{X|\mathbf{u}}$ has its own local iterations, which are meant to find the optimal values of this parameter set. Moreover, the number of local iterations for each parameter set $\theta_{X|\mathbf{u}}$ may be different, depending on the soft evidence pertaining to that set (i.e., $\lambda^i_{x|\mathbf{u}}$) and depending on how Equation 4 is seeded for that particular parameter set (i.e., $\theta^0_{x|\mathbf{u}}$).

This leads to a number of observations on the difference between EM updates (Equation 1) and EDML updates (Equation 4). From a time complexity viewpoint, an EM update implies exactly one local iteration for each parameter set since Equation 1 needs to be evaluated only once for each parameter set. As mentioned earlier, however, an EDML update requires a varying number of local iterations. We have indeed observed that some parameter sets may require several hundred local iterations, depending on the seed of Equation 4 and the convergence criteria used. Another important observation is that EDML has a secondary set of seeds, as compared to EM, which are needed to start off Equation 4 at the beginning of each global iteration of EDML. In our experiments, we seed Equation 4 using the parameter estimates obtained from the previous global iteration of EDML. We note, however, that the choice of these secondary seeds is a subject that can significantly benefit from further research.

6 EDML FIXED POINTS

One of the more well known facts about EM is that its fixed points are precisely the stationary points of the log-likelihood function (or more generally, the posterior density when Dirichlet priors are used). This property has a number of implications, one of which is that EM is capable of converging to every local maxima of the log-likelihood, assuming that the algorithm is seeded appropriately.

In this section, we show that the fixed points of EDML are precisely the fixed points of EM. We start by formally defining what a fixed point is.

Both EM and EDML can be viewed as functions $f(\theta)$ that take a network parameterization θ and returns another network parameterization $f(\theta)$. Each algorithm is seeded with initial parameters θ^0. After the first iteration, each algorithm produces the next parameters $\theta^1 = f(\theta^0)$. More generally, at iteration i, each algorithm produces the parameters $\theta^{i+1} = f(\theta^i)$. When $\theta^{i+1} = \theta^i$, we say that parameters θ^i are a fixed point for an algorithm. We also say that the algorithm has converged to θ^i. We now have the following results.

Theorem 3 *A parameterization θ is a fixed point for EDML if and only if it is a fixed point for EM.*

The proof of this theorem rests on two observations. First, using the update equation of EM given on Line 4 of Algorithm 2, one immediately gets that the EM fixed points are characterized by the following equation

$$Pr(x|\mathbf{u}) = \frac{\psi_{x|\mathbf{u}} - 1 + \sum_{i=1}^{N} Pr(x\mathbf{u}|\mathbf{d}_i)}{\psi_{X|\mathbf{u}} - |X| + \sum_{i=1}^{N} Pr(\mathbf{u}|\mathbf{d}_i)} \qquad (5)$$

That is, we can test whether a network parameterization θ is a fixed point for EM by simply checking the probability distribution Pr it induces to see if satisfies the above equation (this is actually a set of equations, one for each family instantiation $x\mathbf{u}$ in the network).

Consider now EDML updates as given by Equation 4 and suppose that we have reached a fixed point, where $\theta^t_{x|\mathbf{u}} = \theta^{t-1}_{x|\mathbf{u}} = Pr(x|\mathbf{u})$ for all $x\mathbf{u}$. If we now replace each $\lambda^i_{x|\mathbf{u}}$ by its corresponding value in Equation 2 of Algorithm 3, we obtain, after some simplification:

$$Pr(x|\mathbf{u}) = \frac{\psi_{x|\mathbf{u}} - 1 + \sum_{i=1}^{N} \frac{\lambda^i_{x|\mathbf{u}} Pr(x|\mathbf{u})}{\sum_{x^\star} \lambda^i_{x^\star|\mathbf{u}} Pr(x^\star|\mathbf{u})}}{\psi_{X|\mathbf{u}} - |X| + N}$$
$$= \frac{\psi_{x|\mathbf{u}} - 1 + \sum_{i=1}^{N} Pr(x\mathbf{u}|\mathbf{d}_i) + Pr(\neg\mathbf{u}|\mathbf{d}_i) Pr(x|\mathbf{u})}{\psi_{X|\mathbf{u}} - |X| + N} \quad (6)$$

Thus, a parameterization θ is a fixed point for EDML iff it satisfies Equation 6. After noting that $N = \sum_{i=1}^{N} Pr(\mathbf{u}|\mathbf{d}_i) + Pr(\neg\mathbf{u}|\mathbf{d}_i)$, we rearrange Equation 6 to obtain Equation 5, which is the characteristic equation for EM fixed points. Thus, a parameterization θ satisfies Equation 6 iff it is a fixed point for EM.

7 HYBRID EDML/EM

By Theorem 3, EDML and EM share the same fixed points. This result is fairly surprising, considering the theoretical and practical differences between the two algorithms. For example, certain specialized situations were identified where EDML converges to optimal parameter estimates in a single global iteration (regardless of how it is seeded), whereas EM may require many iterations to converge to the same estimates (Choi et al., 2011). On the other hand, EDML is not guaranteed to monotonically improve its estimates after each global iteration as EM does (improvement here is in terms of increasing the likelihood of estimates, or the MAP when Dirichlet priors are used).

By carefully examining EDML updates (Equation 4), one can see that the quantities it needs to perform a single local iteration are the same quantities needed to perform an EM update (Equation 1). Hence, without any additional computational effort, one can obtain EM updates as a side effect of computing EDML updates (the converse is not true since Equation 4 may require many local iterations). As both algorithms share the same fixed points, it thus makes sense to consider a hybrid algorithm that takes advantage of the improved theoretical and practical benefits of EDML (with respect to faster convergence) and the monotonic improvement property of EM.

We thus propose a very simple hybrid algorithm. At each global iteration of EDML, we also compute EM updates simultaneously. We then evaluate each update and choose the one that increases the posterior the most. As EM is guaranteed to improve the posterior, this hybrid algorithm is also trivially guaranteed to monotonically improve the posterior, therefore, inheriting the most celebrated feature of EM.

While additionally computing an EM update is not much overhead if one is computing an EDML update, evaluating both updates with respect to the posterior incurs a non-trivial cost. As we shall see in the following section, however, the improved convergence behavior of EDML enables this hybrid algorithm to realize improvements over EM, both in terms of faster convergence (i.e., number of global iterations) and in terms of time (i.e., when taking both global and local iterations under consideration).

8 EXPERIMENTAL RESULTS

In our first set of experiments, we show that simple EDML, as compared to EM, can often find better estimates in fewer global iterations. In our second set of experiments, we show that a hybrid EDML/EM algorithm can find better estimates in less time, as well as fewer global iterations. We use the following networks: alarm, andes, asia, diagnose, pigs, spect, water, and win95pts. Network spect is a naive Bayes network induced from a dataset in the UCI ML repository, with 1 class variable and 22 attributes. Network diagnose is from the UAI 2008 evaluation. The other networks are commonly used benchmarks.[1]

Using these networks, we simulated data sets of a certain size (2^{10}), then made the data incomplete by randomly selecting a certain percentage of the nodes to be hidden (10%, 25%, 35%, 50%, and 70%). For each of these cases, and for each network, we further generated 3 data sets at random. A combination of a network, a percentage of hidden nodes, and a generated data set constitutes a learning problem. Both EM and EDML are seeded with the same set of randomly generated parameters. Local EDML iterations are seeded with the estimates of the previous global iteration.

8.1 EXPERIMENTS I

First, we study the behavior of EDML compared to EM, with respect to global iterations (more on the computation time, in the next section). For every learning problem, we run both EM and EDML[2] for 1000 global iterations and identify the best MAP estimates achieved by either of them, for the purpose of evaluation. In each global iteration, EM and EDML

[1] Available at `http://www.cs.huji.ac.il/site/labs/compbio/Repository/` and `http://genie.sis.pitt.edu/networks.html`

[2] Soft evidence and parameter updates are damped in EDML, which is typical for algorithms like loopy belief propagation, which EDML is, in part, inspired by (Choi & Darwiche, 2006).

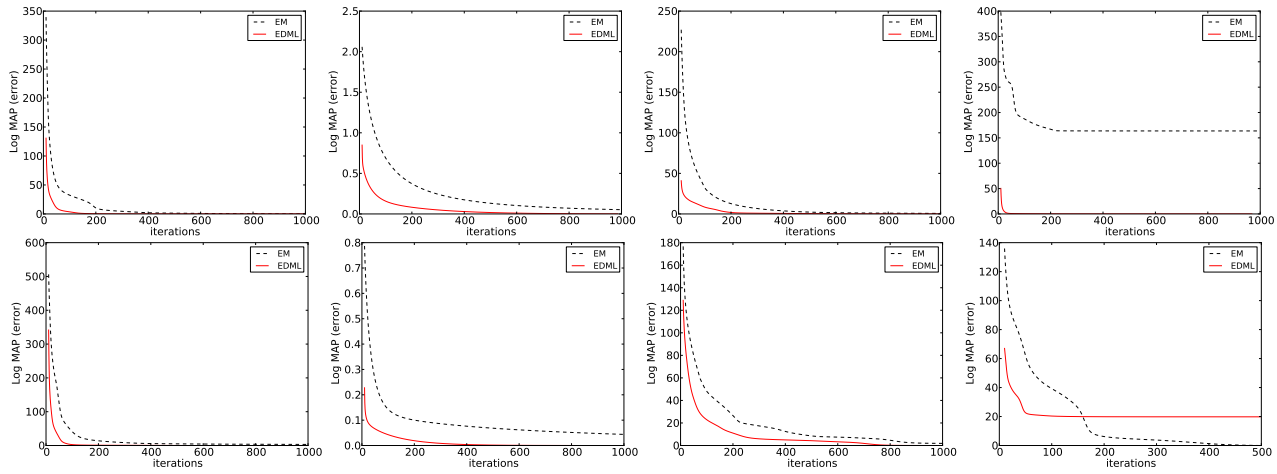


Figure 5: MAP Error of parameter estimates over iterations. Going right on the x-axis, we have increasing iterations. Going up on the y-axis, we have increasing error. EDML is depicted with a solid red line, and EM with a dashed black line. The curves are from left to right, in pairs, for the networks: andes, asia, diagnose, and alarm. Each pair of curves represents a selection of two different datasets of size 2^{10}.

each try to improve their current estimates by improving the posterior (again, EM is provably guaranteed to increase the posterior, while EDML is not). The difference between the (log) posterior of the current estimates, and the best (log) posterior found, by either algorithm, is considered to be the error. The error is measured at every global iteration of EM and EDML until it decreases below 10^{-4}. Table 1 summarizes the results by showing the percentage of global iterations in which each algorithm had less error than the other. In the global iterations where an algorithm had less error, the factor by which it decreases the error of the other algorithm, on average, is computed, and is considered the relative improvement: r and r', for EM and EDML, respectively. Table 1 shows the results for three different breakdowns: (1) by different networks, (2) by different hiding percentages, and (3) on average.

We see here that EDML can obtain better estimates than EM in much fewer global iterations. Interestingly, this is the case even though EDML is not guaranteed to improve estimates after each global iteration, as EM does. Another interesting observation is that decreasing the percentage of hidden nodes widens the gap between EDML and EM, in favor of EDML. This is not surprising though since the approximate inference scheme on which EDML is based becomes more accurate with more observations. In particular, the local optimization problems that EDML solves exactly and independently, become more independent with more observations (i.e., as the dataset becomes more complete). Figure 5 highlights a selection of error curves given by EM and EDML for different learning problems. One can see that in most cases shown, the EDML error goes to zero much faster than EM.

Table 1: Speedup results (iterations)

category	% EDML	% EM	r	r'
alarm	89.25%	10.75%	76.21%	76.44%
andes	75.89%	24.11%	88.95%	79.29%
asia	99.01%	0.99%	92.05%	76.91%
diagnose	78.99%	21.01%	77.99%	80.18%
pigs	83.34%	16.66%	83.51%	60.57%
spect	86.65%	13.35%	82.70%	79.96%
water	82.77%	17.23%	91.55%	83.78%
win95pts	78.73%	21.27%	91.75%	79.89%
hiding 10%	93.82%	6.18%	84.59%	87.13%
hiding 25%	90.95%	9.05%	83.83%	75.70%
hiding 35%	82.24%	17.76%	86.26%	75.09%
hiding 50%	77.61%	22.39%	87.8%	80.21%
hiding 70%	75.65%	24.35%	84.48%	74.21%
average	83.05%	16.95%	85.41%	76.96%

8.2 EXPERIMENTS II

Our first set of experiments showed that EDML can obtain better estimates in significantly fewer global iterations than EM. A global EDML iteration, however, is more costly than a global EM iteration as EDML performs local iterations which are needed to solve the convex optimization problem associated with each parameter set. Thus, EDML can potentially take more time to converge than EM in some cases, therefore reducing the overall benefit over EM, in terms of time (more on this later).[3] EDML can still perform favorably time-wise compared to EM, but we show here

[3]This could be alleviated, for example, by better seeding of the local EDML iterations (to speed up convergence of the local iterations), or by performing the local EDML iterations in parallel, across parameter sets.

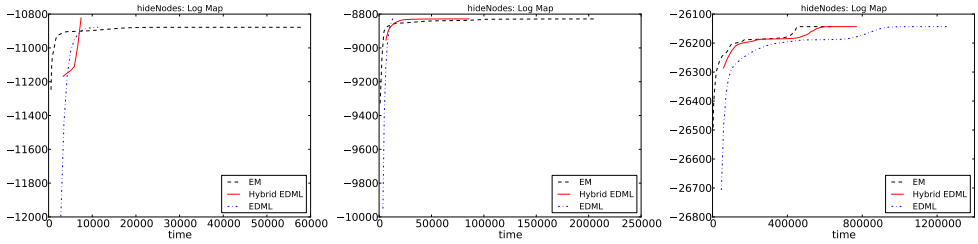


Figure 6: MAP of parameter estimates over time. Going right on the x-axis, we have increasing time (ms). Going up on the y-axis, we have increasing MAP. Hybrid EDML is depicted with a solid red line, EM with a dashed black line, and EDML with a blue dotted dashed line. The curves are from left to right for the following problems: alarm with hiding 25%, win95pts with hiding 35%, and water with hiding 50%.

Table 2: Speedup results (time)

network	% Hybrid	% EM	s	s'
alarm	46.67%	53.33%	75.80%	56.22%
andes	53.33%	46.67%	42.27%	47.08%
asia	66.67%	33.33%	70.37%	41.00%
diagnose	26.67%	73.33%	52.71%	43.14%
pigs	73.33%	26.67%	52.35%	31.32%
spect	100%	0%	98.03%	—
water	35.71%	64.29%	46.15%	43.55%
win95pts	80.00%	20.00%	66.37%	54.95%
average	60.50%	39.50%	67.45%	45.55%

that a hybrid EDML/EM can go even further.

We first run EM until convergence (or until it exceeds 1000 iterations). Second, we run our hybrid EDML/EM until it achieves the same quality of parameter estimates as EM (or until it exceeds 1000 iterations). To summarize the results, Table 2 shows the percentage of learning problems in which each algorithm was faster than the other. In the cases where the hybrid EDML/EM algorithm was faster, the average percentage by which it decreases the execution time of EM is reported as the speedup (s). The speedup for the cases in which EM was faster is given by s'.

The results suggest that hybrid EDML/EM can be used to get better estimates in less time. Specifically, on average, the hybrid method decreases the execution time by a factor of 3.07 in about 60.50% of the cases, and, in the rest of the cases, EM was faster by a factor of roughly 1.84. This is particularly interesting in light of the non-trivial overhead associated with hybrid EDML/EM, as it has to evaluate both EDML and EM estimates in order to select the better estimate. In comparison, EDML alone was on average faster than EM by a factor of 2.59 times in about 54.17% of the cases, whereas EM was faster than EDML alone by a factor of 2.38 in the remaining 45.83% of the cases.

Figure 6 shows a sample of the time curves showing two cases where EDML/EM reached better MAP estimates, faster than EM, and one case where EDML/EM finished slightly after EM. EDML alone is also plotted in Figure 6 where it is usually slower than EDML/EM except in some cases; see the center plot in Figure 6.

9 RELATED WORK

EM has played a critical role in learning probabilistic graphical models and Bayesian networks (Dempster et al., 1977; Lauritzen, 1995; Heckerman, 1998). However, learning (and Bayesian learning in particular) remains challenging in a variety of situations, particularly when there are hidden (latent) variables; see, e.g., (Elidan, Ninio, Friedman, & Shuurmans, 2002; Elidan & Friedman, 2005). More recently, there have been characterizations of EM that also have interesting connections to loopy belief propagation, and related algorithms (Liu & Ihler, 2011; Jiang, Rai, & III, 2011), although their focus is more on approximate (partial) MAP inference in probabilistic graphical models via message-passing, and less on learning.

Slow convergence of EM has also been recognized, particularly in the presence of hidden variables. In such cases, EM can be coupled with algorithms such as gradient ascent, or other more traditional algorithms for optimization; see, e.g. (Aitkin & Aitkin, 1996). A variety of other techniques for accelerating EM have been proposed in the literature; see, e.g., (Thiesson, Meek, & Heckerman, 2001).

Acknowledgements

This work has been partially supported by ONR grant #N00014-12-1-0423, NSF grant #IIS-1118122, and NSF grant #IIS-0916161. This work is also based upon research performed in collaborative facilities renovated with funds from NSF grant #0963183, an award funded under the American Recovery and Reinvestment Act of 2009 (ARRA).

References

Aitkin, M., & Aitkin, I. (1996). A hybrid EM/Gauss-Newton algorithm for maximum likelihood in mixture distributions. *Statistics and Computing*, *6*, 127–130.

Chan, H., & Darwiche, A. (2005). On the revision of probabilistic beliefs using uncertain evidence. *Artificial Intelligence*, *163*, 67–90.

Choi, A., & Darwiche, A. (2006). An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *AAAI*, pp. 1107–1114.

Choi, A., Refaat, K. S., & Darwiche, A. (2011). EDML: A method for learning parameters in Bayesian networks. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

Elidan, G., & Friedman, N. (2005). Learning hidden variable networks: The information bottleneck approach. *JMLR*, *6*, 81–127.

Elidan, G., Ninio, M., Friedman, N., & Shuurmans, D. (2002). Data perturbation for escaping local maxima in learning. In *AAAI/IAAI*, pp. 132–139.

Heckerman, D. (1998). A tutorial on learning with Bayesian networks. In Jordan, M. I. (Ed.), *Learning in Graphical Models*, pp. 301–354. MIT Press.

Jiang, J., Rai, P., & III, H. D. (2011). Message-passing for approximate MAP inference with latent variables. In *NIPS*, pp. 1197–1205.

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Lauritzen, S. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, *19*, 191–201.

Liu, Q., & Ihler, A. T. (2011). Variational algorithms for marginal MAP. In *UAI*, pp. 453–462.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California.

Thiesson, B., Meek, C., & Heckerman, D. (2001). Accelerating EM for large databases. *Machine Learning*, *45*(3), 279–299.

Active Learning with Distributional Estimates

Jens Röder[1], **Boaz Nadler**[2], **Kevin Kunzmann**[3], **and Fred A. Hamprecht**[3]

[1]CR/AEM, Robert Bosch GmbH, Hildesheim, Germany
[2]Department of Computer Science and Applied Math, Weizmann Institute of Science, Rehovot, Israel
[3]Heidelberg Collaboratory for Image Processing (HCI), University of Heidelberg, Heidelberg, Germany

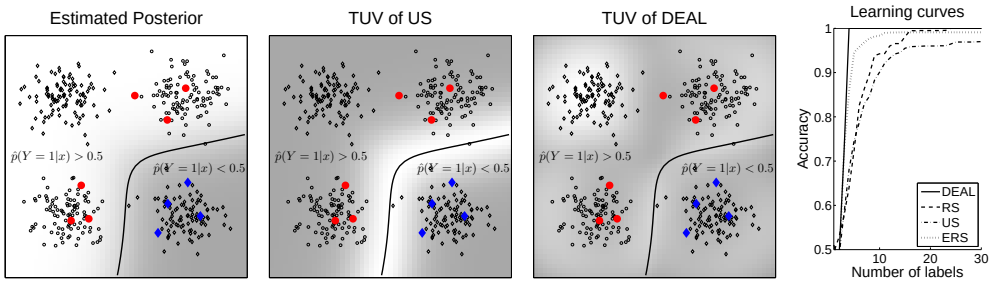
Abstract

Active Learning (AL) is increasingly important in a broad range of applications. Two main AL principles to obtain accurate classification with few labeled data are *refinement* of the current decision boundary and *exploration* of poorly sampled regions. In this paper we derive a novel AL scheme that balances these two principles in a natural way. In contrast to many AL strategies, which are based on an estimated class conditional probability $\hat{p}(y|x)$, a key component of our approach is to view this quantity as a random variable, hence explicitly considering the *uncertainty* in its estimated value. Our main contribution is a novel mathematical framework for uncertainty-based AL, and a corresponding AL scheme, where the uncertainty in $\hat{p}(y|x)$ is modeled by a second-order distribution. On the practical side, we show how to approximate such second-order distributions for kernel density classification. Finally, we find that over a large number of UCI, USPS and Caltech-4 datasets, our AL scheme achieves significantly better learning curves than popular AL methods such as uncertainty sampling and error reduction sampling, when all use the same kernel density classifier.

1 INTRODUCTION

In many applications, including computer vision and natural language processing, unlabeled data abounds while procuring labels for training is costly. Pool-based active learning (AL) schemes judiciously select those among the unlabeled points that are deemed most informative, and thought to help achieve a steeper learning curve. The prospect of reduced labeling effort has spurred intense efforts to improve AL. On the theoretical side, several works considered the sample complexity and potential benefits of AL, see (Beygelzimer et al., 2009; Balcan et al., 2010; Hanneke, 2011). On the practical side, various works suggested concrete AL schemes, recently e.g. (Huang et al., 2010; Siddiquie and Gupta, 2010), with large gains over random labeling in various applications, see (Settles, 2010) for a comprehensive review.

In this paper we focus on pool based AL. We first review a potential weakness common to many popular AL methods, and then derive a new pool-based AL scheme. In a pool-based setting, one typically starts with a small (possibly empty) set of labeled samples $\mathcal{L} = \{x_i, y_i\}_{i=1}^{\ell}$, and a large pool of unlabeled samples $\mathcal{U} = \{x_j\}_{j=\ell+1}^{n}$. Most pool-based AL schemes rely on a classifier – or more precisely, a regressor – that outputs not only a predicted class label $\hat{y}$ at a new sample x, but also an estimate $\hat{p}(y|x)$ of the conditional class probabilities $\Pr[Y = y|X = x]$ for all classes y. Then, sequential one-step looka-



(a) (b) (c) (d)

Figure 1: Active learning on the XOR problem. Small black symbols are unlabeled data, large colored symbols are the labeled ones. (a) Class-conditional probability and decision boundary estimated by kernel density classification, with labeled data in only 3 out of 4 quadrants. (b), (c) Training utility values (TUV) of uncertainty sampling and the proposed DEAL, lighter color representing a higher TUV. Uncertainty sampling prematurely concentrates on local refinement of the current decision boundary. DEAL keeps exploring before reverting to refinement. (d) Resulting learning curves.

head AL schemes compute a Training Utility Value (TUV) for any unlabeled sample, and query the label of the sample with largest TUV.

One popular and successful AL strategy is uncertainty sampling (US) which iteratively selects the sample whose current class prediction is least confident[1] (Baum, 1991; Hwang et al., 1991; Seung et al., 1992; Lewis and Gale, 1994).

Another common strategy is to query that sample whose inclusion in the training set may contribute most towards a "confident" classification. Here, confidence is measured by the entropy of the class conditional probabilities, or the expected estimated risk (MacKay, 1992; Roy and McCallum, 2001; Zhu et al., 2003): the more assertive the resulting classifier, the better, according to these algorithms. A variant in the regression context has been proposed in (Boutilier et al., 2003).

Yet another popular AL strategy is to select samples that minimize the uncertainty in the estimated parameters of a classifier, e.g. by maximally reducing the version space of a SVM (Tong and Koller, 2002) or by minimizing the variance of the parameter estimates in multinomial logistic regression (Schein and Ungar, 2007).

A common theme to all these AL schemes is their use of point estimates $\hat{p}(y|x)$, possibly combined with density estimates $\hat{p}(x)$, but without consideration of the inherent random uncertainties in these quantities. By definition, $\hat{p}(y|x)$ is estimated from the finite, and often small, currently labeled set $\mathcal{L}$.[2] Hence, at any x, $\hat{p}(y|x)$ is a random variable, which may have small bias and variance in some regions, but high uncertainty in others.

In this paper, we propose to capitalize on this seeming flaw, and to put the unavoidable uncertainty in the estimates $\hat{p}(y|x)$ at the very heart of a novel AL scheme: Distributional Estimate Active Learning (DEAL). First, in Section 2 we propose to quantify the uncertainty in the estimate $\hat{p}(y|x)$ via a *second-order distribution*, see Eq. (1). Next, in Section 3 we show how such a second-order distribution can be approximated for kernel density classification; and in Section 4 we show how such distributions can be used, in a principled mathematical framework, for uncertainty-based AL. In Section 5 we show empirically that with a baseline implementa-

[1] Note that this can be defined in many possible ways, in particular in multi-class settings.

[2] And, in the case of semi-supervised active learning, also from the unlabeled set $\mathcal{U}$.

tion using kernel density classification, DEAL performs significantly better than two highly popular AL schemes and random sampling in a thorough benchmark on more than 40 classification problems from the UCI (Frank and Asuncion, 2010) and USPS (LeCun et al., 1990) databases, and on an image classification task using the Caltech-4 dataset.

Our approach is somewhat related to the minimization of uncertainty in Gaussian process regression for space-filling experimental design (Sacks et al., 1989). In the machine learning community, several works devised efficient approximations for the intractable posterior distribution in Gaussian process classification models (Nickisch and Rasmussen, 2008). In particular, these distributions were used to compute Bayesian predictive distributions (Snelson and Ghahramani, 2005), which for classification are *point estimates* of the class conditional probabilities. Gaussian processes were also used for active learning, though there the authors suggested to label those samples whose normalized margin is smallest (Kapoor et al., 2007). Thus, even though second-order distributions were derived for logistic regression and Gaussian processes, to the best of our knowledge, these have not been used in AL for classification. In this paper we thus emphasize the importance, use, and potential benefit of second-order distributions in AL classification problems. As discussed in Section 6, second-order distributions may see potential use beyond AL.

2 CLASSIFIER UNCERTAINTY AND ACTIVE LEARNING

In statistical pattern recognition, agreement prevails that a classifier should not be forced to make a prediction unless reasonably confident about it. This principle is formalized by introducing an auxiliary "doubt" class that the classifier can always vote for, at a fixed cost (Ripley, 2008). In the generic case of a symmetric loss function, minimizing the expected risk leads to an algorithm that, given a sample x, votes for the class with highest conditional probability, $\hat{y} = \arg\max_y \hat{p}(y|x)$, provided that the expected loss of this decision is smaller than the fixed cost of the "doubt" class.

The "doubt" class captures the uncertainty of a prediction $\hat{y}$ at locations x where no class is clearly dominant. Even if the class conditional probabilities are perfectly known, this type of "first-order" uncertainty is still present wherever two classes overlap in feature space. As a direct consequence to AL, if the current labeled set makes it quite clear that two classes are equally probable at some region in feature space, it is futile to attempt reducing this first-order uncertainty by requesting more labels there!

In practice, $p(y|x)$ is unknown, and thus estimated from a finite training set. This induces a *second* kind of uncertainty: not only how confident are we in the predicted label $\hat{y}$, but also how accurate is our point estimate $\hat{p}(y|x)$. An inaccurate point estimate may result in a misleading classifier that errs and votes for the wrong class, with a class conditional probability margin that is deceptively large. Asking for the label of additional training samples in such regions can result in a decisive change in the current decision boundary. Hence, samples with highly uncertain class conditional probability estimates should be prime candidates of a good AL criterion. A point in case is the classical XOR problem, illustrated in Fig. 1. Starting with 10 labeled data in only 3 out of 4 quadrants (an event whose probability is $\sim$20% with 10 randomly selected labeled data), nearest-neighbor type classifiers give an erroneous prediction at the remaining quadrant, with a deceptively large margin. Consequently, AL schemes based on $\hat{p}(y|x)$ do not sample points in the remaining quadrant. This overconfidence of AL schemes was also noted by (Baram et al., 2004), who suggest to label at random once in a while.

Motivated by the above insights, in this paper we derive an AL scheme that incorporates this randomness in $\hat{p}(y|x)$ in a natural way. The key ingredient in our scheme is a *second-order distribution*

$$G_x(q) = \Pr[\hat{p}(y = 1|x) \leq q] \qquad (1)$$

which measures our uncertainty in the point estimate $\hat{p}(y|x)$. Before deriving the DEAL scheme, we first show how such a second-order distribution can be estimated for the kernel density classifier.

3 SECOND-ORDER DISTRIBUTIONS FOR THE KERNEL DENSITY CLASSIFIER

Kernel density classification is a prototypical non-parametric generative classifier. While with limited training data this classifier will likely have a lower accuracy compared to modern discriminative classifiers, we choose it since it is: (i) conceptually simple and easy to implement, (ii) usable in all the active learning criteria that we wish to benchmark and (iii) representative of an entire class of more advanced methods. While beyond the scope of this paper, second-order distributions can also be derived for discriminative classifiers, and then used in our AL scheme.

For simplicity, in the rest of this paper we focus on the binary classification problem, with class labels $y \in \{-1, +1\}$. To derive second-order distributions for the unknown class probabilities $p(1|x) = \Pr[Y=1|X=x]$, we use Bayes rule

$$p(1|x) = \frac{p(x|1)\pi_1}{p(x|-1)\pi_{-1} + p(x|1)\pi_1} \qquad (2)$$

with π_y the prior probability for class y. In kernel density classification, the unknown class densities $p(x|1)$ and $p(x|-1)$ are replaced by their Parzen window estimates. To derive second-order distributions, we thus need to approximate the distribution of these point estimates.

Let $\mathcal{K}$ be a normalized ($\int \mathcal{K}(u)du = 1$) isotropic kernel. Then the kernel density estimate

$$\hat{p}(x|Y=y) = \frac{1}{n_y} \sum_{x_i:y_i=y} \mathcal{K}(x - x_i) \qquad (3)$$

is a random variable. With only a single observation from class y ($n_y = 1$), the exact distribution of this random variable is given by

$$\Pr[\hat{p}(x|y) \le z] = \int \mathbf{1}\left(\mathcal{K}(u-x) \le z\right) p(u|y)du \qquad (4)$$

This distribution depends on the location of the query x, on the kernel function $\mathcal{K}$ and on the unknown density $p(x|Y=y)$. Qualitatively, for non-negative and monotonically decaying kernels $\mathcal{K}$, the resulting density must be zero for $z < 0$ and for $z > \mathcal{K}(0)$. For n_y i.i.d. observations from class y, the distribution of the class density kernel estimate is given by a n_y-fold convolution of the probability density that corresponds to Eq. (4).

Since the exact density $p(x|Y = y)$ is unknown (otherwise no learning would be necessary), we resort to an approximation of the distribution of $\hat{p}(x|y)$. Key requirements are that the approximate distribution be continuous, infinitely divisible, have no mass at $z < 0$ and its derivative should decay to zero as $z \to \infty$. A good candidate meeting these criteria is the Gamma distribution which, with its shape and location parameters k and θ, is also sufficiently rich to faithfully model a variety of situations that arise in practice. A standard estimate of mean and variance of the kernel density estimate (Härdle et al., 2004, chap. 3) allows to apply the method of moments and obtain the shape parameter $k_y = n_y\hat{p}(x|Y=y)/C_2$ and location parameter $\theta_y = C_2/n_y$, where $C_2 = \int \mathcal{K}^2(x)dx$.

When the class prior π_y is estimated by the ratio n_y/n, we obtain the following approximate distribution for the random variable $\hat{p}(x|y)$

$$\hat{p}(x|y)\hat{\pi}_y \sim \Gamma\left(\delta + \tfrac{n_y\hat{p}(x|y)}{C_2}, \tfrac{C_2}{n_y} \cdot \tfrac{n_y}{n}\right). \qquad (5)$$

Here, δ is a small positive constant added both to regularize Eq. (2) in low-density regions, and to guarantee that the shape parameter of the Gamma distribution is strictly positive, even when no labeled observations are available yet for class y.

With $\theta := C_2/n$, inserting Eq. (5) into Eq. (2) gives

$$\begin{aligned}\hat{p}(Y=1|x) &\sim \frac{\Gamma(\delta + k_1, \theta)}{\Gamma(\delta + k_{-1}, \theta) + \Gamma(\delta + k_1, \theta)} \\ &= Beta(\delta + k_1, \delta + k_{-1}) = Beta(\alpha, \beta) \qquad (6)\end{aligned}$$

In particular, for a d-dimensional isotropic Gaussian kernel with bandwidth h, we obtain

$$k_y = 2^{d/2} \sum_{x_i:y_i=y} \exp\left(-\left\|x - x_i\right\|^2 / 2h^2\right). \qquad (7)$$

4 DISTRIBUTIONAL ESTIMATE ACTIVE LEARNING (DEAL)

Our novel AL scheme requires a method (for instance the one described in the previous section, or a Gaussian process classifier) that outputs a second-order distribution $G_x(q)$, Eq. (1).

Our point of departure in deriving our AL scheme, is the following key observation (Friedman, 1997): The performance of a classifier, as measured by its misclassification error, depends only on the location of its decision boundary, and not on the precise estimates of the conditional class probabilities. In particular, inaccurate point estimates $\hat{p}(y|x)$ may still yield the optimal Bayes classifier as long as they result in the same decision boundary.

A second-order distribution can thus help assess the uncertainty in the currently estimated decision boundary. In more detail, given a second-order distribution with density $g_x(q)=\frac{d}{dq}G_x(q)$, we can extract a point estimate for the posterior probability

$$\hat{p}(1|x) = \int_0^1 q\, g_x(q) dq \tag{8}$$

and a corresponding classifier, which for a symmetric loss function is simply

$$\hat{f}(\hat{p}(1|x)) = \text{sgn}(\hat{p}(1|x) - 1/2). \tag{9}$$

The goal of classification is to build a classifier $\hat{f}$ that minimizes the overall risk

$$R = \int R[x, \hat{f}] p(x) dx \tag{10}$$

where the local risk at x is

$$R[x, \hat{f}] = \mathbb{E}_Y[L(y, \hat{f})] = \sum_{y=\pm 1} L(y, \hat{f}) p(Y = y|x). \tag{11}$$

In general, the exact local risk at x is unknown, as we do not know the exact posterior probabilities $p(y|x)$. Replacing these by their estimates gives

$$\widehat{R}[x, \hat{f}(\hat{p}(1|x))] = \sum_{y \pm 1} L(y, \hat{f}(\hat{p}(1|x)))\, \hat{p}(y|x). \tag{12}$$

Note that this formula does not take into account the inherent uncertainty in the estimate $\hat{p}(y|x)$. For example, a second-order distribution $G_x(q)$ with some spread and expectation of 1/2 implies that $p(1|x)$ *could* be much different from 1/2! In such cases, Eq. (12) is hence overly pessimistic.

A second-order distribution mitigates the overpessimism in such regions. A more balanced estimate of the risk that takes into account a second-order distribution is

$$\widehat{ER}[x] = \mathbb{E}_q[\widehat{R}[x, \hat{f}(q)]] = \int_0^1 \widehat{R}[x, \hat{f}(q)]\, g_x(q) dq \tag{13}$$

The intuition behind this estimate is as follows: if the second-order distribution has significant mass near both limits of its domain (i.e., it has a high density for values of $q = \hat{p}(y|x)$ close to 0 and 1), then it may be possible to construct a classifier with low risk at x, by querying additional labels in its neighborhood. As an extreme example, consider a second-order distribution for $\hat{p}(Y=1|x)$ given by Bernoulli(0.5), which implies that the conditional probability of class +1 is either 0 or 1. Then, $\widehat{R}[x] = 0.5$, but $\widehat{ER}[x] = 0$. This fact is taken into account by Eq. (13) but not by Eq. (12).

It is easy to prove that for any density $g_x(q)$, from which $\hat{p}(1|x)$ and $\hat{f}$ are derived via Eqs. (8) and (9), $\widehat{R}[x] \geq \widehat{ER}[x]$. Moreover, equality holds iff the entire mass of the second-order distribution lies on one side of the decision threshold, or if it is a Dirac distribution at 1/2. Interestingly, in these two cases it is of no benefit to query the label at x.

These properties suggest that the difference $\hat{R}[x] - \widehat{ER}[x]$ is a good indicator for the potential importance of acquiring a label at x, though other choices seem possible. Consequently, taking also into account that the *overall* risk is a density-weighted mean of the local risk (see Eq. (10)), we propose the following training utility value (TUV):

$$TUV(x) = \left(\widehat{R}[x] - \widehat{ER}[x]\right) \cdot \hat{p}(x) \tag{14}$$

where $\hat{p}(x)$ is some (non-parametric) estimate of the density at x. The weighting by the total density concentrates the learning effort on those regions

Table 1: One iteration of DEAL

Algorithm DEAL
Input: Labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$.
Output: Selected sample $x \in \mathcal{U}$ and its label $y(x)$
Algorithm:
1: compute density estimate $\hat{p}(x)$
2: for all $x \in \mathcal{U}$ do
- compute second-order distribution of $\hat{p}(Y=1\|x)$ by Eq. (6)
- compute $TUV(x)$ by Eq. (14)
3: query label y of $x \in \mathcal{U}$ with largest TUV

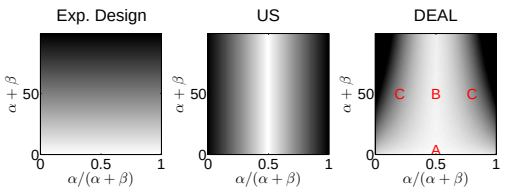


Figure 2: Training utility values, as a function of the two parameters $\alpha/(\alpha+\beta)$ and $\alpha+\beta$ in a second-order distribution of type $Beta(\alpha,\beta)$, for space-filling experimental design (Sacks et al., 1989), uncertainty sampling and DEAL. Roughly, α and β measure the local amount of evidence for either class (Eqs. (6),(7)). Not taking sample density into account, the most interesting points for DEAL are those with little evidence for either class as yet (A), followed by points with evidence for both classes (B), followed by points with strong evidence for one and little for the other class (C). In contrast, US merely takes into account the distance from the decision boundary, pretending (A)≡(B). Space-filling experimental design prefers unexplored regions, regardless of their estimated class conditional probabilities, so that (B)≡(C).

of feature space that are actually relevant. Table 1 describes the pseudo-code of a single iteration of DEAL. Of course, the density estimate $\hat{p}(x)$ need be computed only once at the start of the AL process. Fig. 2 compares the TUV of DEAL to criteria used in experimental design and for US.

5 RESULTS

We compare the empirical performance of DEAL to that of random sampling (RS), uncertainty sampling (US) (Lewis and Gale, 1994) and error reduction sampling (ERS) (Roy and McCallum, 2001). For a meaningful comparison, all methods use the same kernel density classifier, with an isotropic Gaussian kernel whose bandwidth is chosen according to the normal reference rule (Scott, 1992, chapter 6). The density $\hat{p}(x)$ in Eq. (14) is also estimated by kernel density estimation with the same kernel and bandwidth selection rule.

As is well known, non-parametric kernel density estimation with a limited number of samples may be highly inaccurate in high dimensions (Scott, 1992, chapter 7). Therefore, we first project the data to its d leading principal components, where the dimension d is chosen according to the resampling via permutation scheme of (Zhu and Ghodsi, 2006), with the minimum number of components set to two.

We always start with an empty set $\mathcal{L}$ of labeled points. In case of RS, US and ERS, the first query points are selected randomly until there is at least one label for each class. In case of DEAL, its deterministic strategy can be applied from the very beginning, with the first label requested for the point with the highest density estimate. In all our experiments, we set $\delta = 0.5$, consistent with Jeffreys' prior for the Bernoulli distribution (Jeffreys, 1946).

5.1 UCI DATA SETS BENCHMARK

We considered 32 of the most frequently used UCI data sets[3]. Each dataset was preprocessed as follows: (i) Categorical variables with more than two outcomes were replaced by #outcomes-1 indicator variables, (ii) missing values in categorical variables were treated as a separate outcome, (iii) missing values in continuous inputs were replaced by the respective mean, and (iv) the data was normalized to unit variance in each dimension. If a dataset

[3]We excluded datasets with only categorical variables, or with significant missing data. We did not exclude datasets on which our AL scheme did not perform well.

had more than two classes, the classes were joined to create binary problems such that the new classes were approximately equally abundant.

All results are obtained from 10-fold cross validation (CV), i.e., nine tenths of the data were used in active learning, with one tenth reserved for testing. To average out the randomness of the initial labeling for RS, US and ERS, all experiments are repeated 5 times for each of the 10 CV folds. Fig. 3(a) shows

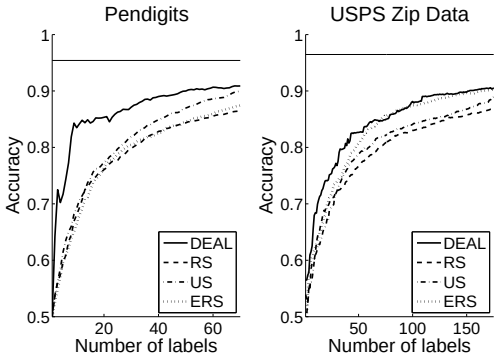


Figure 3: Learning curves for two data sets ("Pendigits" and "USPS, Grouping 10") where DEAL works well. Note that DEAL does not outperform the competing methods on every single data set, but on average across multiple data sets (see tables). The top horizontal line is the asymptotic performance of the classifier, at the end of the complete learning curve (estimated by 10-fold CV for UCI and on separate test set for the USPS data).

the learning curve for the Pendigits dataset. All others appear in the supplementary material.

To reward both initial steepness of the learning curve and early convergence to a high accuracy, we propose to measure performance by the area under the learning curve. All curves are truncated when the worst method achieves 90% of the accuracy of the classifier trained with the completely labeled training data[4], but at the latest after 200 iterations. As we compare only the relative performance

[4]Defined as the average of the 10-fold CV accuracies, each with a different set of 9/10 of the data fully labeled.

Table 2: Average accuracy of the compared AL strategies for 32 data sets from the UCI database with preprocessing as described in text, where n is the total size of the data set and d the dimension of the PCA subspace. The mean rank is computed based on ordering the performance of the AL strategies for each data set. The best and second best methods are indicated by bold font and italics, respectively. The mean rank test statistic is used for the statistical hypothesis tests described in the text.

Dataset (n,d)	RS	US	ERS	DEAL
Anneal (898,17)	.813	*.849*	.802	**.857**
Audiology (226,9)	.664	.650	**.680**	*.666*
Autos (205,14)	*.653*	.638	.614	**.678**
Balance S. (625,2)	**.717**	.705	*.716*	.715
Breast C. (286,16)	*.644*	**.656**	.640	.617
Breast W (699,2)	.807	*.835*	.820	**.855**
Dermatol. (366,4)	.802	*.841*	.789	**.878**
Diabetes (768,2)	.684	.682	*.687*	**.695**
Ecoli (336,3)	*.796*	.793	.793	**.852**
Glass (214,4)	.646	**.688**	*.669*	.668
Heart C (303,8)	.733	.722	*.748*	**.753**
Hepatitis (155,7)	.782	*.796*	.781	**.801**
Hyperth. (3772,11)	.863	*.889*	.865	**.919**
Ionosphere (351,5)	.782	.802	*.817*	**.841**
Iris (150,2)	.793	*.809*	.807	**.924**
Led 24 (1000,2)	*.667*	.643	.667	**.695**
Letter (20000,5)	.631	.627	*.632*	**.652**
Liver (345,2)	.530	**.541**	.516	*.539*
Lymph (148,9)	.671	**.712**	.681	*.692*
Optdig. (5620,18)	.819	*.849*	.791	**.887**
Pendigits (7494,5)	.783	*.804*	.783	**.861**
Primary Tu (339,9)	*.652*	.650	.647	**.696**
Satimage (6435,3)	*.777*	*.819*	.790	**.852**
Segment (2310,3)	*.830*	.741	.737	**.871**
Sonar (208,8)	.695	*.714*	.699	**.725**
Soybean (683,20)	.786	*.811*	.764	**.831**
Vehicle (846,4)	.720	**.736**	.721	*.734*
Vote (435,8)	.803	.799	*.812*	**.841**
Vowel (990,16)	*.671*	.540	.627	**.694**
Waveform (5000,2)	.767	**.793**	.787	*.787*
Wine (178,3)	.831	.847	*.856*	**.895**
Yeast (1484,2)	.571	.562	*.577*	**.592**
Mean Rank	3.09	2.56	2.97	1.38

of different AL strategies for the same classification algorithm, this measure is equivalent to the one proposed in (Baram et al., 2004) and also used in (Schein and Ungar, 2007). The results for all data

Table 3: Average accuracy of the compared AL strategies for 10 different groupings of the USPS Zip Data with preprocessing as described in text. The best and second best method are indicated using bold font and italics, respectively. The mean rank is computed based on ordering the performance of the AL strategies for each grouping.

Grouping	RS	US	ERS	DEAL
$\{1,2,3,4,5\}$	0.777	0.807	*0.829*	**0.832**
$\{0,1,2,3,4\}$	0.786	0.808	*0.831*	**0.837**
$\{1,3,5,7,9\}$	0.782	0.819	**0.832**	*0.830*
$\{0,1,7,8,9\}$	0.774	0.811	*0.817*	**0.830**
$\{1,3,4,5,9\}$	0.793	0.810	*0.828*	**0.838**
$\{1,2,3,7,8\}$	0.782	0.797	*0.825*	**0.833**
$\{0,1,6,8,9\}$	0.777	0.813	*0.824*	**0.846**
$\{0,5,6,7,9\}$	0.777	0.805	*0.815*	**0.830**
$\{0,2,4,5,8\}$	0.750	0.805	*0.815*	**0.821**
$\{3,4,5,6,9\}$	0.791	0.799	*0.825*	**0.840**
Mean Rank	4.000	3.000	1.900	1.100

sets are presented in Table 2.

We compare the performance of the different strategies as recommended in (Demšar, 2006). The Friedman test, which uses the mean performance ranks of Table 2, yields a p-value of $p = 2.53 \times 10^{-9}$ for the null hypothesis of equal performance of all strategies. For comparing all classifiers to each other, we use the two-tailed Nemenyi test. At a 1% significance level its threshold for differences in Mean Rank is 1.004. This means that DEAL performs significantly better than each of the other strategies. The performances of the other methods do not differ significantly from each other, even at the 10% level (corresponding to a threshold of 0.739).

5.2 USPS ZIP DATA

To obtain challenging classification tasks with convoluted decision boundaries, the digit images from the USPS corpus (LeCun et al., 1990) were grouped into two classes in various ways, see Table 3. All images were projected to the $d = 39$ leading principal components, with 7291 samples eligible for active learning and an independent set of 2007 samples held out for testing purposes.

Table 4: Average accuracy of the compared AL strategies for 3 different groupings of the Caltech-4 data set with preprocessing as described in text. The best and second best method are indicated using bold font and italics, respectively.

Grouping	RS	US	ERS	DEAL
$\{1,2\}$ vs. $\{3,4\}$	0.818	*0.846*	0.807	**0.877**
$\{1,3\}$ vs. $\{2,4\}$	0.799	*0.829*	0.803	**0.840**
$\{1,4\}$ vs. $\{2,3\}$	0.803	*0.836*	0.797	**0.872**
Mean Rank	3.333	2.000	3.667	1.000

As Table 3 shows, DEAL performed best in 9 out of the 10 groupings. Fig. 3(b) shows one learning curve, all others are in the supplementary material.

5.3 CALTECH-4

Caltech-4 is a well established standard benchmark for object categorization (Fergus et al., 2003) and has also been used in AL (Kapoor et al., 2007). This dataset consists of 4 different image groups: airplanes (category 1; 800 images), rear views of cars (2; 1155), frontal faces (3; 435) and motorbikes (4; 798). Fig. 4 shows one example from each category. We represent the images by the "Color and Edge Directivity Descriptor" (CEDD) (Chatzichristofis and Boutalis, 2008). The resulting 144-dimensional features were then projected to the 17 leading principal components. To create challenging two-class problems with convoluted decision boundaries, we grouped the 4 categories in three possible ways.

Table 4 presents the resulting performances, based on 10-fold CV with 5 repetitions (see Section 5.1). It shows that DEAL performs best for all groupings. Moreover, for this dataset, US is the second best strategy, probably because the problem is not as challenging as the Zip Data, that originally consists of 10 categories. Interestingly, ERS performs worse than random sampling in two out of three tasks.

6 DISCUSSION

In this paper we derived a new AL strategy, which considers not only the density and distance of an

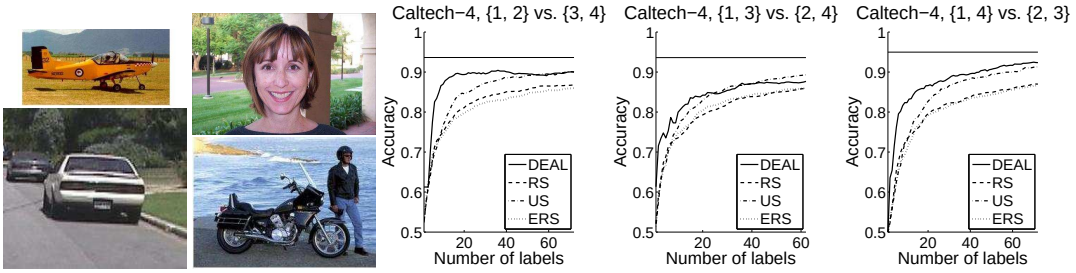


Figure 4: Left: Example images of the 4 object categories of Caltech-4 (airplane, car, face, motorbike). Right: Learning curves for three possible groupings of the 4 categories. DEAL performed best in all cases and US second best. Interestingly, ERS is not better than RS here (see also Table 4). The top horizontal line is the asymptotic accuracy of the classifier, with all training data labeled (estimated by 10-fold CV).

unlabeled sample to the decision boundary, but also the number of labeled points in its neighborhood. All this information is taken into account by requiring that the underlying classifier provide a distributional estimate at each unlabeled point, leading to a natural definition of its training utility value.

Information similar to that contained in a second-order distribution is *implicitly* used by methods that minimize the expected estimated risk (MacKay, 1992; Roy and McCallum, 2001; Zhu et al., 2003). These AL schemes indirectly measure the uncertainty of a point estimate, by perturbing the current classifier with hypothetical new labels and investing where the potential reduction in estimated risk is greatest.

In contrast, DEAL makes this dependence on the uncertainty explicit. Not only is it simple to implement, it also empirically outperformed error reduction sampling, uncertainty and random sampling schemes on a large collection of UCI, USPS and Caltech data sets. Note that one cannot expect a single strategy to perform best on all data sets. For instance, if the decision boundary is simple, strongly favoring exploitation over exploration (as in uncertainty sampling) may be the best strategy. For more challenging classification problems with complex class boundaries, balancing exploration and refinement, as DEAL does, seems a crucial ingredient for active learning.

While our AL scheme is general and applicable to any classifier that outputs second-order distributions, in this paper we focused for simplicity on its implementation with kernel density classification. As we shall describe in a future publication, second-order distributions can be derived for other classifiers, most notably random forest (Breiman, 2001). Encouragingly, with random forest as the base classifier, not only are lower classification errors achieved, but also the advantage of DEAL over the other AL strategies continues to hold.

Finally, we note that second-order distributions are not limited to AL. In the presence of few training data, they may be used to extend the "doubt" class to also include poorly explored regions with high uncertainty. While beyond the scope of this paper, second-order distributions are also useful for outlier detection, in applications such as optical inspection, where not all defects are known in advance when training the classifier. These extensions, as well as generalizing our AL scheme to multi-class problems, and deriving second-order distributions for other (discriminative) classifiers, are interesting topics for future research.

Acknowledgements.

BN was supported by a grant from the Israeli Science Foundation (ISF).

References

M. Balcan, S. Hanneke, and J. Wortman. The true sample complexity of active learning. *Machine Learning*, 80: 111–139, 2010.

Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

E. Baum. Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks*, 2(1):5–19, 1991.

A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *ICML*, pages 49–56, 2009.

C. Boutilier, R. S. Zemel, and B. Marlin. Active collaborative filtering. In *Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 98–106, 2003.

L. Breiman. Random forests. *Machine Learning*, 45(1): 5–32, 2001.

S. Chatzichristofis and Y. Boutalis. CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval. *Computer Vision Systems*, pages 312–322, 2008.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, volume 2, pages 264–271, 2003.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

J. H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

S. Hanneke. Rates of convergence in active learning. *Annals of Statistics*, 39(1):333–361, 2011.

W. Härdle, M. Müller, S. Sperlich, and A. Werwatz. *Nonparametric and Semiparametric Models*. Springer, 2004.

S.-J. Huang, R. Jin, and Z.-H. Zhou. Active learning by querying informative and representative examples. In *NIPS*, pages 892–900, 2010.

J. Hwang, J. Choi, S. Oh, R. Marks, et al. Query-based learning applied to partially trained multilayer perceptrons. *IEEE Transactions on Neural Networks*, 2(1): 131–136, 1991.

H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007):453–461, 1946.

A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *ICCV*, 2007.

Y. LeCun, O. Matan, B. Boser, J. S. Denker, et al. Handwritten zip code recognition with multilayer networks. In *ICPR*, volume II, pages 35–40, 1990.

D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR Conf on Research and Development in Information Retrieval*, pages 3–12, 1994.

D. J. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4:720–736, 1992.

H. Nickisch and C. E. Rasmussen. Approximations for binary gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078, 2008.

B. Ripley. *Pattern recognition and neural networks*. Cambridge Univ Press, 2008.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, pages 441–448, 2001.

J. Sacks, W. Welch, T. Mitchell, and H. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

A. Schein and L. Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68:235–265, 2007.

D. W. Scott. *Multivariate Density Estimation*. John Wiley, 1992.

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2010.

H. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *5th Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.

B. Siddiquie and A. Gupta. Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *CVPR*, pages 2979–2986, 2010.

E. Snelson and Z. Ghahramani. Compact approximations to bayesian predictive distributions. In *ICML*, pages 840–847, 2005.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2002.

M. Zhu and A. Ghodsi. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis*, 51: 918–930, 2006.

X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.

Integrated Nested Laplace Approximation for Bayesian Nonparametric Phylodynamics

Julia A. Palacios
Department of Statistics
University of Washington
jpalacio@uw.edu

Vladimir N. Minin
Department of Statistics
University of Washington
vminin@uw.edu

Abstract

The goal of phylodynamics, an area on the intersection of phylogenetics and population genetics, is to reconstruct population size dynamics from genetic data. Recently, a series of nonparametric Bayesian methods have been proposed for such demographic reconstructions. These methods rely on prior specifications based on Gaussian processes and proceed by approximating the posterior distribution of population size trajectories via Markov chain Monte Carlo (MCMC) methods. In this paper, we adapt an integrated nested Laplace approximation (INLA), a recently proposed approximate Bayesian inference for latent Gaussian models, to the estimation of population size trajectories. We show that when a genealogy of sampled individuals can be reliably estimated from genetic data, INLA enjoys high accuracy and can replace MCMC entirely. We demonstrate significant computational efficiency over the state-of-the-art MCMC methods. We illustrate INLA-based population size inference using simulations and genealogies of hepatitis C and human influenza viruses.

1 INTRODUCTION

Estimation of population size dynamics from molecular data is a fundamental task in ecology and public health. Since population size fluctuations affect the variability of population gene frequencies, current molecular sequence data provide information about the past population size trajectory. Such indirect inference is particularly useful in retrospective studies, where assessing past population sizes via sampling or fossil records is impossible. For example, RNA samples of hepatitis C virus (HCV) obtained in 1993 were sufficient to estimate the dynamics of HCV infections in Egypt from 1895 to 1993 (Pybus et al., 2003); and ancient and modern musk ox mitochondrial DNA samples, dated from 56,900 radiocarbon years old to contemporaneous, allowed for estimation of musk ox population dynamics throughout the late Pleistocene to the present (Campos et al., 2010).

Molecular sequence data of individuals sampled at a single time point (*isochronous* sampling) or at different points in time (*heterochronous* sampling) are related to each other via, a usually unknown, genealogical relationship. A genealogy is a rooted bifurcating tree that describes the ancestral relationships of the sampled individuals (left upper box in Figure 1). In the genealogy, each internal node indicates that the two lineages met a common ancestor. Such events are called *coalescent events*, and these events occur at *coalescent times*.

Kingman's coalescent (Kingman, 1982) is a probability model that describes a stochastic process of generating a genealogy of a random sample of molecular sequences given the effective population size (Nordborg, 2001; Hein et al., 2005). The original formulation, that considered only a constant population size, was later generalized to a variable population size (Slatkin and Hudson, 1991; Griffiths and Tavaré, 1994). Statistically, the coalescent model was an important advance, because it allowed for likelihood-based inference of population dynamics.

Many coalescent-based methods for estimation of effective population size trajectories have been developed over the last 10 years. For a recent review see (Ho and Shapiro, 2011). Some methods assume that a fixed genealogy is available (Fu, 1994; Pybus et al., 2000) and others may or may not consider the genealogical uncertainty and can produce estimates of population size trajectories from a fixed genealogy or directly from molecular data (Kuhner et al., 1995; Drummond et al., 2002, 2005; Minin et al., 2008). Felsenstein (1992) showed that likelihood-based methods that ac-

count for genealogical uncertainty are statistically the most efficient. However, all methods that incorporate genealogical uncertainty in population size dynamics reconstruction integrate over the space of genealogies using Markov chain Monte Carlo (MCMC). Such MCMC sampling of genealogies is computationally expensive. Sometimes, a single genealogy estimated from sequences that contain sufficient phylogenetic information is enough to estimate population trajectories accurately (Pybus et al., 2000; Minin et al., 2008). In this paper, we are interested in providing a fast estimation of population size trajectories from a fixed genealogy.

Some coalescent-based methods assume a simple parametric form of the population size trajectory (e.g., exponential or logistic growth), allowing the model parameters to be estimated by maximum likelihood or Bayesian methods. However, more flexible nonparametric methods are preferable for populations with poorly understood population dynamics, where it may be difficult to justify a simple parametric form of the population size trajectory. In fact, all recently developed methods rely on Bayesian nonparametric techniques to perform inference (Opgen-Rhein et al., 2005; Drummond et al., 2005; Heled and Drummond, 2008; Minin et al., 2008; Palacios and Minin, 2011). A common characteristic of most of these methods is the assumption of a piece-wise linear trajectory of effective population sizes and the possibility of the number of parameters growing with the number of samples. Bayesian skyline methods (Drummond et al., 2005; Heled and Drummond, 2008) and Opgen-Rhein et al. (2005) use multiple change point models to estimate population trajectories in a Bayesian framework. The method of Opgen-Rhein et al. (2005) is implemented only for a fixed genealogy. Recently, Bayesian nonparametric approaches that rely on Gaussian processes have been successfully implemented (Minin et al., 2008; Palacios and Minin, 2011). These methods model the effective population size as a function of a Gaussian process (GP) *a priori*, providing more flexible priors than previous Bayesian nonparametric methods.

GP-based models use MCMC methods to perform Bayesian inference. We show that when the genealogy remains fixed, these models fall into a general class of latent Gaussian models, for which integrated nested Laplace approximation (INLA) can be used to perform computationally efficient approximate Bayesian inference (Rue et al., 2009; Illian et al., 2012). Here, we adapt the INLA methodology to the estimation of population size trajectories and replace MCMC entirely. Our approximation is accurate and much faster than MCMC, while still providing the benefits of the Gaussian process-based Bayesian nonparametric approach. We illustrate the performance of our method with simulated and two real data sets.

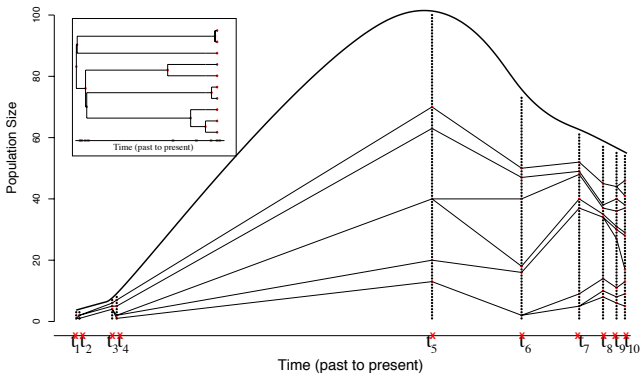


Figure 1: Example of a genealogy of 10 individuals randomly sampled at time t_{10} (red circles) from the population depicted as black circles. When we follow the ancestry of the samples back in time, two of those lineages coalesce at time t_9, the rest of the lineages continue to coalesce until the time to the most recent common ancestor of the sample at time t_1. The population size trajectory is shown as the solid black curve. When the population size is large (around t_5), for any pair of lineages that exist at time t_5 (red circles at t_5), it takes longer to meet their most recent common ancestor (t_4). The upper left box shows the genealogy reconstructed by following the ancestry of the 10 sampled individuals. The genealogy in the upper left corner is the aligned representation of the genealogy depicted in the main plot.

2 COALESCENT BACKGROUND

We assume that a genealogy with time measured in units of generations is available. Let $t_n = 0$ denote the present time when all n available sequences are sampled (*isochronous*) and let $t_n = 0 < t_{n-1} < ... < t_1$ denote the coalescent times of lineages in the genealogy. Figure 1 depicts an example of such a genealogy with time going backwards, that is, the first coalescent time occurred t_{n-1} generations ago and all the samples meet the common ancestor t_1 generations ago. Let $N_e(t)$ denote the time evolution of the effective population size as we move into the past. Then, the conditional density of the coalescent time t_{k-1}, given the previous coalescent time t_k, takes the following form:

$$P[t_{k-1}|t_k, N_e(t)] = \frac{C_k \exp\left[-\int_{t_k}^{t_{k-1}} \frac{C_k}{N_e(t)} dt\right]}{N_e(t_{k-1})}, \quad (1)$$

where $C_k = \binom{k}{2}$ is the coalescent factor that depends on the number of lineages $k = 2, ..., n$, meaning that the density for the next coalescent time is quadratic in the number of lineages and inversely proportional to the effective population size. The larger the popu-

lation size, the more genetic variability is in the population and hence, the longer it takes for two lineages to coalesce. The larger the number of lineages, the faster two of them meet their common ancestor. Figure 1 shows an example of a population that experiences growth and then a decay in population size. In this case, no pair of lineages coalesces between times t_4 and t_5, because the population is large during this time interval, while it takes little time for a pair of lineages to find their common ancestor after time t_4, when the population size becomes very small.

The *heterochronous* coalescent arises when samples of sequences are collected at different times. The conditional density of a coalescent time t_{k-1} is slightly different than Eq. 1 since it takes into account the fact that the number of lineages at each time point depends not only on the number of coalescent events (in which case, the number of lineages decreases by one each time), but also on the new samples incorporated into the analysis at any time after the last coalescent time t_k. The details of the heterochronous coalescent are omitted for brevity, however, all methods described here have been implemented to incorporate heterochronous sampling. See (Felsenstein and Rodrigo, 1999) and (Drummond et al., 2002) for a more detailed account of heterochronous sampling.

Under this coalescent-based framework, we ignore the effects of population structure, recombination and selection (Nordborg, 2001). The parameter of interest, the effective population size, can be used to approximate census population size by knowing the generation time in calendar units and the population variability in the number of offspring. The latter quantity might be difficult to know *a priori*, however, sometimes it suffices to analyze an arbitrarily rescaled population size trajectory, assuming the variability in the number of offspring remains constant.

2.1 ESTIMATION OF $N_e(t)$ USING A DISCRETE-TIME GMRF

There are two approaches to estimation of effective population size trajectories that use Gaussian processes. The first approach, developed by Minin et al. (2008), assumes *a priori* that given a genealogy, the effective population size trajectory is a piecewise constant trajectory with change points (knots) placed at coalescent times. That is,

$$N_e(t) = \sum_{k=2}^{n} \exp(\gamma_k)\, 1_{(t_k, t_{k-1}]}(t), \tag{2}$$

where

$$\boldsymbol{\gamma} = (\gamma_2, ..., \gamma_k) \sim MVN\left(0, (\tau \mathbf{Q})^{-\mathbf{1}}\right) \text{ and}$$

$$1_{(t_k, t_{k-1}]}(t) = \begin{cases} 1 & \text{if } t \in (t_k, t_{k-1}], \\ 0 & \text{otherwise.} \end{cases}$$

More specifically, *a priori* $\boldsymbol{\gamma}$ is assumed to be an intrinsic Gaussian Markov random field (GMRF) on a chain graph connecting nodes 2 through n. Minin et al. (2008) used a random walk of the first order (rw1) on an irregular grid of mid-points of inter-coalescent time intervals. For this reason, we refer to this method here as the coalescent grid Gaussian process (CGGP). The random walk construction implies that matrix $\mathbf{Q}$ is tridiagonal and positive semidefinite (hence the intrinsic GMRF). See (Rue and Held, 2005) for background on GMRFs. The precision parameter τ has a Gamma prior distribution with $\alpha = \beta = 0.001$. The authors estimate $\boldsymbol{\gamma}$ and τ by MCMC sampling from the posterior distribution of these parameters. The estimated trajectory and the corresponding uncertainty are reported in the form of pointwise posterior medians and 95% Bayesian credible intervals (BCIs) obtained from the MCMC samples.

2.2 ESTIMATION OF $N_e(t)$ USING A CONTINUOUS-TIME GP

Instead of modelling $N_e(t)$ as a piecewise continuous function *a priori*, Palacios and Minin (2011) propose a more flexible prior specification and place a transformed Gaussian process prior on $N_e(t)$. The transformation is a sigmoidal function with a lower bound. This particular transformation is required by the authors in order to perform exact posterior inference via a data augmentation scheme, which is similar to the work of Adams et al. (2009). However, a log-Gaussian transformation using a finely discretized Gaussian process, in principle, would produce similar results (Møller et al., 1998; Adams et al., 2009).

2.2.1 EXACT POSTERIOR INFERENCE WITH GP

Palacios and Minin (2011) place the following prior on $N_e(t)$:

$$N_e(t) = \left(\frac{\lambda}{1 + \exp[-\gamma(t)]}\right)^{-1}, \tag{3}$$

where

$$\gamma(t) \sim \mathcal{GP}(0, C) \tag{4}$$

and $\mathcal{GP}(0, C)$ denotes a Gaussian process with mean function 0 and covariance function C. A Gaussian process restricted to finite data is a multivariate Gaussian distribution. That is, $\gamma(t_1), ..., \gamma(t_B) \sim MVN(\mathbf{0}, \boldsymbol{\Sigma})$. *A priori*, $1/N_e(t)$ is a sigmoidal Gaussian process, a scaled logistic function of a Gaussian process which range is restricted to lie in $[0, \lambda]$; λ is a positive constant hyperparameter, inverse of which serves as a

lower bound of $N_e(t)$ (Adams et al., 2009). The likelihood function is the product of the conditional densities in Eq. 1 and involves integration of $N_e(t)$, that under the $\mathcal{GP}$ assumption, is intractable. The authors, following earlier work by Adams et al. (2009) on Poisson processes, do inference assuming an augmented data likelihood which allows to bypass intractability in the likelihood. The authors implement their method for the Brownian motion $\mathcal{GP}$ with a precision parameter τ. They place a Gamma prior distribution on the precision hyperparameter τ with $\alpha = \beta = 0.001$ and a mixture of uniform and exponential distributions on an upper bound of $1/N_e(t)$ (or equivalently, a lower bound on $N_e(t)$) as follows:

$$P(\lambda) = \epsilon \frac{1}{\hat{\lambda}} I_{\{\lambda < \hat{\lambda}\}} + (1-\epsilon)\frac{1}{\hat{\lambda}} e^{-\frac{1}{\hat{\lambda}}(\lambda - \hat{\lambda})} I_{\{\lambda \geq \hat{\lambda}\}}, \quad (5)$$

where $\epsilon > 0$ is a mixing proportion and $\hat{\lambda}$ is our best guess of the upper bound, possibly obtained from previous studies. The authors estimate τ and $N_e(t)$, or equivalently, τ, $\gamma(t)$ and λ by MCMC sampling from the posterior distribution of these parameters. The estimated trajectory and the corresponding uncertainty are reported in the form of the pointwise posterior medians and 95% BCIs evaluated at a grid of points $\{s_1, ..., s_B\}$ obtained from the MCMC samples. This grid can be made as fine as necessary after the MCMC is finished. The values of $\{\gamma(s_1), \gamma(s_2), .., .\gamma(s_B)\}$ are obtained via the $\mathcal{GP}$ predictive distribution conditioning on the values of each iteration. This method will be referred to as exact Gaussian process (EGP).

2.2.2 DISCRETIZED CONTINUOUS-TIME GP

The continuous-time version of the prior specified in Eq. 2, is

$$N_e(t) = \exp\left[\gamma(t)\right], \quad (6)$$

where $\gamma(t)$ is the Gaussian process described in Eq. 4. However, for the same reason described in section 2.2.1, the likelihood function becomes intractable. Palacios and Minin (2011) showed that estimation of the effective population size is analogous to the estimation of an inhomogeneous intensity of a point process. In this context, and under the prior described in Eq. 6, estimation of $N_e(t)$ is computationally equivalent to the estimation of the intensity function of a Log-Gaussian Cox process (Møller et al., 1998). In a Log-Gaussian Cox process, the likelihood is commonly approximated by discretization. The approximation method proceeds by constructing a fine regular grid $\{s_1, ..., s_B\}$ over the observation window and approximate

$$\int \frac{dt}{N_e(t)} = \int \exp\left[-\gamma(t)\right] dt, \quad (7)$$

by

$$\sum_{j=2}^{B} \exp\left(-\gamma_j^*\right) \Delta, \quad (8)$$

where Δ is the distance between grid points, and γ_j^* is a representative value of $\gamma(t)$ in the interval (s_{j-1}, s_j), usually $\gamma((s_j - s_{j-1})/2)$. Note that if the Gaussian process is a Brownian motion process, this approximation is similar to the CGGP method described in section 2.1. The difference is in the construction of the grid. In the CGGP method, the grid is irregular and determined by the coalescent times. For this reason, we call approximation (8) a regular grid Gaussian process (RGGP).

3 INTEGRATED NESTED LAPLACE APPROXIMATION

INLA provides fast and accurate Bayesian approximation to posterior marginals in *latent Gaussian models* (Rue et al., 2009). Latent Gaussian models are a wide class of hierarchical models in which the response variables $\mathbf{y} = (y_1, \ldots, y_n)$ are assumed to be conditionally independent given some latent parameters $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)$ and other parameters $\boldsymbol{\theta}_1$. The second hierarchical level corresponds to specifying $\boldsymbol{\eta}$ as a function of a GMRF $\mathbf{x} = (x_1, \ldots, x_n)$ with a precision matrix $\mathbf{Q}$ and hyperparameters $\boldsymbol{\theta}_2$, and the third and last hierarchical stage corresponds to prior specifications for the hyperparameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ Formally,

$$\pi(\mathbf{y}|\boldsymbol{\eta}, \boldsymbol{\theta}_1) = \prod_j \pi(y_j|\eta_j(x_j), \boldsymbol{\theta}_1), \quad (9)$$

$$\mathbf{x} \sim MVN(\mathbf{0}, \mathbf{Q}^{-1}(\boldsymbol{\theta}_2)), \quad (10)$$

and

$$\boldsymbol{\theta} \sim P(\boldsymbol{\theta}). \quad (11)$$

An interface in `R`, called `INLA`, implements a wide variety of likelihoods (Eq. 9), link functions ($\boldsymbol{\eta}$) and GMRFs (Eq. 10), including the Poisson likelihood model for each observed value of y_j (not necessarily the same for every y_j) with a logarithmic additive link function and random walk of first order as a GMRF. See `www.r-inla.org` for documentation.

The coalescent with variable population size (Eq. 1), together with the GMRF prior specification (Eq. 2) falls into the latent Gaussian model class, so INLA can be implemented for these coalescent models. In the case of the continuously specified $\mathcal{GP}$ (section 2.2), the approximate posterior method described in Section 2.2.2 (RGGP) also falls into the latent Gaussian model class.

3.1 INLA FOR PHYLODYNAMICS

Although INLA is implemented for a wide variety of latent Gaussian models, we will only describe the main steps of the approximation for posterior inference of effective population size trajectories. A typical summary of the posterior distribution of the effective population size trajectory, $N_e(t)$, is described by posterior medians and 95% BCIs evaluated pointwise on a grid of time points. These values can be obtained from the posterior marginals of the population trajectory on the grid. For the CGGP model described in section 2.1, we then wish to obtain the posterior marginals

$$\Pr(\gamma_i|\mathbf{t}) = \int_0^\infty \Pr(\gamma_i|\tau,\mathbf{t})\Pr(\tau|\mathbf{t})d\tau, i = 2,..,n \quad (12)$$

and

$$\Pr(\tau|\mathbf{t}), \quad (13)$$

where $\mathbf{t}$ denotes the vector of coalescent times. A nested procedure is used to construct approximations of $\Pr(\gamma_i|\tau,\mathbf{t})$ and $\Pr(\tau|\mathbf{t})$ first and then numerically integrate out τ to arrive at $\Pr(\gamma_i|\mathbf{t})$. The approximation of the marginal of τ is

$$\tilde{\Pr}(\tau|\mathbf{t}) \propto \left.\frac{\Pr(\boldsymbol{\gamma},\tau,\mathbf{t})}{\tilde{\Pr}_{\mathbf{G}}(\boldsymbol{\gamma}|\tau,\mathbf{t})}\right|_{\boldsymbol{\gamma}^*(\tau)}, \quad (14)$$

where $\boldsymbol{\gamma}^*(\tau)$ is the mode of the full conditional $\Pr(\boldsymbol{\gamma}|\tau,\mathbf{t})$, obtained using the Newton-Raphson algorithm, and $\tilde{\Pr}_G(\boldsymbol{\gamma}|\tau,\mathbf{t})$ is the Gaussian approximation of this full conditional constructed via a Taylor expansion around $\boldsymbol{\gamma}^*(\tau)$. The resulting $\tilde{\Pr}_G(\boldsymbol{\gamma}|\tau,\mathbf{t})$ is a Gaussian distribution with mean $\boldsymbol{\gamma}^*$ and precision matrix $\mathbf{Q}$ +diag($\mathbf{c}$), where $\mathbf{Q}$ is the prior precision matrix of the GMRF $\boldsymbol{\gamma}$ and a vector $\mathbf{c}$ consists of the second order Taylor series coefficients.

The approximation to the full conditional $\Pr(\gamma_i|\tau,\mathbf{t})$ is the following:

$$\tilde{\Pr}(\gamma_i|\tau,\mathbf{t}) \propto \left.\frac{\Pr(\boldsymbol{\gamma},\tau,\mathbf{t})}{\tilde{\Pr}_{\mathbf{G}}(\boldsymbol{\gamma}_{-i}|\tau,\mathbf{t})}\right|_{\boldsymbol{\gamma}^*_{-i}}, \quad (15)$$

where $\boldsymbol{\gamma}^*_{-i} = \mathrm{E}_G(\boldsymbol{\gamma}_{-i}|\gamma_i,\tau,\mathbf{t})$ and $\tilde{\Pr}_G(\boldsymbol{\gamma}_{-i}|\tau,\mathbf{t})$ are derived from $\tilde{\Pr}_G(\boldsymbol{\gamma}|\tau,\mathbf{t})$.

For the continuously specified GP approximation described in section 2.2.2, the INLA approximation is, in essence the same, but the GMRF is placed at the mid-points of a finer and regular grid. In this case, there are two levels of approximation, one level corresponding to the likelihood discretization and another level corresponding to the approximation of marginal posterior distributions of model parameters.

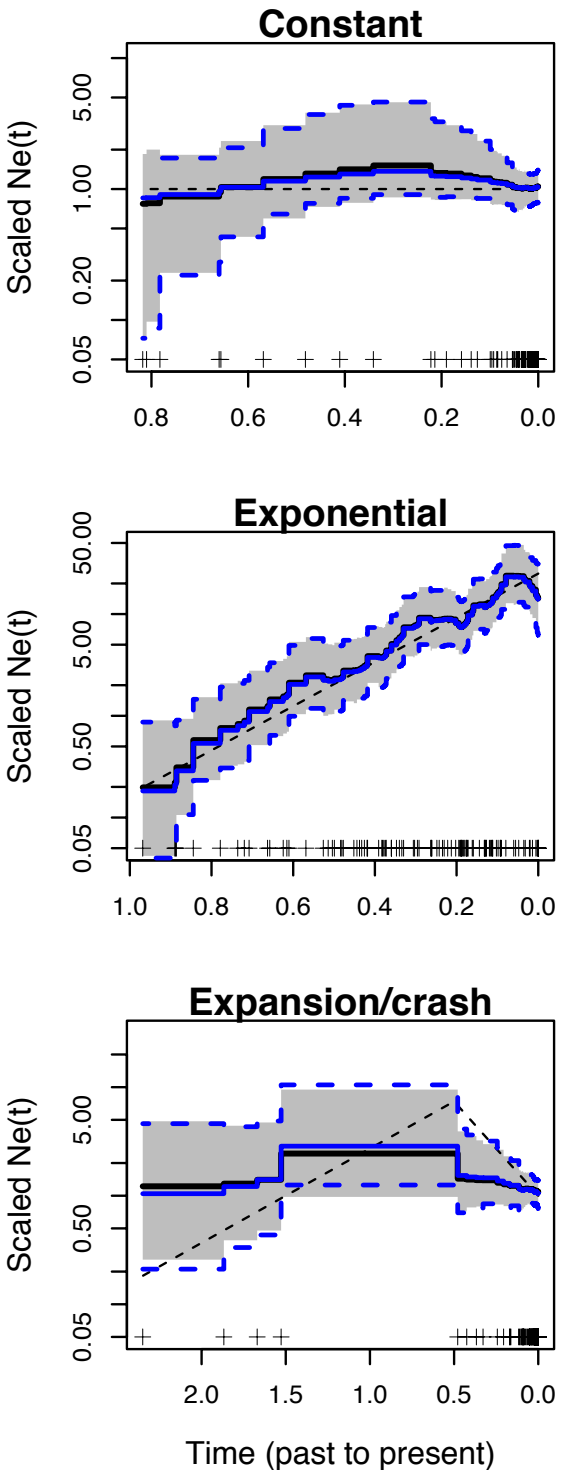


Figure 2: INLA vs MCMC for CGGP: Simulated data under the constant population size (first row), exponential growth (second row) and expansion followed by a crash (third row). The true trajectories are represented by black dashed lines. We show posterior medians estimated with MCMC sampling (solid black lines) and 95% BCIs estimated with MCMC (gray shaded areas). Posterior medians obtained using INLA are denoted by solid blue lines and INLA 95% BCIs are shown as dashed blue lines.

4 RESULTS

4.1 SIMULATED DATA

We compare INLA and MCMC approaches for the models described in sections 2.1 and 2.2. We simulate three genealogies relating $n = 100$ individuals under the following demographic scenarios:

1. Constant population size trajectory: $N_e(t) = 1$.
2. Exponential growth: $N_e(t) = 25e^{-5t}$.
3. Population expansion followed by a crash:

$$N_e(t) = \begin{cases} e^{4t} & t \in [0, 0.5], \\ e^{-2t+3} & t \in (0.5, \infty). \end{cases} \quad (16)$$

Figure 2 shows the log effective population size trajectories recovered for the three scenarios under the CGGP model using the MCMC approach (black lines and gray shaded areas) and the INLA approach (blue dark lines and blue dashed lines). In all the cases, the INLA approximation is very close to the results obtained using MCMC.

Figure 3 shows the log effective population size trajectories recovered for the same three scenarios for the continuously specified GP. In this case, the comparison is not entirely fair because we are comparing the exact MCMC method (EGP) with the doubly approximated INLA on the RGGP model. Nevertheless, both estimations look very similar for the last two cases (exponential growth and expansion followed by crash). In all cases, INLA results are very similar to the results for the CGGP model and the difference between the MCMC method and INLA methods in the constant trajectory example could be an artifact of the likelihood approximation and the convergence of the MCMC method. However, a more likely explanation is poor approximation of the marginal posterior of the Brownian motion precision, τ, by INLA. Indeed, when we examined MCMC-based and INLA-based marginal posteriors of τ, we found that the two marginals did not agree at all.

4.2 HEPATITIS C VIRUS IN EGYPT

We analyze a genealogy estimated from 63 HCV E1 sequences sampled in 1993 in Egypt. This is perhaps the most commonly used dataset for evaluating different methodologies for estimation of population size trajectories. Minin et al. (2008) compared population size trajectories recovered from a single fixed genealogy and from the sequence data directly. The authors show that there is little difference between these two

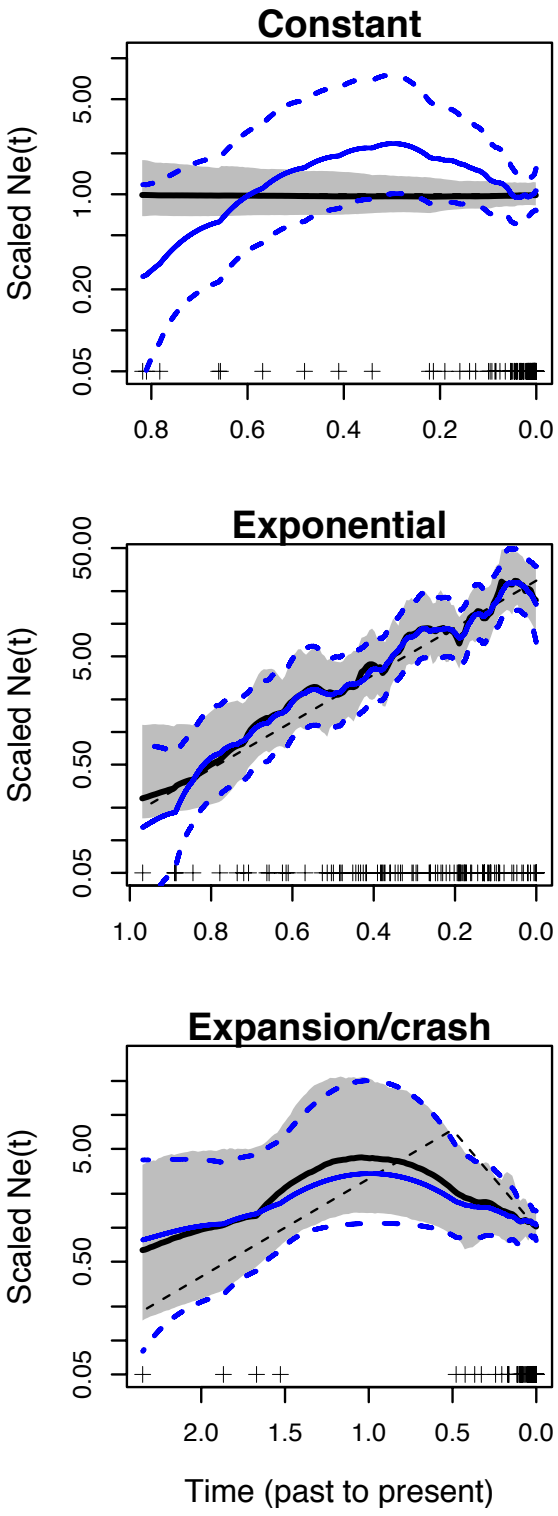


Figure 3: INLA vs MCMC for RGGP and EGP respectively: see Figure 2 for the legend.

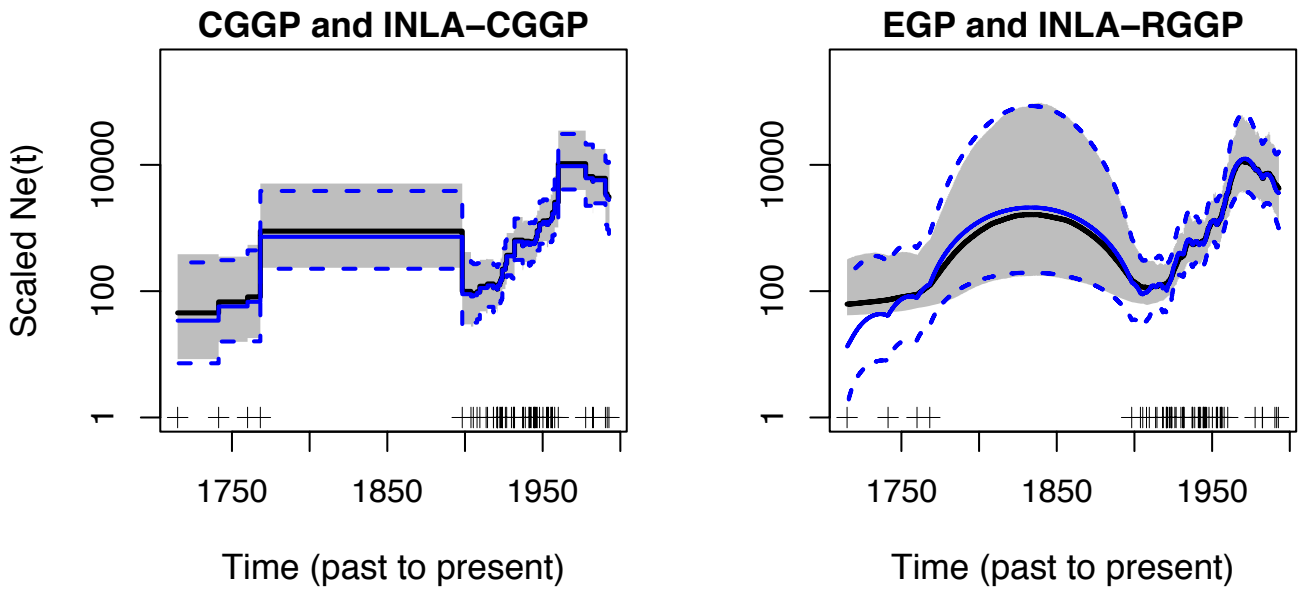


Figure 4: HCV in Egypt. Estimation of the log effective population size trajectories. In both plots, INLA approximations to posterior medians and 95% BCIs are represented by blue solid lines and blue dashed lines respectively. Approximations using MCMC sampling are represented by black solid lines and shaded areas. The left plot shows the results assuming the CGGP model and the right plot shows the result assuming the EGP for the MCMC sampling results and the RGGP model for the INLA approximation.

estimation protocols. They argue that in this case genealogical uncertainty does not play a significant role in the estimation of the Egyptian HCV population dynamics.

Figure 4 shows the recovered effective population sizes as black lines and uncertainty as gray shaded areas for the CGGP (left plot) and the EGP (right plot) using MCMC and as blue solid lines and blue dashed lines for the INLA approximation for CGGP (left plot) and RGGP (right plot). In this case, it is remarkable how similar the INLA approximations are to the MCMC results, even for the continuously specified model with the double approximation (INLA-RGGP). In all cases, the known aspects of the HCV epidemic in Egypt are recovered: an exponential growth starting around 1920s and a decline in population size after 1970s (Pybus et al., 2003).

4.3 INFLUENZA A VIRUS IN NEW YORK

We analyze a genealogy estimated from 288 H3N2 sequences sampled in New York state from January, 2001 to March, 2005 to estimate population size dynamics of human influenza A in New York. This genealogy has also been analyzed before (Palacios and Minin, 2011) and can be obtained from the authors. The key aspects of the influenza A virus epidemic in temperate regions like New York are the epidemic peaks during winters followed by strong bottlenecks at the end of the winter season. The first plot in Figure 5 shows the recovered population size trajectories assuming the CGGP model. In this case, the MCMC and the INLA approximation deviate from each other substantially, however, the expected peaks during the winter seasons in 2002, 2004 and 2005 are recovered by both methods. The MCMC approach does not recover a peak in the 2003 season, while the INLA approximation resemble more the results from the continuously specified model. INLA and MCMC results are very similar for the continuously specified model (right plot of Figure 5) with the notable differences in 95% BCIs near the time to the most recent common ancestor. This difference again may be an artifact of the double approximation involved.

4.4 RUNNING TIMES

The MCMC chains used for the CGGP model have length 1,000,000 with 100,000 of burn-in and generated using the BEAST software (Drummond and Rambaut, 2007; Minin et al., 2008) on a desktop PC. The running times range from 20 minutes to a couple of hours depending on the data. For the INLA approach, results were generated using the `R` interface `INLA` on the same computer in less than 2 seconds for all scenarios.

For the continuously specified GP model described in section 2.2, MCMC times are at best as fast as MCMC for the CGGP approach, while the results obtained

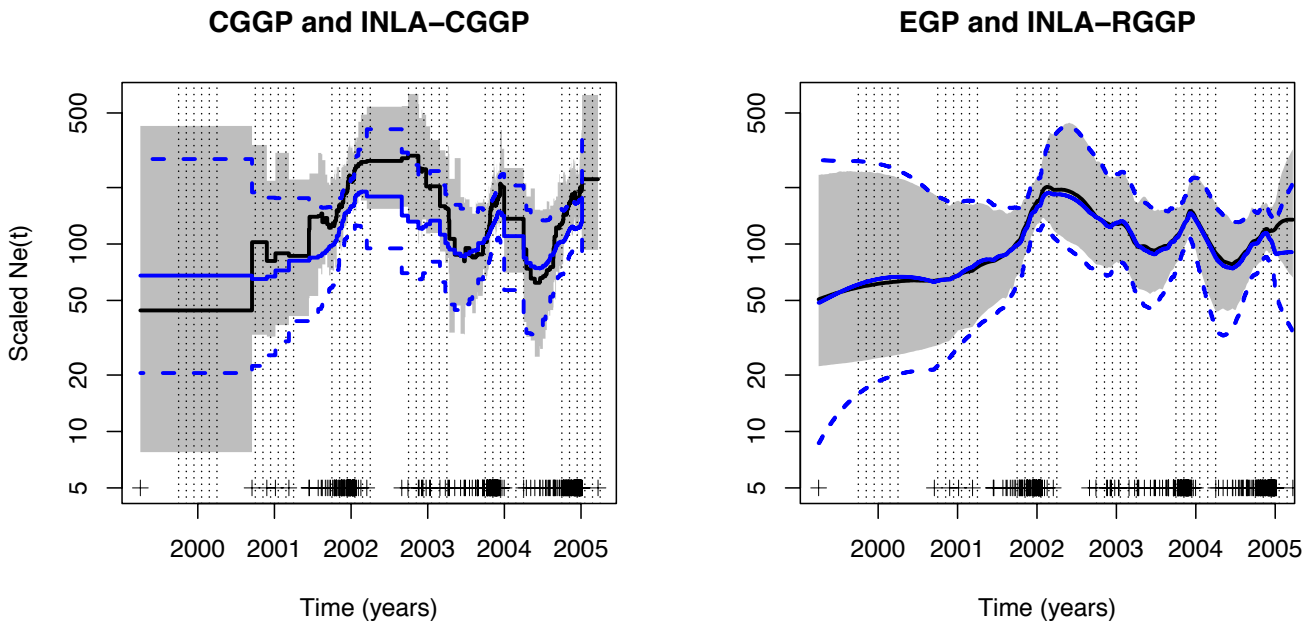


Figure 5: Influenza A in New York. Estimation of the log effective population size trajectories. In both plots, INLA approximations to posterior medians and 95% BCIs are represented by blue solid lines and blue dashed lines respectively. Approximations using MCMC sampling are represented by black solid lines and shaded areas. The left plot shows the results assuming the CGGP model and the right plot shows the result assuming the EGP for the MCMC sampling results and the RGGP model for the INLA approximation.

using `INLA`, were generated in less than 5 seconds on a grid of size 1000.

5 DISCUSSION

We show that recent Gaussian process-based Bayesian nonparametric approaches to estimation of effective population size trajectories fall into a larger class of latent Gaussian models, allowing us to perform approximate Bayesian inference using INLA. We show that it is possible to estimate population size trajectories from fixed genealogies in seconds without sacrificing any modeling advantages of recently developed Bayesian nonparametric methods.

We did observe a significant discrepancy between the INLA approximation and MCMC inference for the continuously specified GP model in the case of constant population size. We want to point out that in this case, we are not comparing apples to apples. We should be comparing INLA approximation to the MCMC for the regular grid approximation of the continuously specified GP. However, we did not have access to approximate GP-based MCMC for phylodynamics. In the absence of a better option, we are comparing INLA to the *exact* MCMC for this GP model (Palacios and Minin, 2011). Therefore, we remain uncertain whether the grid approximation or the INLA approximation is to blame for the discrepancy observed in the top plot of Figure 2. The discrepancy between the marginal posterior distributions estimated by INLA and MCMC and the fact that the precision of the RGGP likelihood discretization did not have any effect on our results suggest that INLA approximation indeed fails in this simulation scenario. This assertion is supported by another disagreement of INLA and MCMC for the CGGP model in the influenza A example, where we are comparing apples to apples.

A natural extension of the methods presented here is the incorporation of genealogical uncertainty into the model. This extension can be accomplished by introducing another level of hierarchical modeling and analyzing molecular data directly (Drummond et al., 2005; Minin et al., 2008). Even though the full posterior distribution of population trajectories from molecular sequence data no longer falls into the latent Gaussian model class, we believe that the extension is possible using Metropolis independence sampler (Rue et al., 2004). Nevertheless, the ability to obtain fast estimates of population size trajectories from a fixed genealogy (as with INLA) should be a boon for biological researchers who need to screen multiple populations of interest quickly or to provide an online analysis of epidemic outbreaks with enormous flow of molecular data in real time (Fraser et al., 2009).

There are other approaches to the estimation of effec-

tive population sizes under more complicated coalescent models that include recombination (McVean and Cardin, 2005; Li and Durbin, 2011). These methods assume a simple change point model for the effective population size trajectory. In principle, Bayesian nonparametric approaches similar to the approaches discussed here can be applied in this setting. However, presence of recombination makes such extensions potentially challenging.

Other approximate Bayesian methods could be applied to Bayesian nonparametric phylodynamics, such as variational Bayes (VB) (Bishop, 2006) and expectation propagation (EP) (Cseke and Heskes, 2010). For our particular application with a sparse GP prior, such as Brownian motion, Cseke and Heskes (2010) show that INLA should be faster than EP methods.

ACKNOWLEDGEMENTS

We acknowledge the `R-INLA` discussion group for helpful comments and the reviewers for their comments and suggestions. This work was supported by the NSF grant No. DMS-0856099. The authors partially completed this research while participating in the Program on Mathematical and Computational Approaches in High-Throughput Genomics at the NSF Institute of Pure and Applied Mathematics, UCLA.

References

Adams, R. P., Murray, I., and MacKay, D. J. (2009). Tractable nonparametric Bayesian inference in Poisson processes with Gaussian process intensities. *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 9–16.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer New York.

Campos, P. F., Willerslev, E., Sher, A., Orlando, L., Axelsson, E., Tikhonov, A., Aaris-Sørensen, K., Greenwood, A. D., Kahlke, R., Kosintsev, P., Krakhmalnaya, T., Kuznetsova, T., Lemey, P., MacPhee, R., Norris, C. A., Shepherd, K., Suchard, M. A., Zazula, G. D., Shapiro, B., and Gilbert, M. T. P. (2010). Ancient DNA analyses exclude humans as the driving force behind late pleistocene musk ox (*Ovibos moschatus*) population dynamics. *Proceedings of the National Academy of Sciences*, 107(12):5675–5680.

Cseke, B. and Heskes, T. (2010). Improving posterior marginal approximations in latent Gaussian models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 121–128.

Drummond, A. and Rambaut, A. (2007). BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology*, 7(1):214.

Drummond, A. J., Nicholls, G. K., Rodrigo, A. G., and Solomon, W. (2002). Estimating mutation parameters, population history and genealogy simultaneously from temporally spaced sequence data. *Genetics*, 161(3):1307–1320.

Drummond, A. J., Rambaut, A., Shapiro, B., and Pybus, O. G. (2005). Bayesian coalescent inference of past population dynamics from molecular sequences. *Molecular Biology and Evolution*, 22(5):1185–1192.

Felsenstein, J. (1992). Estimating effective population size from samples of sequences: inefficiency of pairwise and segregating sites as compared to phylogenetic estimates. *Genetical Research*, 59(2):139–147.

Felsenstein, J. and Rodrigo, A. G. (1999). Coalescent approaches to HIV population genetics. In *The Evolution of HIV*, pages 233–272. Johns Hopkins University Press.

Fraser, C., Donnelly, C. A., Cauchemez, S., Hanage, W. P., Van Kerkhove, M. D., Hollingsworth, T. D., Griffin, J., Baggaley, R. F., Jenkins, H. E., Lyons, E. J., Jombart, T., Hinsley, W. R., Grassly, N. C., Balloux, F., Ghani, A. C., Ferguson, N. M., Rambaut, A., Pybus, O. G., Lopez-Gatell, H., Alpuche-Aranda, C. M., Chapela, I. B., Zavala, E. P., Guevara, D. M. E., Checchi, F., Garcia, E., Hugonnet, S., Roth, C., and Collaboration, T. W. R. P. A. (2009). Pandemic potential of a strain of influenza A (H1N1): Early findings. *Science*, 324(5934):1557–1561.

Fu, Y. (1994). Estimating effective population size or mutation rate using the frequencies of mutations of various classes in a sample of DNA sequences. *Genetics*, 138(4):1375–1386.

Griffiths, R. C. and Tavaré, S. (1994). Sampling theory for neutral alleles in a varying environment. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 344(1310):403–410.

Hein, J., Schierup, M. H., and Wiuf, C. (2005). *Gene Genealogies, Variation and Evolution: A Primer in Coalescent Theory*. Oxford University Press, USA, 1st edition.

Heled, J. and Drummond, A. (2008). Bayesian inference of population size history from multiple loci. *BMC Evolutionary Biology*, 8(1):1–289.

Ho, S. Y. W. and Shapiro, B. (2011). Skyline-plot methods for estimating demographic history from nucleotide sequences. *Molecular Ecology Resources*, 11(3):423–434.

Illian, J., Sorbye, S. H., and Rue, H. (2012). A toolbox for fitting complex spatial point process models using integrated nested Laplace approximation (INLA). *Annals of Applied Statistics*, 1(2).

Kingman, J. (1982). The coalescent. *Stochastic Processes and Their Applications*, 13(3):235–248.

Kuhner, M., Yamato, J., and Felsenstein, J. (1995). Estimating effective population size and mutation rate from sequence data using Metropolis-Hastings sampling. *Genetics*, 140(4):1421–1430.

Li, H. and Durbin, R. (2011). Inference of human population history from individual whole-genome sequences. *Nature*, 475(7357):493–496.

McVean, G. and Cardin, N. (2005). Approximating the coalescent with recombination. *Philos Trans R Soc Lond B Biol Sci*, 360(1459):1387–1393.

Minin, V. N., Bloomquist, E. W., and Suchard, M. A. (2008). Smooth skyride through a rough skyline: Bayesian coalescent-based inference of population dynamics. *Molecular Biology and Evolution*, 25(7):1459–1471.

Møller, J., Syversveen, A. R., and Waagepetersen, R. P. (1998). Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482.

Nordborg, M. (2001). Coalescent theory. In *Handbook of Statistical Genetics*, chapter Coalescent Theory, pages 179–212. John Wiley & Sons, Chichester, U.K.

Opgen-Rhein, R., Fahrmeir, L., and Strimmer, K. (2005). Inference of demographic history from genealogical trees using reversible jump Markov chain Monte Carlo. *BMC Evolutionary Biology*, 5:1–6.

Palacios, J. A. and Minin, V. N. (2011). Gaussian process-based Bayesian nonparametric inference of population trajectories from gene genealogies. *ArXiv e-prints*, arXiv:1112.4138v1 [stat.ME].

Pybus, O. G., Drummond, A. J., Nakano, T., Robertson, B. H., and Rambaut, A. (2003). The epidemiology and iatrogenic transmission of hepatitis C virus in Egypt: A Bayesian coalescent approach. *Molecular Biology and Evolution*, 20(3):381–387.

Pybus, O. G., Rambaut, A., and Harvey, P. H. (2000). An integrated framework for the inference of viral population history from reconstructed genealogies. *Genetics*, 155(3):1429–1437.

Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications*. Chapman and Hall.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B*, 71(2):319–392.

Rue, H., Steinsland, I., and Erland, S. (2004). Approximating hidden Gaussian Markov random fields. *Journal of the Royal Statistical Society. Series B*, 66(4):pp. 877–892.

Slatkin, M. and Hudson, R. R. (1991). Pairwise comparisons of mitochondrial DNA sequences in stable and exponentially growing populations. *Genetics*, 129(2):555–562.

From imprecise probability assessments to conditional probabilities with quasi additive classes of conditioning events

Giuseppe Sanfilippo
Dipartimento di Scienze Statistiche e Matematiche "S. Vianelli",
University of Palermo, Italy
giuseppe.sanfilippo@unipa.it

Abstract

In this paper, starting from a *generalized coherent* (i.e. avoiding uniform loss) interval-valued probability assessment on a finite family of conditional events, we construct conditional probabilities with *quasi additive* classes of conditioning events which are consistent with the given initial assessment. Quasi additivity assures coherence for the obtained conditional probabilities. In order to reach our goal we define a finite sequence of conditional probabilities by exploiting some theoretical results on g-coherence. In particular, we use solutions of a finite sequence of linear systems.

1 Introduction

In many applications of Artificial Intelligence we need to reason with uncertain information under vague or partial knowledge. A possible approach to uncertain reasoning can be based on imprecise probabilistic assessments on a family of conditional events which has no particular algebraic structure. In such a case a general framework is obtained by using suitable generalizations of the coherence principle of de Finetti (de Finetti [1974]), or similar principles adopted for lower and upper probabilities. A further advantage of using these approaches is given by the possibility of looking at the conditional probability $P(E|H)$ as a primitive concept, with no need of assuming that the probability of the conditioning event H be positive. In the past, the coherence-based approach to probabilistic reasoning has covered several natural semantics for conditionals, included the ones used in default reasoning such as System P (see, e.g., Gilio [2002], Biazzo et al. [2005], Gilio and Sanfilippo [2011]). In this paper we adopt a notion of generalized coherence called *g-coherence* given in Biazzo and Gilio [2000] (see also Coletti [1994], Gilio [1995b], Biazzo et al. [2005]). The notion of g-coherence is weaker than the notion of coherence given for lower and upper probabilities (see Walley [1991], Williams [2007]) and is equivalent (Biazzo and Gilio [2002]) to the property of "avoiding uniform loss" given in Walley [1991, 1997]. Using some algorithms, a g-coherent probability assessment can be corrected obtaining a coherent assessment. We recall that, given a coherent (precise) assessment $\mathcal{P}$ on an arbitrary family $\mathcal{F}$ of conditional events, there always exists a coherent extension of $\mathcal{P}$ which is a conditional probability (Regazzini [1985]), see also Holzer [1985], Rigo [1988]). Conversely, given a conditional probability P on $\mathcal{E} \times \mathcal{X}$, where $\mathcal{E}$ is an algebra and $\mathcal{X}$ is a nonempty subset of $\mathcal{E} \setminus \{\emptyset\}$, in Rigo [1988] is shown that a suitable condition given in Császár [1955] (called Csàszàr's condition) is necessary and sufficient for coherence of P. If no restrictions are supposed regarding the class of conditioning events $\mathcal{X}$ the function P could be not coherent (Gilio and Spezzaferri [1992], Gilio [1995a], Coletti and Scozzafava [2002]). Moreover, if $\mathcal{X}$ has a particular structure, for instance $\mathcal{X}$ is additive (Holzer [1985], Coletti and Scozzafava [2002]) or quasi-additive (Gilio [1989]), then P is coherent. As quasi-additivity property is weaker than additivity it is of some interest to check if a given coherent assessment $\mathcal{P}$ could be extended as a conditional probability with the class of conditioning events quasi-additive. We remark that the such conditional probabilities may be useful because in many applications in order to take a decision we need to choose a precise and (possibly complete) probability assessment. Hence, the kind of problems studied in this paper can help the decision maker to 'integrate knowledge' by computing a probability distribution consistent with the 'incomplete or vague' initial information quantified by the imprecise assessment.

In what follows, starting from a g-coherent interval-valued probability assessment $\mathcal{A}$ on a finite family $\mathcal{F}$ of conditional events we first determine the associated interval-valued assessment $\mathcal{A}^*$ on $\mathcal{F}$ coherent in the

sense of Walley. Then, after recalling the notion of finitely additive conditional probability and for the set of conditioning events the property of quasi additivity, we construct a sequence of conditional probabilities with quasi additive classes of conditioning events. We extend the conditional probabilities of this sequence using the same (quasi additive) classes of conditioning events. Thus, we are able to define a conditional probability $P_{01\ldots k}$ on $\mathcal{E} \times \mathcal{X}$, where $\mathcal{E}$ is the algebra generated by $\mathcal{F}$ and $\mathcal{X}$ is a quasi additive class which coincides with the union of the previous classes of conditioning events. Moreover, $\mathcal{X}$ contains the initial set of conditioning events of $\mathcal{F}$. Finally, we observe that $P_{01\ldots k}$ is a coherent conditional probability on $\mathcal{F}$ consistent with $\mathcal{A}$. By this procedure one can obtain a large (potentially infinite) number of coherent conditional probabilities. The paper is organized as follows. In Section 2 we first give some preliminary notions and results on generalized coherence. Then, we recall the concepts of conditional probability and quasi additive class of events. Finally we recall a sufficient condition for coherence of conditional probability. In Section 3 we show how a g-coherence assessment, using some algorithms, can be corrected obtaining a coherent assessment. In Section 4 given a g-coherent interval-valued assessment $\mathcal{A}$ on $\mathcal{F}$ we construct a finite sequence $P_0, P_1, \ldots, P_k$ of conditional probabilities with quasi additive families of conditioning events. In Section 5 we introduce a new sequence of conditional probabilities $P_0^0, P_1^0, \ldots, P_k^0$ with quasi additive families of conditioning events such that, for each i, the probability P_i^0 is a particular extension of P_i. We also show that P_i^0 is coherent. In Section 6 we define a conditional probability $P_{01\ldots k}$ with a quasi additive class of conditioning events which contains all conditioning events on $\mathcal{F}$. We show that $P_{01\ldots k}$ is coherent and its restriction on $\mathcal{F}$ is a coherent assessment which is consistent with $\mathcal{A}$. In Section 7 we give a characterization theorem of coherent precise assessments and a relevant way of introducing quasi additive classes. Finally, we give some conclusions.

2 Preliminary notions and results

In this section we set up notation and terminology. We also recall some basic concepts and results, related with the checking of g-coherence and propagation of conditional probability bounds. Next, we recall the definition of conditional probabilities. Finally, we recall the notion of quasi-additive class of events and a sufficient condition for coherence which will be used in the next sections.

2.1 Notations

For each integer n we set $J_n = \{1, 2, \ldots, n\}$. Moreover, we denote respectively by Ω the sure event, by $\emptyset$ the impossible event and the empty set, by E^c the negation of the event E, by $E \vee H$ the disjunction of E and H and by $E \wedge H$ (or simply by EH) the conjunction of E and H. The logical implication from E to H, namely *if E is true, then also H is true*, is denoted by $E \subseteq H$.

2.2 Constituents

Given any pair of events E and H, with $H \neq \emptyset$, we look at the conditional event $E|H$ as a three-valued logical entity which is true, or false, or void, according to whether EH is true, or E^cH is true, or H^c is true. In the setting of coherence, agreeing to the betting metaphor, if you assess $P(E|H) = p$, then you agree to pay an amount p, by receiving 1, or 0, or p, according to whether $E|H$ is true, or $E|H$ is false, or $E|H$ is void (bet called off). Given a family of conditional events $\mathcal{F}_{J_n} = \{E_i|H_i, i \in J_n\}$, we set $\mathcal{H}_{J_n} = \bigvee_{j \in J_n} H_j$. For each $r \in \{0, 1, \ldots 3^n - 1\}$, we denote by $(r_n r_{n-1} \cdots r_2 r_1)$ its ternary representation, such as $r = r_1 3^0 + r_2 3^1 + \cdots + r_n 3^{r_n - 1}$, and by R_r the event defined as $R_r = R_1^{r_1} R_2^{r_2} \cdots R_n^{r_n}$, where

$$R_i^{r_i} = \begin{cases} E_i H_i, & \text{if } r_i = 1, \\ E_i^c H_i, & \text{if } r_i = 0, \\ H_i^c, & \text{if } r_i = 2. \end{cases}$$

Then, we introduce the set

$$\mathcal{C}_{J_n} = \{R_r \neq \emptyset, r \in \{0, 1, 2, \ldots, 3^n - 1\}\}$$

containing every R_r that is not impossible. As some conjunction R_r may be impossible, the cardinality of $\mathcal{C}_{J_n}$ is less than or equal to 3^n. We call *constituents* or *possible worlds* associated with $\mathcal{F}_{J_n}$ the elements of $\mathcal{C}_{J_n}$. Obviously $\mathcal{C}_{J_n}$ is a partition of Ω, that is
1. $\bigvee_{C \in \mathcal{C}_{J_n}} C = \Omega$,
2. $C' \wedge C'' = \emptyset$ for each $C', C'' \in \mathcal{C}_{J_n}$ with $C' \neq C''$.
With each event E we associate the following subset of $\mathcal{C}_{J_n}$ $C_{J_n}(E) = \{C \in \mathcal{C}_{J_n} : C \subseteq E\}$. Notice that for each pair of events E_iH_i,H_i, since $E_iH_i \subseteq H_i \subseteq \mathcal{H}_{J_n}$, it follows that $C_{J_n}(E_iH_i) \subseteq C_{J_n}(H_i) \subseteq C_{J_n}(\mathcal{H}_{J_n})$.

Example 1. Let $\mathcal{F}_{J_3} = \{E_1|H_1, E_2|H_2, E_3|H_3\} = \{ABC|D, B|AC, C|AB\}$ be a family of three conditional events. The set $\mathcal{C}_{J_3}$ of the constituents associated with $\mathcal{F}_{J_3}$ is

$$\mathcal{C}_{J_3} = \{R_6, R_8, R_{13}, R_{14}, R_{18}, R_{20}, R_{24}, R_{26}\} = \\ = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\},$$

where

$$C_1 = ABC^cD,\ C_2 = ABC^cD^c,\ C_3 = ABCD, \\ C_4 = ABCD^c,\ C_5 = AB^cCD,\ C_6 = AB^cCD^c, \\ C_7 = (A^c \vee B^cC^c)D,\ C_8 = (A^c \vee B^cC^c)D^c\,.$$

Moreover, we have that $\mathcal{H}_{J_3} = \{D \vee AC \vee AB\}$ and

$$\mathcal{C}_{J_3}(\mathcal{H}_{J_3}) = \mathcal{C}_{J_3} \setminus \{C_8\} = \{C_1, C_2, \ldots, C_7\}. \quad (1)$$

2.3 Coherence and g-coherence

Given an arbitrary family of conditional events $\mathcal{F}$ and a real function $\mathcal{P}$ on $\mathcal{F}$, for every $n \in \mathbb{N}$, let $\mathcal{F}_{J_n} = \{E_i|H_i, i \in J_n\}$ be a subfamily of $\mathcal{F}$ and $\mathcal{P}_{J_n}$ the vector $(p_i, i \in J_n)$, where $p_i = \mathcal{P}(E_i|H_i)$. We use the same symbols for events and their indicator. Then, considering the random gain

$$G_{J_n} = \sum_{i \in J_n} s_i H_i (E_i - p_i),$$

with $s_i, i \in J_n$, arbitrary real numbers, we denote by $G_{J_n}|\mathcal{H}_{J_n}$ the restriction of G_{J_n} to $\mathcal{H}_{J_n}$. Then, based on the betting scheme, we have

Definition 1. The function $\mathcal{P}$ is said coherent iff

$$\max G_{J_n}|\mathcal{H}_{J_n} \geq 0,\ \forall n \geq 1, \forall \mathcal{F}_{J_n} \subseteq \mathcal{F}, \forall s_i \in \mathbb{R}, i \in J_n.$$

Given an interval-valued probability assessment $\mathcal{A}_{J_n} = ([a_1, b_1], \ldots, [a_n, b_n])$ on a family $\mathcal{F}_{J_n} = \{E_i|H_i, i \in J_n\}$ we adopt the following condition of generalized coherence (*g-coherence*) given in Biazzo and Gilio [2000].

Definition 2. The interval-valued probability assessment $\mathcal{A}_{J_n}$ on $\mathcal{F}_{J_n}$ is said g-coherent if and only if there exists a precise coherent assessment $\mathcal{P}_{J_n} = (p_i,\ i \in J_n)$ on $\mathcal{F}_{J_n}$, with $p_i = P(E_i|H_i)$, which is consistent with $\mathcal{A}_{J_n}$, that is such that $a_i \leq p_i \leq b_i$ for each $i \in J_n$.

Given an interval-valued probability assessment $\mathcal{A}_{J_n}$ on $\mathcal{F}_{J_n}$, we denote by $(\mathcal{S}_{J_n})$ the following system, associated with the pair $(\mathcal{F}_{J_n}, \mathcal{A}_{J_n})$, with nonnegative unknown $\underline{\lambda} = (\lambda_C,\ C \in \mathcal{C}_{J_n}(\mathcal{H}_{J_n}))$:

$$(\mathcal{S}_{J_n}) \begin{cases} \sum_{C \in \mathcal{C}_{J_n}(E_i H_i)} \lambda_C \leq b_i \cdot \sum_{C \in \mathcal{C}_{J_n}(H_i)} \lambda_C\,, \forall i \in J_n \\ \sum_{C \in \mathcal{C}_{J_n}(E_i H_i)} \lambda_C \geq a_i \cdot \sum_{C \in \mathcal{C}_{J_n}(H_i)} \lambda_C\,, \forall i \in J_n \\ \sum_{C \in \mathcal{C}_{J_n}(\mathcal{H}_{J_n})} \lambda_C = 1\,,\ \ \lambda_C \geq 0, \forall C \in \mathcal{C}_{J_n}(\mathcal{H}_{J_n}). \end{cases}$$

In an analogous way, given a subset J of J_n, we denote by $(\mathcal{F}_J, \mathcal{A}_J)$ the pair corresponding to J, by $(\mathcal{S}_J)$ the system associated with $(\mathcal{F}_J, \mathcal{A}_J)$ and by $\mathcal{C}_J$ the set of constituents associated with $\mathcal{F}_J$.
We recall a result given in Gilio [1995a], where the notion of g-coherence (see Biazzo and Gilio [2000]) was simply named coherence.

Theorem 1. *The interval-valued probability assessment $\mathcal{A}_{J_n}$ on $\mathcal{F}_{J_n}$ is g-coherent if and only if, for every $J \subseteq J_n$, the system $(\mathcal{S}_J)$ is solvable.*

2.4 Conditional probabilities

Given an algebra of events $\mathcal{E}$ and a non empty subfamily $\mathcal{X}$ of $\mathcal{E}$, with $\emptyset \notin \mathcal{X}$, a (finitely-additive) *conditional probability* on $\mathcal{A} \times \mathcal{X}$ is a real-valued function P defined on $\mathcal{E} \times \mathcal{X}$ satisfying the following properties (Dubins [1975], see also Rényi [1955], Császár [1955], Regazzini [1985], Coletti [1994], Coletti and Scozzafava [2002]):

- (i) $P(\cdot|H)$ is a finitely additive probability on $\mathcal{E}$, for each $H \in \mathcal{X}$;
- (ii) $P(H|H)$=1, for each $H \in \mathcal{X}$;
- (iii) $P(E_1E_2|H) = P(E_2|E_1H)P(E_1|H)$, for every E_1, E_2, H, with $E_1 \in \mathcal{E}$, $E_2 \in \mathcal{E}$, $H \in \mathcal{X}$ and $E_1H \in \mathcal{X}$.

As in Rényi [1955] we do not suppose any restrictions regarding the class of conditioning events $\mathcal{X}$ of the conditional probability P. In Dubins [1975], in order to define a finitely-additive conditional probability, it is required that $\mathcal{X} \cup \{\emptyset\}$ must be a subalgebra of $\mathcal{E}$.

2.5 Quasi additivity and coherence

Let P be a conditional probability on $\mathcal{E} \times \mathcal{X}$, the family $\mathcal{X}$ of conditioning events is said a *$P-$quasi-additive* class (or quasi additive w.r.t. P) if, for every $H_1 \in \mathcal{X}, H_2 \in \mathcal{X}$, there exists $K \in \mathcal{X}$ such that (Császár [1955])

$$\begin{array}{l} (i)\ \ H_1 \vee H_2 \subseteq K, \\ (ii)\ \ P(H_1|K) + P(H_2|K) > 0. \end{array} \quad (2)$$

We observe that, given any conditional probability P on $\mathcal{E} \times \mathcal{X}$, if $\mathcal{X}$ is additive or $\mathcal{X} \cup \{\emptyset\}$ is a subalgebra of $\mathcal{E}$, then $\mathcal{X}$ is $P-$quasi additive. Some sufficient conditions for coherence of a conditional probability are given in Gilio [1989] from which we recall the following result

Theorem 2. *If P is a conditional probability on $\mathcal{E} \times \mathcal{X}$, with $\mathcal{E}$ algebra and $\mathcal{X}$ quasi additive class w.r.t. P, then P is coherent.*

3 From g-coherent to coherent lower/upper probabilities

Let $\mathcal{A}_{J_n} = ([a_i, b_i],\ i \in J_n)$ be a g-coherent interval-valued probability assessment on $\mathcal{F}_{J_n} = \{E_i|H_i,\ i \in J_n\}$. Of course, we can assume that, if $E_i|H_i \in \mathcal{F}_{J_n}$, then $E_i^c|H_i \notin \mathcal{F}_{J_n}$. Moreover, if we replace each upper bound $P(E_i|H_i) \leq b_i$ by the equivalent lower bound $P(E_i^c|H_i) \geq 1 - b_i$, the assessment $\mathcal{A}_{J_n}$ on $\mathcal{F}_{J_n}$ can be seen as a lower probability $(a_i, 1 - b_i,\ i \in J_n)$ on the family $\{E_i|H_i, E_i^c|H_i,\ i \in J_n\}$. Then, using a suitable alternative theorem, it can be shown that g-coherence

and avoiding uniform loss (AUL) property of lower and upper probabilities (Walley [1991]) are equivalent (see Biazzo and Gilio [2002], Theorem 10).
Given a further conditional event $E_{n+1}|H_{n+1}$, as it can be verified (Biazzo and Gilio [2000]), there exists a suitable interval $[p_\circ, p^\circ]$ such that

Theorem 3. *The interval-valued assessment* $\mathcal{A}_{J_{n+1}} = ([a_i, b_i],\ i \in J_{n+1})$ *with* $a_{n+1} = b_{n+1} = p_{n+1}$ *on the family* $\mathcal{F}_{J_{n+1}} = \{E_i|H_i,\ i \in J_{n+1}\}$, *is g-coherent if and only if* $p_{n+1} \in [p_\circ, p^\circ]$.

Then, it immediately follows

Theorem 4. *Given a g-coherent interval-valued assessment* $\mathcal{A}_{J_n} = ([a_i, b_i],\ i \in J_n)$ *on the family* $\mathcal{F}_{J_n} = \{E_i|H_i,\ i \in J_n\}$, *the extension* $[a_{n+1}, b_{n+1}]$ *of* $\mathcal{A}_{J_n}$ *to a further conditional event* $E_{n+1}|H_{n+1}$ *is g-coherent if and only if* $[a_{n+1}, b_{n+1}] \cap [p_\circ, p^\circ] \neq \emptyset$.

The values $p_\circ, p^\circ$ can be determined by exploiting a suitable algorithm given in Biazzo and Gilio [2000]. By the same algorithm, starting with a g-coherent assessment $\mathcal{A}_{J_n}$ on $\mathcal{F}_{J_n}$, we can make the "least-committal" correction (see Pelessoni and Vicig [1998]) of $\mathcal{A}_{J_n}$.
In this way, we obtain the coherent (lower and upper) probability $\mathcal{A}^*_{J_n}$ on $\mathcal{F}_{J_n}$ which would be produced by applying the natural extension principle proposed in Walley [1991].
To determine $\mathcal{A}^*_{J_n}$, we just need to apply n times such algorithm, by replacing each time $E_{n+1}|H_{n+1}$ by $E_j|H_j$, $j \in J_n$, using as probabilistic constraints on the conditional events of $\mathcal{F}_{J_n}$ the g-coherent assessment $\mathcal{A}_{J_n}$.
Example 1 (continued)
Let $\mathcal{A}_{J_3} = ([\frac{1}{2}, 1], [0, \frac{1}{2}], [\frac{1}{3}, \frac{2}{3}])$ be an imprecise probability assessment on $\mathcal{F}_{J_3} = \{ABC|D, B|AC, C|AB\}$. By applying the algorithm for checking g-coherence given in Biazzo and Gilio [2000] it can be proved that $\mathcal{A}_{J_3}$ is g-coherent. Moreover, $\mathcal{A}_{J_3}$ is coherent [1] and then $\mathcal{A}_{J_3} = \mathcal{A}^*_{J_3}$.

4 Construction of classes of conditional probabilities with quasi additive families of conditioning events

In this section, starting with a finite interval-valued probability assessment, we will construct classes of conditional probabilities with quasi additive families of conditioning events. Quasi additivity will assure coherence for the obtained conditional probabilities.

Remark 1. We observe that $\mathcal{C}_{J_n} = \mathcal{C}_{J_n}(\mathcal{H}_{J_n}) \cup \{\mathcal{H}^c_{J_n}\}$. In particular, $\mathcal{C}_{J_n} = \mathcal{C}_{J_n}(\mathcal{H}_{J_n})$, if $\mathcal{H}_{J_n} = \Omega$. Moreover,

[1] Coherence can also be checked by the CkC-package (Baioletti et al.) available at
`http://www.dmi.unipg.it/~upkd/paid/software.html`

the system $(\mathcal{S}_{J_n})$ is solvable if and only if the following system is solvable

$$(\mathcal{S}^*_{J_n}) \begin{cases} \sum_{C \in \mathcal{C}_{J_n}(E_i H_i)} \lambda_C \leq b_i \cdot \sum_{C \in \mathcal{C}_{J_n}(H_i)} \lambda_C, \forall\, i \in J_n \\ \sum_{C \in \mathcal{C}_{J_n}(E_i H_i)} \lambda_C \geq a_i \cdot \sum_{C \in \mathcal{C}_{J_n}(H_i)} \lambda_C, \forall\, i \in J_n \\ \sum_{C \in \mathcal{C}_{J_n}(\mathcal{H}_{J_n})} \lambda_C > 0\,, \quad \lambda_C \geq 0 \ \ \forall\, C \in \mathcal{C}_{J_n}. \end{cases}$$

In fact, given a solution $\underline{\lambda}^* = (\lambda^*_C, C \in \mathcal{C}_{J_n})$ of $(\mathcal{S}^*_{J_n})$, the vector $\underline{\lambda}' = (\lambda'_C, C \in \mathcal{C}_{J_n}(\mathcal{H}_{J_n}))$, defined by

$$\lambda'_C = \frac{\lambda^*_C}{\sum_{C \subseteq \mathcal{H}_{J_n}} \lambda^*_C}\,,\ \forall\, C \in \mathcal{C}_{J_n}(\mathcal{H}_{J_n})\,,$$

is a solution of $(\mathcal{S}_{J_n})$. Conversely, given a solution $\underline{\lambda}' = (\lambda'_C, C \in \mathcal{C}_{J_n}(\mathcal{H}_{J_n}))$ of $(\mathcal{S}_{J_n})$, any vector $\underline{\lambda}^* = (\lambda^*_C, C \in \mathcal{C}_{J_n})$, with

$$\lambda^*_C = \alpha \lambda'_C\,,\ \forall C \in \mathcal{C}_{J_n}(\mathcal{H}_{J_n})\,,\ \ \alpha > 0\,,\ \ \lambda^*_{\mathcal{H}^c_{J_n}} \geq 0\,,$$

is a solution of $(\mathcal{S}^*_{J_n})$. Of course, the variable $\lambda_{\mathcal{H}^c_{J_n}}$ in $(\mathcal{S}^*_{J_n})$ disappears when $\mathcal{H}^c_{J_n} = \emptyset$.
In what follows, we will use system $(\mathcal{S}^*_{J_n})$ in the equivalent formulation given below

$$(\mathcal{S}^*_{J_n}) \begin{cases} \sum_{C \in \mathcal{C}_{J_n}(E_i H_i)} \lambda_C \leq b_i \cdot \sum_{C \in \mathcal{C}_{J_n}(H_i)} \lambda_C, \forall i \in J_n \\ \sum_{C \in \mathcal{C}_{J_n}(E_i H_i)} \lambda_C \geq a_i \cdot \sum_{C \in \mathcal{C}_{J_n}(H_i)} \lambda_C, \forall i \in J_n \\ \sum_{C \in \mathcal{C}_{J_n}} \lambda_C = \sum_{C \in \mathcal{C}_{J_n}(\mathcal{H}_{J_n})} \lambda_C = 1\,, \\ \lambda_C \geq 0\ \ , \forall C \in \mathcal{C}_{J_n}. \end{cases}$$

Remark 2. We denote by Π the set of coherent precise assessments $\mathcal{P} = (p_{E|H},\ \ E|H \in \mathcal{F}_{J_n})$ on $\mathcal{F}_{J_n}$ which are consistent with $\mathcal{A}_{J_n}$. Let $\mathcal{A}^*_{J_n} = ([a^*_i, b^*_i], i \in J_n)$ be the coherent assessment associated with $\mathcal{A}_{J_n}$, computed by the procedure cited in the previous section. We recall that coherence of $\mathcal{A}^*_{J_n}$ amounts to the existence, for any given $j \in J_n$ and any $x_j \in [a^*_j, b^*_j]$, of a coherent precise probability assessment $(p_{E_i|H_i},\ \ i \in J_n)$ on $\mathcal{F}_{J_n}$, which is consistent with $\mathcal{A}^*_{J_n}$ and is such that $p_{E_j|H_j} = x_j$. We observe that, denoting by Π^* the set of coherent precise assessments $\mathcal{P} = (p_{E|H},\ \ E|H \in \mathcal{F}_{J_n})$ on $\mathcal{F}_{J_n}$ which are consistent with $\mathcal{A}^*_{J_n}$, it holds that $\Pi = \Pi^*$.

To construct the classes of conditional probabilities associated with a given pair $(\mathcal{F}_{J_n}, \mathcal{A}_{J_n})$, we will use a suitable finite sequence $(\mathcal{F}_0, \mathcal{A}_0)$, $(\mathcal{F}_1, \mathcal{A}_1), \ldots, (\mathcal{F}_k, \mathcal{A}_k)$, with $(\mathcal{F}_0, \mathcal{A}_0) = (\mathcal{F}_{J_n}, \mathcal{A}_{J_n})$ and $\mathcal{F}_0 \supset \mathcal{F}_1 \supset \cdots \supset \mathcal{F}_k$, where $\mathcal{A}_j$ is the sub-assessment associated with $\mathcal{F}_j$.
Rather than discuss this in full generality, let us look at

$(\mathcal{F}_0, \mathcal{A}_0)$. Let $\mathcal{C}_0$ be the set of constituents associated with $\mathcal{F}_0$ and Π_0 the set of coherent precise assessments $\mathcal{P}$ on $\mathcal{F}_0$ which are consistent with $\mathcal{A}_0$. Then, given a precise coherent assessment $\mathcal{P}_0 = (p^{(0)}_{E|H}, \quad E|H \in \mathcal{F}_0) \in \Pi_0$ on $\mathcal{F}_0$ we consider the following system $(\mathcal{S}_0)$ in the unknowns $\underline{\lambda} = (\lambda_C, C \in \mathcal{C}_0)$ associated with $(\mathcal{P}_0, \mathcal{F}_0)$

$$(\mathcal{S}_0) \begin{cases} \sum\limits_{C \in \mathcal{C}_0(EH)} \lambda_C = p_{E|H} \cdot \sum\limits_{C \in \mathcal{C}_0(H)} \lambda_C, \ \forall E|H \in \mathcal{F}_0 \\ \sum\limits_{C \in \mathcal{C}_0} \lambda_C = \sum\limits_{C \in \mathcal{C}_0(\mathcal{H}_0)} \lambda_C = 1\,, \ \ \lambda_C \geq 0 \ \ \forall C \in \mathcal{C}_0, \end{cases}$$

where $\mathcal{H}_0 = \bigvee_{E|H \in \mathcal{F}_0} H$.
We observe that $\mathcal{P}_0$ is a particular interval-valued probability assessment on $\mathcal{F}_0$ where each upper bound coincides with the respective lower bound. Therefore, based on Theorem 1 and Remark 1, the system $(\mathcal{S}_0)$ is solvable. Denoting by $\mathcal{E}_0$ the algebra generated by the elements of $\mathcal{C}_0$, we introduce the following real function of the pair $(E, \underline{\lambda})$, with E in $\mathcal{E}_0$ and $\underline{\lambda} = (\lambda_C, C \in \mathcal{C}_0)$ a vector of non-negative numbers,

$$\phi_0(E, \underline{\lambda}) = \begin{cases} \sum_{C \in \mathcal{C}_0(E)} \lambda_C, & \text{if } C_0(E) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Next, let $\underline{\lambda}^\star = (\lambda^\star_C : C \in \mathcal{C}_0)$ be a solution of the system $(\mathcal{S}_0)$ and $D_0 = \{H : E|H \in \mathcal{F}_0\}$ be the set of conditioning events of $\mathcal{F}_0$, we consider the following partition of D_0

$$D_0^z = \{H \in D_0 \ : \phi_0(H, \underline{\lambda}^\star) = 0\}, \tag{4}$$

$$D_0^+ = D_0 \setminus D_0^z = \{H \in D_0 \ : \phi_0(H, \underline{\lambda}^\star) > 0 \ \}. \tag{5}$$

Remark 3. Note that the subset D_0^+ cannot be empty (that is $D_0^z \subset D_0$). In fact, as $\underline{\lambda}^\star$ is a solution of (S_0) we have

$$\sum_{H \in D_0} \phi_0(H, \underline{\lambda}^\star) \geq \sum_{C \in \mathcal{C}_0(\mathcal{H}_0)} \lambda^\star_C = \sum_{C \in \mathcal{C}_0} \lambda^\star_C = 1.$$

Hence, there exists an event $H \in D_0$ such that $\phi(H, \underline{\lambda}^\star) > 0$, that is $D_0^+ \neq \emptyset$.

Let $\mathcal{X}_0 = D_0^+ \cup \{\mathcal{H}_0\}$ be a family of conditioning events and $\underline{\lambda}^\star$ be a solution of (S_0), we set

$$P_0(E|H) = \frac{\phi_0(EH, \underline{\lambda}^\star)}{\phi_0(H, \underline{\lambda}^\star)}, \forall E|H, \text{ with } E \in \mathcal{E}_0, \ H \in \mathcal{X}_0. \tag{6}$$

Then, we have

Theorem 5. *The real function P_0 is a conditional probability on $\mathcal{E}_0 \times \mathcal{X}_0$, with $\mathcal{X}_0$ quasi-additive w.r.t. P_0.*

Proof. In order to prove that P_0 is a conditional probability on $\mathcal{E}_0 \times \mathcal{X}_0$ we need to show that P_0 satisfies properties $(i), (ii), (iii)$. Then, such a proof will be divided into three steps.

1) Let H be a conditioning event in $\mathcal{X}_0$. We obviously have
$$P_0(\Omega|H) = \frac{\phi_0(H, \underline{\lambda}^\star)}{\phi_0(H, \underline{\lambda}^\star)} = 1.$$
Moreover, for every $E \in \mathcal{E}_0$ one has $P_0(E|H) \geq 0$. Finally, for every couple of incompatible events E_1, E_2 in $\mathcal{E}_0$, we have
$$P_0(E_1 \vee E_2|H) = \tfrac{\phi_0(E_1H \vee E_2H, \underline{\lambda}^\star)}{\phi_0(H, \underline{\lambda}^\star)} = \tfrac{\phi_0(E_1H, \underline{\lambda}^\star)}{\phi_0(H, \underline{\lambda}^\star)} + \tfrac{\phi_0(E_2H, \underline{\lambda}^\star)}{\phi_0(H, \underline{\lambda}^\star)} = P_0(E_1|H) + P_0(E_2|H).$$
Therefore, for each $H \in \mathcal{X}_0$, $P_0(\cdot|H)$ is a finitely additive probability on $\mathcal{E}_0$; that is P_0 satisfies property (i).

2) For each conditioning event $H \in \mathcal{X}_0$ we trivially have $P_0(H|H) = 1$. Then, P_0 satisfies property (ii).

3) Let E_1, E_2, H be three events in $\mathcal{E}_0$ such that H and E_1H are in $\mathcal{X}_0$. Then, one has $\phi_0(H, \underline{\lambda}^\star) > 0$ and $\phi_0(E_1H, \underline{\lambda}^\star) > 0$. Therefore, we obtain
$$\begin{aligned} P_0(E_1E_2|H) &= \tfrac{\phi_0(E_1E_2H, \underline{\lambda}^\star)}{\phi_0(H, \underline{\lambda}^\star)}, \\ P_0(E_2|E_1H) &= \tfrac{\phi_0(E_1E_2H, \underline{\lambda}^\star)}{\phi_0(E_1H, \underline{\lambda}^\star)}, \\ P_0(E_1|H) &= \tfrac{\phi_0(E_1H, \underline{\lambda}^\star)}{\phi_0(H, \underline{\lambda}^\star)}. \end{aligned}$$
Hence, it is easily seen that $P_0(E_1E_2|H) = P_0(E_2|E_1H)P_0(E_1|H)$, so that P_0 satisfies property (iii).

Therefore, P_0 is a conditional probability on $\mathcal{E}_0 \times \mathcal{X}_0$. Next, we prove that $\mathcal{X}_0$ is a P_0-quasi additive class. Given two conditioning events $H_1, H_2 \in \mathcal{X}_0$ we have $H_1 \vee H_2 \subseteq \mathcal{H}_0$. In addition, as $\phi_0(H_1, \underline{\lambda}^\star) > 0$ and $\phi_0(H_2, \underline{\lambda}^\star) > 0$, it follows that $P_0(H_1|\mathcal{H}_0) > 0$ and $P_0(H_2|\mathcal{H}_0) > 0$. Hence, conditions (i) and (ii) in (2) hold with $K = \mathcal{H}_0$. Hence, $\mathcal{X}_0$ is a P_0-quasi additive class. □

From Theorem 5 and Theorem 2, it immediately follows

Corollary 1. *The conditional probability P_0 on $\mathcal{E}_0 \times \mathcal{X}_0$ is coherent.*

We define the following partition of $\mathcal{F}_0$

$$\mathcal{F}_0^z = \{E|H \in \mathcal{F}_0 : H \in D_0^z\}$$
$$\mathcal{F}_0^+ = \{E|H \in \mathcal{F}_0 : H \in D_0^+\}.$$

Remark 4. Note that as $\underline{\lambda}^\star$ is a solution of system $(\mathcal{S}_0)$ one has

$$P_0(E|H) = p^{(0)}_{E|H}, \ \forall E|H \in \mathcal{F}_0^+ \,. \tag{7}$$

If $\mathcal{F}_0^z \neq \emptyset$, setting $\mathcal{F}_1 = \mathcal{F}_0^z$, we consider the pair $(\mathcal{F}_1, \mathcal{A}_1)$. Note that, since D_0^+ cannot be empty it follows that $\mathcal{F}_1$ is a strict subset of $\mathcal{F}_0$. Then, repeating what has already been done to construct P_0, replacing index 0 by 1, we are able to define a real function P_1 on $\mathcal{E}_1 \times \mathcal{X}_1$ which is a coherent conditional probability on $\mathcal{E}_1 \times \mathcal{X}_1$, with $\mathcal{X}_1$ quasi additive w.r.t. P_1.
Consequently, if $\mathcal{F}_1^z \neq \emptyset$, we set $\mathcal{F}_2 = \mathcal{F}_1^z$ and so on. We continue in this way until we obtain $\mathcal{F}_k^z = \emptyset$ for some integer k. Therefore, we construct a strictly decreasing sequence $(\mathcal{F}_0, \mathcal{A}_0), (\mathcal{F}_1, \mathcal{A}_1), \dots (\mathcal{F}_k, \mathcal{A}_k)$. Incidentally, we also have a sequence $\{\mathcal{F}_0^+, \mathcal{F}_1^+, \dots, \mathcal{F}_k^+\}$ which is a partition of $\mathcal{F}_{J_n}$. In this way we are able to construct a finite sequence $P_0, P_1, \dots, P_k$ such that, for each $i = 0, 1, \dots, k$, P_i is a conditional probability on $\mathcal{E}_i \times \mathcal{X}_i$, with $\mathcal{X}_i$ quasi additive w.r.t. P_i. Moreover, each P_i is also coherent. Based on a reasoning as in Remark 4, for each $i = 0, 1, \dots, k$, we have

$$P_i(E|H) = p_{E|H}^{(i)} \quad , \forall E|H \in \mathcal{F}_i^+ \, . \tag{8}$$

Finally, we note that the finite sequence $\{\mathcal{E}_0, \mathcal{E}_1, \dots, \mathcal{E}_k\}$ is not increasing and given two sets $\mathcal{X}_i, \mathcal{X}_j$ in $\{\mathcal{X}_0, \mathcal{X}_1, \dots, \mathcal{X}_k\}$, with $i \neq j$, one has

$$\mathcal{X}_i \cap \mathcal{X}_j = \emptyset. \tag{9}$$

Example 1 (continued)
We set $(\mathcal{F}_0, \mathcal{A}_0) = (\mathcal{F}_{J_3}, \mathcal{A}_{J_3})$. The set $\mathcal{C}_0(\mathcal{H}_0)$ of the constituents associated with $\mathcal{F}_0$ and contained in $\mathcal{H}_0 = \{D \vee AC \vee AB\}$ is given in (1). By setting $\mathcal{P}_0 = (\frac{1}{2}, 0, \frac{1}{3})$ as a precise assessment on $\mathcal{F}_0$, it can be proved that $\mathcal{P}_0 \in \Pi_0$; moreover the system $(\mathcal{S}_0)$ associated with $(\mathcal{F}_0, \mathcal{P}_0)$ is

$$(\mathcal{S}_0) \begin{cases} \lambda_3 = \frac{1}{2}(\lambda_1 + \lambda_3 + \lambda_5 + \lambda_7) \\ \lambda_3 + \lambda_4 = 0(\lambda_3 + \lambda_4 + \lambda_5 + \lambda_6) \\ \lambda_3 + \lambda_4 = \frac{1}{3}(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4) \\ \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 = 1 \\ \lambda_h \geq 0, \ h = 1, 2, \dots, 7 \, . \end{cases}$$

Coherence of $\mathcal{P}_0$ requires that system $(\mathcal{S}_0)$ is solvable. The vector $\underline{\lambda}^* = (\lambda_h^*, h = 1, \dots, 7)$ with $\lambda_6^* = 1$ and $\lambda_h^* = 0$, if $h \neq 6$, is a solution of $(\mathcal{S}_0)$. Moreover, we have $\phi_0(D, \underline{\lambda}^*) = \lambda_1^* + \lambda_3^* + \lambda_5^* + \lambda_7^* = 0$, $\phi_0(AC, \underline{\lambda}^*) = \lambda_3^* + \lambda_4^* + \lambda_5^* + \lambda_6^* = 1$ and $\phi_0(AB, \underline{\lambda}^*) = \lambda_1^* + \lambda_2^* + \lambda_3^* + \lambda_4^* = 0$. Thus, $D_0^z = \{D, AB\}$ and $D_0^+ = \{AC\}$. Then, we set $P_0 : \mathcal{E}_0 \times \mathcal{X}_0$ as in (6), where $\mathcal{E}_0$ is the algebra generated by the constituents given in (1) and $\mathcal{X}_0 = \{AC, D \vee AC \vee AB\}$ is trivially a quasi additive class w.r.t. P_0. In particular, the value

$$P_0(E_2|H_2) = P_0(B|AC) = \frac{\lambda_3^* + \lambda_4^*}{\lambda_3^* + \lambda_4^* + \lambda_5^* + \lambda_6^*} = 0$$

is consistent with $\mathcal{A}_{J_3}$. Now, since $\mathcal{F}_0^+ = \{B|AC\}$, we set $\mathcal{F}_1 = \mathcal{F}_0 \setminus \{B|AC\}$ and we consider the pair $(\mathcal{F}_1, \mathcal{A}_1) = (\{ABC|D, C|AB\}, ([\frac{1}{2}, 1], [\frac{1}{3}, \frac{2}{3}]))$. The set $\mathcal{C}_1$ of the constituents associated with the family $\mathcal{F}_1$ is

$$\mathcal{C}_1 = \{R_0, R_2, R_4, R_5, R_6, R_7\} = \{C_1, \dots, C_6\},$$

where $C_1 = ABC^cD$, $C_2 = ABC^cD^c$, $C_3 = ABCD$, $C_4 = ABCD^c$, $C_5 = (A^c \vee B^c)D$, $C_6 = (A^c \vee B^c)D^c$. Moreover $\mathcal{H}_1 = \{D \vee AB\}$ and $C_1(\mathcal{H}_1) = \mathcal{C}_1 \setminus \{C_6\}$. We choose the sub-assessment $(\frac{1}{2}, \frac{1}{3})$ of $\mathcal{P}_0$ on $\mathcal{F}_1$ as the precise coherent assessment $\mathcal{P}_1$ on $\mathcal{F}_1$ consistent with $\mathcal{A}_1$ (we could have chosen any other coherent assessment $\mathcal{P}_1$ on $\mathcal{F}_1$ consistent with $\mathcal{A}_1$). The system $(\mathcal{S}_1)$ associated with the pair $(\mathcal{F}_1, \mathcal{P}_1)$, where $\mathcal{P}_1 = (\frac{1}{2}, \frac{1}{3})$, is

$$(\mathcal{S}_1) \begin{cases} \lambda_3 = \frac{1}{2}(\lambda_1 + \lambda_3 + \lambda_5) \\ \lambda_3 + \lambda_4 = \frac{1}{3}(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4) \\ \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 = 1 \\ \lambda_h \geq 0, \ h = 1, 2, \dots, 5 \, . \end{cases}$$

Since coherence of $\mathcal{P}_0$ requires coherence of each subvector, we have that the assessment $\mathcal{P}_1$ on $\mathcal{F}_1$ is coherent. Then, system $(\mathcal{S}_1)$ is solvable and a solution is given by $\underline{\lambda}^* = (\lambda_h^*, h = 1, \dots, 5)$ with $\lambda_1^* = \lambda_2^* = \lambda_3^* = \frac{1}{3}$ and $\lambda_4^* = \lambda_5^* = 0$. We have $\phi_1(D, \underline{\lambda}^*) = \lambda_1^* + \lambda_3^* + \lambda_5^* = \frac{2}{3}$ and $\phi_0(AB, \underline{\lambda}^*) = \lambda_1^* + \lambda_2^* + \lambda_3^* + \lambda_4^* = 1$. Thus, $D_1^z = \emptyset$ and $D_1^+ = \{D, AB\}$. Then, we set $P_1 : \mathcal{E}_1 \times \mathcal{X}_1$ as in (6) where index 0 is replaced by index 1, $\mathcal{E}_1$ is the algebra generated by the constituents $C_1, \dots, C_5$ and $\mathcal{X}_1$ is the class of conditional events $\{D, AB, D \vee AB\}$. In particular, the values

$$P_1(ABC|D) = \frac{\lambda_3^*}{\lambda_1^* + \lambda_3^* + \lambda_5^*} = \frac{\frac{1}{3}}{\frac{2}{3}} = \frac{1}{2},$$
$$P_1(C|AB) = \frac{\lambda_3^* + \lambda_4^*}{\lambda_1^* + \lambda_2^* + \lambda_3^* + \lambda_4^*} = \frac{\frac{1}{3}}{1} = \frac{1}{3}$$

are consistent with $\mathcal{A}_{J_3}$. Moreover, $\mathcal{X}_1$ is quasi additive w.r.t. P_1 ($\mathcal{X}_1$ is also additive); in particular $P_1(D|\mathcal{H}_1) + P_1(AB|\mathcal{H}_1) > 0$.

5 Extension of the probability P_i on $\mathcal{E}_i \times \mathcal{X}_i$ to P_i^0 on $\mathcal{E}_0 \times \mathcal{X}_i$

In this section, for each $i = 1, \dots, k$ (supposing $k \geq 1$), we introduce a conditional probability P_i^0 on $\mathcal{E}_0 \times \mathcal{X}_i$, with $\mathcal{X}_i$ quasi additive w.r.t. P_i^0, such that its restriction on $\mathcal{E}_i \times \mathcal{X}_i$ is the probability P_i defined above. Rather than discuss this in full generality, let us look at $i = 1$. Based on definition (6), where we have replaced index 0 by 1, for every $E|H$ with $E \in \mathcal{E}_1$, $H \in \mathcal{X}_1$ the probability P_1 is defined as follows

$$P_1(E|H) = \frac{\phi_1(EH, \underline{\delta}^\star)}{\phi_1(H, \underline{\delta}^\star)}, \tag{10}$$

where $\underline{\delta}^\star$ is a solution of the following system $(\mathcal{S}_1)$ in the unknown $\underline{\delta} = (\delta_B,\ B \in \mathcal{C}_1)$

$$(\mathcal{S}_1)\begin{cases} \sum_{B\in\mathcal{C}_1(EH)} \delta_B = p_{E|H}\cdot \sum_{B\in\mathcal{C}_1(H)} \delta_B\,, \forall E|H \in \mathcal{F}_1 \\ \sum_{B\in\mathcal{C}_1} \delta_B = \sum_{B\in\mathcal{C}_1(\mathcal{H}_1)} \delta_B = 1\,,\ \ \delta_B \geq 0\ \ \forall B \in \mathcal{C}_1. \end{cases}$$

Since $\mathcal{E}_0 \supseteq \mathcal{E}_1$, each constituent $B \in \mathcal{C}_1$ can be written as the disjunction of the constituents contained in $\mathcal{C}_0(B)$, such as

$$B = \bigvee_{C\in\mathcal{C}_0(B)} C. \tag{11}$$

We note that, as $\mathcal{C}_0$ is a partition of Ω, each set $\mathcal{C}_0(B)$ is non empty. Moreover, given two constituents B' and B'' belonging to $\mathcal{C}_1$, as $B'B'' = \emptyset$, we have

$$\mathcal{C}_0(B') \cap \mathcal{C}_0(B'') = \emptyset. \tag{12}$$

Next, adding, for each component δ_B of $\underline{\delta}$, the following constraints

$$\delta_B = \sum_{C\in\mathcal{C}_0(B)} \lambda_C,\ \ \lambda_C \geq 0,\ \ \forall C \in \mathcal{C}_1(B) \tag{13}$$

to system $(\mathcal{S}_1)$, we obtain the following system (S_1^0) in the unknown $\underline{\lambda} = (\lambda_C,\ C \in \mathcal{C}_0)$ associated with (S_1)

$$\begin{cases} \sum_{B\in\mathcal{C}_1(EH)} (\sum_{C\in\mathcal{C}_0(B)} \lambda_C) = p_{E|H}\cdot \sum_{B\in\mathcal{C}_1(H)} (\sum_{C\in\mathcal{C}_0(B)} \lambda_C), \\ \qquad\qquad \forall E|H \in \mathcal{F}_1, \\ \sum_{B\in\mathcal{C}_1} (\sum_{C\in\mathcal{C}_0(B)} \lambda_C) = \sum_{B\in\mathcal{C}_1(\mathcal{H}_1)} (\sum_{C\in\mathcal{C}_0(B)} \lambda_C) = 1, \\ \delta_B = \sum_{C\in\mathcal{C}_0(B)} \lambda_C,\ \forall B \in \mathcal{C}_1, \\ \lambda_C \geq 0, \quad \forall\, C \in \mathcal{C}_0(B)\,,\ \forall\, B \in \mathcal{C}_1. \end{cases}$$

Then, from (11) and (12) we deduce that for each solution of (S_1) there exists at least a solution (in general infinite solutions) of (S_1^0). In particular, indicating by $r_0(\cdot)$ the cardinality of $\mathcal{C}_0(\cdot)$, a solution of (S_1) could be $\underline{\lambda}^* = (\lambda_C^*,\ C \in \mathcal{C}_0)$ with

$$\lambda_C^* = \frac{\delta_C^\star}{r_0(B)}\,,\ \ \forall C \in \mathcal{C}_0(B)\,,\ \ B \in \mathcal{C}_1\,. \tag{14}$$

Therefore, choosing a solution $\underline{\lambda}^*$ of $\mathcal{S}_1^0$ we set

$$P_1^0(E|H) = \frac{\phi_0(EH,\underline{\lambda}^*)}{\phi_0(H,\underline{\lambda}^*)}, \forall E|H,\ \text{with}\ E \in \mathcal{E}_0,\ H \in \mathcal{X}_1. \tag{15}$$

Then, we have

Proposition 1. *The restriction to $\mathcal{E}_1\times\mathcal{X}_1$ of the function P_1^0 defined on $\mathcal{E}_0 \times \mathcal{X}_1$ is P_1.*

Proof. It is sufficient to prove that for every event $E \in \mathcal{E}_1$ it must be $\phi_1(E,\underline{\delta}^\star) = \phi_0(E,\underline{\lambda}^*)$. From (11) and (12) we have $C_0(E) = \bigvee_{B\in\mathcal{C}_1(E)} \left(\bigvee_{C\in\mathcal{C}_0(B)} C\right)$. Then, we obtain

$$\phi_1(E,\underline{\delta}^\star) = \sum_{B\in\mathcal{C}_1(E)} \delta_B^\star = \sum_{B\in\mathcal{C}_1(E)} \sum_{C\in\mathcal{C}_0(B)} \lambda_C^* = \sum_{C\in\mathcal{C}_0(E)} \lambda_C^* = \phi_0(E,\underline{\lambda}^*).$$

□

Finally, we have

Theorem 6. *The real function P_1^0 is a conditional probability on $\mathcal{E}_0\times\mathcal{X}_1$. Moreover, the class $\mathcal{X}_1$ is quasi-additive w.r.t. P_1^0.*

Proof. By repeating the reasoning done in the first part of the proof of Theorem 5, it can be shown that P_1^0 is a conditional probability on $\mathcal{E}_0\times\mathcal{X}_1$. In addition, we observe that quasi additive property involves only conditioning events in $\mathcal{X}_1 \subseteq \mathcal{E}_1$, where $\mathcal{X}_1$ is P_1-quasi additive. Then, from Proposition 1 it follows that $\mathcal{X}_1$ is a quasi additive class w.r.t. P_1^0. □

By repeating the reasoning above for $i = 2\ldots k$ and by setting $P_0^0 = P_0$ we are able to construct a sequence of conditional probabilities P_i^0 defined on $\mathcal{E}_0 \times \mathcal{X}_i$, $i = 0,\ldots k$, such that, for each i, $\mathcal{X}_i$ is quasi additive w.r.t. P_i^0. Furthermore, by Theorem 2, each P_i^0 is a coherent conditional probability on $\mathcal{E}_0 \times \mathcal{X}_i$.

6 A quasi additive class with all conditioning events of $\mathcal{F}$.

In this section we will construct a conditional probability $P_{01\ldots k}$ on $\mathcal{E}\times\mathcal{X}$, where $\mathcal{E} = \mathcal{E}_0$ and $\mathcal{X}$ is the union of the classes $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_k$. Therefore, each conditioning event of $\mathcal{F}$ will belong to $\mathcal{X}$. Next, we will show that $\mathcal{X}$ is a quasi additive class respect to this probability. Finally, we will see that $P_{01\ldots k}$ coincides with $\mathcal{P}$ on $\mathcal{F}$.
Let $\{P_i^0,\ i = 0,\ldots,k\}$ be the sequence of conditional probabilities introduced in the previous section, with P_i^0 defined on $\mathcal{E}_0 \times \mathcal{X}_i$. For every $E|H$ with $E \in \mathcal{E}_0$ and $H \in \mathcal{X}$, where $\mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_1 \cup \ldots \cup \mathcal{X}_k$, we set

$$P_{01\ldots k}(E|H) = P_i^0(E|H),\ \ \text{with } i \text{ such that } H \in \mathcal{X}_i \tag{16}$$

Remark 5. The previous definition of $P_{01\ldots k}$ is not ambiguous. In fact, given an event $H \in \mathcal{X}$, by relation (9) there exists a unique $\mathcal{X}_i$ in $\{\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_k\}$ such that $H \in \mathcal{X}_i$.

Then, we have

Theorem 7. *The real function $P_{01\ldots k}$ is a conditional probability on $\mathcal{E}_0 \times \mathcal{X}$. Moreover, the class $\mathcal{X}$ is quasi-additive w.r.t. $P_{01\ldots k}$.*

Proof. In order to prove that $P_{01\ldots k}$ is a conditional probability on $\mathcal{E}_0 \times \mathcal{X}$ we have to show that $P_{01\ldots k}$ satisfies properties $(i), (ii), (iii)$. Let H be a conditioning event in $\mathcal{X}$, we have that $H \in \mathcal{X}_r$ for some $r = \{0, 1, \ldots, k\}$. Since both properties (i) and (ii) are known to hold for P_r^0, from definition (15) it follows that they also hold for $P_{01\ldots k}$. Next, given three events E_1, E_2 and H, with $E_1, E_2 \in \mathcal{E}_0$ and $H, E_1H \in \mathcal{X}$, the proof that $P_{01\ldots k}$ satisfies property (iii) falls naturally into two cases.
a) If $H \in \mathcal{X}_r$ and $E_1H \in \mathcal{X}_r$ for some $r \in \{0, 1, \ldots, k\}$, then property (iii) holds for $P_{01\ldots k}$ since it holds for P_r^0.
b) If $H \in \mathcal{X}_r$ and $E_1H \in \mathcal{X}_s$, for some $r, s \in \{0, 1, \ldots, k\}$ with $r < s$, then it follows that $E_1H \notin \mathcal{X}_r$. Therefore, we have $P_r^0(E_1H|\mathcal{H}_r) = 0$. Moreover, as $E_1E_2H \subseteq E_1H$ one has $P_r^0(E_1E_2H|\mathcal{H}_r) = 0$. Thus, we obtain

$$P_{01\ldots k}(E_1E_2|H) = \frac{P_r^0(E_1E_2H|\mathcal{H}_r)}{P_r^0(H|\mathcal{H}_r)} = 0$$

and

$$P_{01\ldots k}(E_1|H) = \frac{P_r^0(E_1H|\mathcal{H}_r)}{P_r^0(H|\mathcal{H}_r)} = 0.$$

Then, property (iii), that is $P_{01\ldots k}(E_1E_2|H) = P_{01\ldots k}(E_2|E_1H)P_{01\ldots k}(E_1|H)$, is satisfied by $0 = 0$.
To prove that $\mathcal{X}$ is a quasi additive class w.r.t.$P_{01\ldots k}$ we have to show that, the conditions (i) and (ii) in (2) are satisfied by $P_{01\ldots k}$. Let H_1 and H_2 be two conditioning events in $\mathcal{X}$, we distinguish two cases:
a) $H_1, H_2 \in \mathcal{X}_r$, for some $r \in \{0, 1, \ldots, k\}$. Quasi additivity conditions (i) and (ii) in (2) hold for $P_{01\ldots k}$ since they hold for P_r^0;
b) $H_1 \in \mathcal{X}_r$ and $H_2 \in \mathcal{X}_s$ for some $r, s \in \{0, 1, \ldots, k\}$. In this case, we consider $\mathcal{H}_m$, where $m = \min\{r, s\}$. Since $\mathcal{H}_m$ is the disjunction of the conditioning events contained in D_m, it follows that $\mathcal{H}_m \in \mathcal{X}_m$. Moreover, we have

$$H_1 \vee H_2 \subseteq \bigvee_{H \in D_m} H = \mathcal{H}_m,$$

then condition (i) in (2) is satisfied by $K = \mathcal{H}_m$. Next, we have

$$P_{01\ldots k}(H_1|\mathcal{H}_m) + P_{01\ldots k}(H_2|\mathcal{H}_m) > 0.$$

In fact, if $m = r$, then $P_{01\ldots k}(H_1|\mathcal{H}_m) = P_r^0(H_1|\mathcal{H}_r) > 0$, otherwise if $m = s$, then $P_{01\ldots k}(H_2|\mathcal{H}_m) = P_s^0(H_2|\mathcal{H}_s) > 0$. We can conclude that condition (ii) in (2) is satisfied by $K = \mathcal{H}_m$. □

Remark 6. We observe that in the first part of the previous proof the further case $H \in \mathcal{X}_r$ and $E_1H \in \mathcal{X}_s$, with $r > s$, cannot be possible. In fact, if it were true, as $H \notin \mathcal{X}_s$, there would be $P_s(H|\mathcal{H}_s) = 0$. Therefore, as $E_1H \subseteq H$ we would have

$$0 = P_s^0(H|\mathcal{H}_s) \geq P_s^0(E_1H|\mathcal{H}_s) > 0,$$

which is absurd.

Finally, based on Theorem 7 and Theorem 2 we can conclude that $P_{01\ldots k}$ is a coherent probability on $\mathcal{E}_0 \times \mathcal{X}$. Moreover, given a conditional event $E|H \in \mathcal{F}_{J_n}$ as $(\mathcal{F}_0^+, \mathcal{F}_1^+, \ldots, F_k^+)$ is a partition of $\mathcal{F}_{J_n}$ there exist $i \in \{0, 1, \ldots, k\}$ such that $E|H \in \mathcal{F}_i^+$. Then, from (15) and (8), for each $i = 0, 1, \ldots, k$ it follows that

$$P_{01\ldots k}(E|H) = p_{E|H}^{(i)},\ \forall E|H \in \mathcal{F}_i^+ .$$

Therefore $P_{01\ldots k}$ is consistent with $\mathcal{A}_{J_n}$.

Remark 7. Since any restriction of a coherent conditional probability is coherent too, then the restriction of $P_{01\ldots k}$ on $\mathcal{F}_{J_n}$ is coherent. We observe that $P_{01\ldots k}$ is, in general, not unique, although we start from the same precise coherent assessment. Moreover, two obtained conditional probabilities $P_{01\ldots k}$ and $P'_{01\ldots k}$ could be defined on different sets of conditioning events.

Example 1 (continued)
We set $P_0^0 = P_0$ and, by applying (15), we extend the conditional probabilities $P_1 : \mathcal{E}_1 \times \mathcal{X}_1$ to P_1^0 on $\mathcal{E}_0 \times \mathcal{X}_1$. Then, the function $P_{01} : \mathcal{E}_0 \times \mathcal{X}$ defined as

$$P_{01}(E|H) = \begin{cases} P_0^0(E|H), \text{ if } H \in \{AC, D \vee AC \vee AB\} \\ P_1^0(E|H), \text{ if } H \in \{D, AB, D \vee AB\}, \end{cases}$$

where $\mathcal{X} = \{AC, D \vee AC \vee AB, D, AB, D \vee AB\}$, is a coherent conditional probability on $\mathcal{E}_0 \times \mathcal{X} \supseteq \mathcal{F}$ Moreover, we have that: (a) P_{01} is consistent with $\mathcal{A}_{J_3}$; (b) $P_{01}(E|H) = p_{E|H}$ for every $E|H \in \mathcal{F}_{J_3}$, that is

$$P_{01}(ABC|D) = \tfrac{1}{2},\ P_{01}(B|AC) = 0, P_{01}(C|AB) = \tfrac{1}{3}\ ;$$

(c) $\mathcal{X}$ is P_{01}-quasi additive. We observe that the class $\mathcal{X}$ is not additive (for instance $D \vee AC \notin \mathcal{X}$).

7 Further results

In this section we will give a characterization theorem of coherent precise assessment on a finite family of conditional events and a relevant way of introducing quasi additive classes.
Let $\mathcal{P}$ be a precise coherent assessment on a finite family $\mathcal{F}_{J_n}$ of conditional events. We recall that $\mathcal{P}$ is a particular interval-valued probability assessment on $\mathcal{F}_{J_n}$ where each upper bound coincides with the respective lower bound, then by setting $\mathcal{A}_{J_n} = \mathcal{P}$ and based on the results obtained in previous we are able

to construct (in a direct way) a coherent conditional probability $P_{01\ldots k}$ on $\mathcal{E}_0 \times \mathcal{X} \supseteq \mathcal{F}_{J_n}$, with $\mathcal{X}$ quasi additive w.r.t. $P_{01\ldots k}$, such that the restriction of $P_{01\ldots k}$ on $\mathcal{F}_{J_n}$ coincides with $\mathcal{P}$. Moreover, as any restriction of a coherent conditional probability is coherent too, we have

Theorem 8. *A real-valued function $\mathcal{P}$ on a finite family of conditional events $\mathcal{F}_{J_n}$ is coherent if and only if $\mathcal{P}$ can be extended as a conditional probability $P_{01\ldots k}$ on $\mathcal{E}_0 \times \mathcal{X} \supseteq \mathcal{F}_{J_n}$, with $\mathcal{X}$ quasi additive w.r.t. $P_{01\ldots k}$.*

A similar result, which also has been proved when the family of conditional events is infinite, has been given in Coletti and Scozzafava [2002] (see also Coletti and Scozzafava [1999]) by assuming $\mathcal{X}$ additive. Moreover, given a precise coherent assessment $\mathcal{P}$ on a finite family of conditional events $\mathcal{F}_{J_n}$, let $\mathcal{D}$ be the associated set of conditioning events. By the results of the previous section we can show that the cardinality of the quasi additive set $\mathcal{X} \supseteq \mathcal{D}$ w.r.t. any coherent extension $P_{01\ldots k}$ is always at most $2n$, where n denotes the cardinality of $\mathcal{D}$. We observe that for an additive set $\mathcal{X}' \supseteq \mathcal{D}$ the cardinality could be at most $2^n - 1$.

Remark 8. The problem of giving an algorithm that characterizes the whole set of conditional probabilities consistent with the initial imprecise assessment seems not easy and we do not consider it in this paper. However, we illustrate a procedure for constructing relevant cases of conditional probabilities by considering the algorithm for checking g-coherence of imprecise assessments given in Biazzo and Gilio [2000]. Based on (3), denoting by Λ_0 the set of solutions of system S_0, we introduce the following sets of conditional events

$$I_0 = \{H \in D_0 : \phi_0(H, \underline{\lambda}) = 0,\ \forall \underline{\lambda} \in \Lambda_0\}\,,$$
$$\Gamma_0 = D_0 \setminus I_0\,.$$

We observe that I_0 is equivalent to the set, denoted by the same symbol, used in the algorithm for checking g-coherence given in Biazzo and Gilio [2000]. As for any given $H \in I_0$ there exists a solution $\underline{\lambda}_H$ such that $\phi_0(H, \underline{\lambda}_H) > 0$, then (by a convex linear combination of the vectors $\underline{\lambda}_H$'s, with coefficients all positive) there exists a solution $\underline{\lambda}$ such that $\phi_0(H, \underline{\lambda}) > 0$ for all $H \in I_0$. Therefore, by recalling (5), Γ_0 is the set D_0^+ associated with the solution $\underline{\lambda}$, while in general the sets D_0^+, D_0^z associated with any other solution $\underline{\lambda}^* \neq \underline{\lambda}$ are such that $\Gamma_0 \supseteq D_0^+$ and $I_0 \subseteq D_0^z$. Then, for each $I \subseteq \Gamma_0$ and for each J such that $I \subseteq J \subseteq D_0$, the class $I \cup \{\mathcal{H}_J\}$, where $\mathcal{H}_J = \bigvee_{H \in J} H$, is a quasi additive class which can be introduced at the first step of the procedure. Similar quasi additive classes can be introduced in the other steps of the procedure; at the end, based on the union of these quasi additive classes, we obtain a conditional probability consistent with the initial imprecise assessment.

We can apply the procedure above to any subset Λ_0' of the set of solutions Λ_0 of the system S_0; in this way we can determine other quasi additive classes of conditioning events and the associated conditional probabilities consistent with the initial assessment.
Concerning the computational complexity, we observe that the procedure to construct the coherent probability $P_{01\ldots k}$ is related to the problem of the global checking of coherence of the initial assessment, which tends to become intractable when the cardinality of the starting family of conditional events increases (for an analysis of complexity on this kind of problems see e.g. Biazzo et al. [2005]). Local methods for reducing the computational difficulties have been developed in some papers (see e.g. Biazzo et al. [2003], Capotorti and Vantaggi [2002], Capotorti et al. [2003]).

Example 2. Let be given an algebra of event $\mathcal{E}_0$ and a probability P_0 on $\mathcal{E}_0$. Of course, P_0 is coherent. Moreover, let $\mathcal{X} = \{H_1, H_2, \ldots, H_n, \Omega\}$ be any subset of $\mathcal{E}_0 \setminus \{\emptyset\}$, with $P_0(H_i) > 0$, $i = 1, \ldots, n$. The probability P_0 can be extended to a conditional probability P on $\mathcal{E}_0 \times \mathcal{X}$ defined as

$$P(E|H_i) = \frac{P_0(EH_i)}{P_0(H_i)},\ \forall E \in \mathcal{E}, H_i \in \mathcal{X},$$

with $P(E|\Omega) = P_0(E)$, for every $\mathcal{E}$. We remark that: (i) $H_i \vee H_j \subseteq \Omega$ for every subset $\{i, j\}$ of $\{1, 2, \ldots, n\}$; (ii) $P(H_i|\Omega) + P(H_j|\Omega) > 0$; thus, $\mathcal{X}$ is P-quasi additive and, of course, the conditional probability P is coherent. As we can see, the property of quasi additivity is implicitly exploited in all the cases in which we construct a (coherent) conditional probability by means of ratios of unconditional probabilities.

8 Conclusions

In this paper starting from a g-coherent interval-valued probability assessment $\mathcal{A}_{J_n}$ on a finite family $\mathcal{F}_{J_n}$ we first have constructed a sequence of conditional probabilities with quasi additive classes of conditioning events. Then we have extended each probability in this sequence to obtain a new sequence of conditional probabilities with quasi additive classes of conditioning events. Moreover, we have defined a conditional probability $P_{01\ldots k}$ on $\mathcal{E} \times \mathcal{X}$, where $\mathcal{E}$ is the algebra generated by the constituents associated with $\mathcal{F}_{J_n}$ and $\mathcal{X}$ is a quasi additive class of conditioning events which contains all conditioning events of $\mathcal{F}_{J_n}$. We have shown that $P_{01\ldots k}$ is coherent and consistent with the imprecise assessment $\mathcal{A}_{J_n}$ on $\mathcal{F}_{J_n}$. Finally, we have given a characterization theorem of coherent precise assessments and a relevant way of introducing quasi additive classes.

Acknowledgements

Thanks to the reviewers for helpful comments and suggestions.

References

V. Biazzo and A. Gilio. A generalization of the fundamental theorem of de Finetti for imprecise conditional probability assessments. *International Journal of Approximate Reasoning*, 24:251–272, 2000.

V. Biazzo and A. Gilio. On the linear structure of betting criterion and the checking of coherence. *Annals of Mathematics and Artificial Intelligence*, 35: 83–106, 2002.

V. Biazzo, A. Gilio, and G. Sanfilippo. Coherence checking and propagation of lower probability bounds. *Soft Computing*, 7(5):310–320, 2003.

V. Biazzo, A. Gilio, T. Lukasiewicz, and G. Sanfilippo. Probabilistic logic under coherence: complexity and algorithms. *Ann. Math. Artif. Intell.*, 45(1-2):35–81, 2005.

A. Capotorti and B. Vantaggi. Locally strong coherence in inferential processes. *Annals of Mathematics and Artificial Intelligence*, 35:125–149, 2002.

A. Capotorti, L. Galli L., and B. Vantaggi. Locally strong coherence and inference with lower-upper probabilities. *Soft Computing*, 7(5):280–287, 2003.

G. Coletti. Coherent numerical and ordinal probabilistic assessments. *IEEE Trans. on Systems, Man, and Cybernetics*, 24(12):1747–1754, 1994.

G. Coletti and R. Scozzafava. Conditioning and inference in intelligent systems. *Soft Computing*, 3(3): 118–130, 1999.

G. Coletti and R. Scozzafava. *Probabilistic logic in a coherent setting*, volume 15 of *Trends in logics*. Kluwer Academic Publishers, Dordrecht / Boston / London, 2002.

A. Császár. Sur la structure des espace de probabilité conditionelle. *Acta Mathematica Academiae Scientiarum Hungaricae*, 6:337–361, 1955.

B. de Finetti. *Theory of Probability, vol. 1*. Wiley, Chichester, 1974.

L. E. Dubins. Finitely additive conditional probabilities, conglomerability and disintegrations. *The Annals of Probability*, 3:89–99, 1975.

A. Gilio. Classi quasi additive di eventi e coerenza di probabilità condizionate. *Rendiconti dell'Istituto di Matematica dell'Università di Trieste*, XXI(1):22–38, 1989.

A. Gilio. Algorithms for precise and imprecise conditional probability assessments. *Mathematical Models for Handling Partial Knowledge in Artificial Intelligence*, pages 231–254, 1995a.

A. Gilio. Probabilistic consistency of conditional probability bounds. In *Advances in Intelligent Computing*, volume 945 of *LNCS*, pages 200–209, Berlin Heidelberg, 1995b. (B. Bouchon-Meunier, R. R. Yager, and L. A. Zadeh, Eds.) Springer-Verlag.

A. Gilio. Probabilistic reasoning under coherence in system P. *Annals of Mathematics and Artificial Intelligence*, 34:5–34, 2002.

A. Gilio and G. Sanfilippo. Quasi conjunction and inclusion relation in probabilistic default reasoning. In Weiru Liu, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 6717 of *Lecture Notes in Computer Science*, pages 497–508. Springer Berlin / Heidelberg, 2011.

A. Gilio and F. Spezzaferri. Knowledge integration for conditional probability assessments. In D. Dubois, M. P. Wellman, B. D'Ambrosio, and P. Smets, editors, *Uncertainty in Artificial Intelligence*, pages 98–103. Morgan Kaufmann Publishers, 1992.

S. Holzer. On coherence and conditional prevision. *Boll. Un. Mat. Ital.*, 4(6):441–460, 1985.

R. Pelessoni and P. Vicig. A consistency problem for imprecise conditional probability assessments. In *Proceedings IPMU-98*, pages 1478–1485, Paris, France, 1998.

E. Regazzini. Finitely additive conditional probabilities. *Rend. Sem. Mat. Fis. Milano*, 55:69–89, 1985.

A. Rényi. On a new axiomatic theory of probability. *Acta Mathematica Hungarica*, 6:285–335, 1955.

P. Rigo. Un teorema di estensione per probabilità condizionate finitamente additive. *Atti della XXXIV Riunione Scientifica S.I.S.*, pages 27–34, 1988.

P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

P. Walley. Coherent upper and lower previsions, 1997. The Imprecise Probabilities Project.

P.M. Williams. Notes on conditional previsions. *International Journal of Approximate Reasoning*, 44 (3):366 – 383, 2007. Reprinted in a revised form of "Notes on conditional previsions", Tech. Report, 1975.

Efficient MRF Energy Minimization via Adaptive Diminishing Smoothing

Bogdan Savchynskyy[1] **Stefan Schmidt**[1] **Jörg Kappes**[2] **Christoph Schnörr**[1,2]
[1]Heidelberg Collaboratory for Image Processing, Heidelberg University, Germany,
[2]Image and Pattern Analysis Group, Heidelberg University, Germany

Abstract

We consider the linear programming relaxation of an energy minimization problem for Markov Random Fields. The dual objective of this problem can be treated as a concave and unconstrained, but non-smooth function. The idea of smoothing the objective prior to optimization was recently proposed in a series of papers. Some of them suggested the idea to decrease the amount of smoothing (so called temperature) while getting closer to the optimum. However, no theoretical substantiation was provided.

We propose an adaptive smoothing diminishing algorithm based on the duality gap between relaxed primal and dual objectives and demonstrate the efficiency of our approach with a smoothed version of Sequential Tree-Reweighted Message Passing (TRW-S) algorithm. The strategy is applicable to other algorithms as well, avoids adhoc tuning of the smoothing during iterations, and provably guarantees convergence to the optimum.

1 INTRODUCTION

We consider the problem of computing the most likely configuration x for a given graphical model, i.e. a distribution $p_{\mathcal{G}}(x;\theta) \propto \exp(-E_{\mathcal{G}}(\theta,x))$. The problem to compute the most likely labeling x (*MAP labeling problem*) amounts to minimizing the energy function[1]

$$\min_{x\in\mathcal{X}} E_{\mathcal{G}}(\theta,x) = \min_{x\in\mathcal{X}} \left\{ \sum_{v\in\mathcal{V}} \theta_{v,x_v} + \sum_{uv\in\mathcal{E}} \theta_{uv,x_{uv}} \right\}. \quad (1)$$

While problem (1) is known to be NP-hard, we will concentrate mainly on the linear programming (LP) relaxation of the problem, originally proposed by Schlesinger [16] – see [24] for a recent review.

A popular approach for solving the relaxed problem is based on maximization of its dual objective, which constitutes a lower bound for the initial objective (1). It is well-known that the dual can be treated as a concave but non-smooth unconstrained problem (see e.g. [17]). There are a number of algorithmic schemes targeting it in its original non-smooth form, e.g. [16, 24, 8, 17, 9, 4]. Some of them, namely sub-gradient algorithms [17, 9] are guaranteed to reach its solution, but are extremely slow, others – message passing (e.g. [8, 4, 11]) – typically perform faster, but may get stuck in non-optimal points, since they can be considered as (block-)coordinate ascent.

Slow convergence of sub-gradient methods and stalling of message-passing ones are caused by the non-smoothness of the objective. Hence the idea of applying smoothing was proposed in a series of recent papers [6, 25, 7, 15]. However to reach a good approximate solution of the initial non-smooth objective the smoothing degree should be selected properly. There are two approaches how this can be done:
(i) **Precision oriented smoothing** approach [7, 15], following the fundamental paper [12]. The smoothing degree depends on the precision to be achieved and is selected and held fixed (or it changes only slightly) during the algorithm iterations.
(ii) Iterative or **diminishing smoothing** approach follows the idea to decrease the smoothing degree as the current iterate gets closer to the optimum. So far we are not aware of any algorithm employing this scenario for the problem (1), although the basic idea is mentioned in several papers, cf. [6, 25].

The idea of the diminishing smoothing corresponds to the following intuition: as long the iterate is far from the optimum one can use a coarse (very smooth) approximation of the objective, since it allows for faster optimization. The closer the iterate is to the optimum, the finer (and thus less smooth and computationally more costly) the approximation is required to guarantee a certain solution accuracy.

[1]We describe our notation in Section 2.

Both, precision oriented and diminishing smoothing approaches operate with the *smoothing gap* i.e. the largest difference between an objective function and its smooth approximation. The smoothing gap however can be controlled only indirectly via some *smoothing parameter*. In its turn the smoothing parameter can be set based on a **worst-case** analysis describing its influence on the smoothing gap. Alternatively this influence can be estimated **adaptively**, as discussed in [15].

For our diminishing smoothing approach some method for estimating the upper bound for the LP relaxation of the energy minimization problem is required. There are several recent papers addressing this issue [26, 18, 15] and we found that the method proposed in [15] fits our needs best. Our algorithm itself was inspired by [13], but in contrast we consider a purely dual optimization instead of the primal-dual one (to avoid massive memory allocation for the primal problem). Furthermore we focus on efficiency of the algorithm on average rather than in the worst-case by exploiting the specific structure of our problem.

We demonstrate the efficiency of our approach with a smooth version of the Sequential Tree-Reweighted Message Passing (S-TRWS) algorithm evaluated on the Middlebury database [20] and a series of computer generated examples. This algorithm, proposed for fixed smoothing in [11] and partially analyzed in [25] estimates the tree-reweighted free energy and can be considered as a smooth version of TRW-S [8]. Contrary to the fixed smoothing we consider a diminishing sequence of the smoothing parameter and show that this solves the LP relaxation of the energy minimization problem (1) up to a given precision. We compare our approach to the S-TRWS algorithm with a final precision oriented smoothing and a Nesterov first-order smoothing based optimization scheme proposed in [7] and [15] and show that it significantly outperforms all of them.

Without loss of generality, we concentrate on the common special case of grid-structured models here (as the benchmark [20] deals with this setting), but our results apply to non-grid-structured graphs as well.

Contribution. The contribution of this paper is four-fold:

1. We describe a diminishing smoothing algorithm for the LP relaxation of the problem (1), based on the duality gap of the relaxed problem.

2. We analyze the method of adaptive selection of the smoothing parameter proposed in [15] and show that in general it does not guarantee attainment of the prescribed precision (as implicitly stated in [15]). We propose a modification of the method, which eliminates this disadvantage, while preserves its efficiency.

3. Additionally, we propose a method for obtaining the sound stopping criterion for algorithms which maximize the *tree-reweighted free energy* [22] via minimizing its dual [5, 3, 11]. To this end we provide a method for computing a primal bound from a current dual iterate. This bound approaches the optimum of the *tree-reweighted free energy* together with the dual iterates.

4. We evaluate the S-TRWS algorithm with different smoothing selection strategies as a method for solving the LP relaxation of the MRF energy minimization problem (1).

Paper Organization. Section 2 briefly describes basic notions. Section 3 is devoted to estimation of upper (primal) bounds for the dual objective and its smooth approximation. Section 4 describes possible strategies of smoothing selection. And finally Sections 5 and 6 contain results of experimental evaluation and conclusions respectively.

2 PRELIMINARIES

We denote by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ the undirected graph with nodes $\mathcal{V}$ and edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Associated with the nodes $v \in \mathcal{V}$ are finite *sets of labels* $\mathcal{X}_v$. The Cartesian product set $\mathcal{X} = \otimes_{v\in\mathcal{V}}\mathcal{X}_v$ is the space of possible *labelings* $x \in \mathcal{X}$, i.e. a labeling is a collection $(x_v\colon v \in \mathcal{V})$ of labels. We use the short-hand notation x_{uv} for a pair of labels (x_u, x_v), and $\mathcal{X}_{uv}$ for the corresponding space $\mathcal{X}_u \times \mathcal{X}_v$. The cost functions $\theta_v : \mathcal{X}_v \to \mathbb{R},\ v \in \mathcal{V}$, and $\theta_{uv} : \mathcal{X}_{uv} \to \mathbb{R},\ uv \in \mathcal{E}$, are referred to as *unary* and *pairwise potentials*, respectively, with the complete set of potentials associated with the graph denoted as θ.

As mentioned in the introduction, our objective function is the dual of a linear programming relaxation of the energy minimization problem (1). As is widely known (see e.g. [24]), the linear programming relaxation of the problem (1) reads

$$\min_{\mu} \sum_{v\in\mathcal{V}} \sum_{x_v\in\mathcal{X}_v} \theta_{v,x_v}\mu_{v,x_v} + \sum_{uv\in\mathcal{E}} \sum_{x_{uv}\in\mathcal{X}_{uv}} \theta_{uv,x_{uv}}\mu_{uv,x_{uv}}$$
$$\text{s.t.} \begin{cases} \sum_{x_v\in\mathcal{V}} \mu_{v,x_v} = 1,\ v \in \mathcal{V}\,, \\ \sum_{x_v\in\mathcal{V}} \mu_{uv,x_{uv}} = \mu_{u,x_u},\ x_u \in \mathcal{X}_u,\ uv \in \mathcal{E}\,, \\ \sum_{x_u\in\mathcal{V}} \mu_{uv,x_{uv}} = \mu_{v,x_v},\ x_v \in \mathcal{X}_v,\ uv \in \mathcal{E}\,, \\ \mu_{uv,x_{uv}} \geq 0,\ x_{uv} \in \mathcal{X}_{uv},\ uv \in \mathcal{E}\,. \end{cases} \tag{2}$$

Such a formulation in terms of relaxed indicator vectors μ is commonly referred to as the overcomplete representation of a discrete graphical model, in which the *local polytope* $\mathcal{L}(\mathcal{G})$, as defined by the constraints of (2), represents a simpler outer bound to the marginal polytope, i.e. the convex hull of all indicator vectors of allowable labelings, cf. [23]. The linear program (2) can be written in short as $\min_{\mu\in\mathcal{L}(\mathcal{G})} \langle\theta, \mu\rangle$. Abusing notation we will denote $\langle\theta, \mu\rangle$ by $E(\mu)$.

It turns out that a dual problem for (2) has a simpler structure and less variables, hence its optimization is easier than

the direct optimization of (2). There are several ways how the dual problem can be derived and represented. We will follow *a dual decomposition approach* [21, 9], since it allows for efficient optimization of real-sized problems (see a comparison in [19]). For the sake of simplicity we consider a special case typical for grid graphs.

Let $\mathcal{G}^i = (\mathcal{V}^i, \mathcal{E}^i),\ i \in \{1, 2\}$, be two *acyclic* subgraphs of the *master graph* $\mathcal{G}$. Let $\mathcal{V}^1 = \mathcal{V}^2 = \mathcal{V}$, $\mathcal{E}^1 \bigcup \mathcal{E}^2 = \mathcal{E}$ and $\mathcal{E}^1 \bigcap \mathcal{E}^2 = \emptyset$ (e.g. if $\mathcal{G}$ is a grid graph, $\mathcal{E}^1$ may contain all horizontal edges of $\mathcal{G}$ and $\mathcal{E}^2$ all vertical ones). Then the overall energy becomes the sum of the energies corresponding to these subgraphs,

$$\begin{aligned} E_{\mathcal{G}}(\theta, x) &= \sum_{i=1}^{2} \sum_{v \in \mathcal{V}^i} \theta^i_{v,x_v} + \sum_{uv \in \mathcal{E}^i} \theta^i_{uv,x_{uv}} \\ &= E_{\mathcal{G}^1}(\theta^1, x) + E_{\mathcal{G}^2}(\theta^2, x), \end{aligned} \tag{3}$$

provided $\theta^i_{uv} = \theta_{uv},\ uv \in \mathcal{E}^i,\ i = 1, 2$ and $\theta^1_{v,x_v} + \theta^2_{v,x_v} = \theta_{v,x_v},\ \forall v \in \mathcal{V}, x_v \in \mathcal{X}_v$. The latter condition can be represented in a parametric way as $\theta^1_{v,x_v} = \frac{\theta_{v,x_v}}{2} + \lambda_{v,x_v}$ and $\theta^2_{v,x_v} = \frac{\theta_{v,x_v}}{2} - \lambda_{v,x_v}, v \in \mathcal{V},\ x_v \in \mathcal{X}_v$, where $\lambda_{v,x_v} \in \mathbb{R}$. The dual space $\{\lambda_{v,x_v} \in \mathbb{R} | v \in \mathcal{V},\ x_v \in \mathcal{X}_v\}$ will be denoted as Λ. Thus we consider θ^i as a function of λ and obviously have

$$\min_{x \in \mathcal{X}} E_{\mathcal{G}}(\theta, x) \geq \max_{\lambda \in \Lambda} \sum_{i=1}^{2} \min_{x \in \mathcal{X}} E_{\mathcal{G}^i}(\theta^i(\lambda), x). \tag{4}$$

It has been shown [10] that all decompositions into acyclic subgraphs which cover the original graph yield the same lower bound as the one in (4). Furthermore, it is established that this bound equals the solution of the linear programming problem (2).

Hence, in order to solve the relaxed problem (2) we will maximize its dual, the concave but non-smooth function U defined by the maximand of (4), i.e.

$$U(\lambda) = \sum_{i=1}^{2} \min_{x \in \mathcal{X}} E_{\mathcal{G}^i}(\theta^i(\lambda), x)\,. \tag{5}$$

We note that the function $U(\lambda)$ can be easily evaluated by dynamic programming for any acyclic graph $\mathcal{G}^i$.

Smoothed Dual Objective and its Primal Counterpart. There is a number of approaches to maximize (5). Along with [6, 25, 7, 15] we propose to use a smoothing technique [12] to obtain a smooth approximation of $U(\lambda)$. This would allow both to utilize powerful smooth optimization algorithms (e.g. [15]) and to obtain guarantees for convergence to the optimum of the smooth (and as we will see strictly convex) function for efficient (block-)coordinate descent methods similar to TRW-S [8].

The way we construct the smooth approximation is dictated by the need to preserve the efficiency of the decomposition. The smooth function should be easily computable over subgraphs $\mathcal{G}^i$, like its non-smooth counterpart U.

Each summand in (5) may be written as the inner product of the potential vector θ^i and a suitable binary indicator vector $\phi(x)$, i.e.

$$U^i(\lambda) := \min_{x \in \mathcal{X}} E_{\mathcal{G}^i}(\theta^i(\lambda), x) = \min_{x \in \mathcal{X}} \left\langle \theta^i(\lambda), \phi(x) \right\rangle. \tag{6}$$

As this minimum is non-smooth in λ, the same holds for the objective function (5). To obtain a smooth approximation, we replace $\min$ (or rather $-\max$) by the well-known log-sum-exp (or soft-max) function (cf. [12]), yielding

$$\tilde{U}^i_\rho(\lambda) = -\rho \log \sum_{x \in \mathcal{X}} \exp \left\langle -\theta^i(\lambda)/\rho, \phi(x) \right\rangle \tag{7}$$

with *smoothing parameter* ρ. The resulting function $\tilde{U}^i_\rho$ uniformly approximates U^i, that is

$$\tilde{U}^i_\rho(\lambda) + \rho \log |\mathcal{X}| \geq U^i(\lambda) \geq \tilde{U}^i_\rho(\lambda). \tag{8}$$

This directly implies

$$\tilde{U}_\rho(\lambda) + 2\rho \log |\mathcal{X}| \geq U(\lambda) \geq \tilde{U}_\rho(\lambda) \tag{9}$$

for $\tilde{U}_\rho := \sum_{i=1}^{2} \tilde{U}^i_\rho$.

Please note that for acyclic graphs $\mathcal{G}^i$ evaluating $\tilde{U}^i_\rho$ (and thus $\tilde{U}_\rho$) is as easy as U^i, and can be done by dynamic programming.

Inequality (9) provides a possibility to exchange optimization of the non-smooth function U with the optimization of its smooth approximation $\tilde{U}_\rho$. Selecting the smoothing parameter ρ small enough (see Section 4.1) or properly decreasing it during optimization (Section 4.2), one can guarantee attainment of the optimum of U with a given precision.

Additionally for $\rho = 1$ the function $\tilde{U}_\rho$ has another important meaning: it is a dual function for *the tree-reweighted free energy* [22] and can be used to estimate approximate marginals [5, 3, 11].

Let us denote by N_{uv} the number of subgraphs containing the edge $uv \in \mathcal{E}$ and by N_v the number of subgraphs containing the node $v \in \mathcal{V}$. In our special case $N_{uv} = 1$ and $N_v = 2$.

Theorem 1 ([22, 25]) *The tree-reweighted free energy*

$$\begin{aligned} \tilde{E}_\rho(\mu) := \langle \theta, \mu \rangle - \rho \Big(& \sum_{v \in \mathcal{V}} \sum_{x_v \in \mathcal{X}_v} N_v \mu_{v,x_v} \log \mu_{v,x_v} \\ & + \sum_{uv \in \mathcal{E}} \sum_{x_{uv} \in \mathcal{X}_{uv}} N_{uv} \mu_{uv,x_{uv}} \log \frac{\mu_{uv,x_{uv}}}{\mu_{v,x_v} \mu_{u,x_u}} \Big) \end{aligned} \tag{10}$$

defined over $\mathcal{L}(\mathcal{G})$ is a Lagrange dual for $\tilde{U}_\rho(\lambda)$. And since strict duality holds $\min_{\mu \in \mathcal{L}(\mathcal{G})} \tilde{E}_\rho(\mu) = \max_{\lambda \in \Lambda} \tilde{U}_\rho(\lambda)$.

S-TRWS Algorithm There are a number of approaches to optimize $\tilde{U}_\rho$ (see e.g. [11, 15, 5]). In this paper we evaluate the S-TRWS algorithm originally devoted to computation of the function $\tilde{U}_\rho$ with a fixed value $\rho = 1$ [11]. Contrary to this we will operate ρ in a way to obtain a good (up to a given precision) approximation for the maximum of the non-smooth function U. We selected this algorithm because its original, non-smoothed analogue [8] is one of the most efficient schemes for optimizing U. The second advantage of S-TRWS is that it is guaranteed to converge to the optimum of $\tilde{U}_\rho$ for any fixed ρ contrary to TRW-S, which does not converge to the optimum of U in general.

We introduce vector of "marginals" $\nu^i_\rho(\lambda) \in \mathbb{R}^{\sum_{w\in\mathcal{V}\cup\mathcal{E}}|\mathcal{X}_w|},\ i \in \{1,2\}$ by

$$\nu^i_\rho(\lambda)_{w,x_w} := \frac{\sum\limits_{x'\in\mathcal{X},x'_w=x_w} \exp\left\langle -\theta^i(\lambda)/\rho, \phi(x')\right\rangle}{\exp(-\tilde{U}^i_\rho(\lambda)/\rho)}\,. \quad (11)$$

It is well-known (see e.g. [15, Lemma 1]), that the coordinates of the gradient $\nabla\tilde{U}_\rho \in \mathbb{R}^{\sum_{v\in\mathcal{V}}|\mathcal{X}_v|}$ are equal to

$$\nabla\tilde{U}_\rho(\lambda)_{v,x_v} = \nu^1_\rho(\lambda)_{v,x_v} - \nu^2_\rho(\lambda)_{v,x_v}\,. \quad (12)$$

The S-TRWS algorithm is a specially organized coordinate descent procedure applied to the smoothed function $\tilde{U}_\rho$. It sequentially updates variables λ according to the rule

$$\lambda_{v,x_v} := \lambda_{v,x_v} + (\log\nu^1_\rho(\lambda)_{v,x_v} - \log\nu^2_\rho(\lambda)_{v,x_v})/2\,. \quad (13)$$

Applying this rule corresponds to a coordinate ascent step w.r.t. variable λ_{v,x_v}. The power of this algorithm stems from an extremely efficient way of processing updates (13). Namely, updates are done sequentially, but the computational cost to perform all these $\sum_{v\in\mathcal{V}}|\mathcal{X}_v|$ updates is basically the same as just computing gradient coordinates $\nu^i_\rho(\lambda)$ (see [8] for details), which makes it superior to any general-purpose gradient based algorithm. Due to (12), it is easy to see that an optimum of $\tilde{U}_\rho$ is a fix-point of this algorithm. We refer to [11] for a proof of convergence. We also note that the implementation of this algorithm is far from trivial because of the necessity to exponentiate very large numbers, corresponding to small values of ρ in (11). We coped with this difficulty efficiently based on the idea provided in [12, p.140].

3 UPPER BOUNDS ESTIMATION

There are at least two reasons why it is important to be able to reconstruct a sequence of primal feasible points $\mu^t \in \mathcal{L}(\mathcal{G})$ converging to a solution of the problem (2) or (10), from a sequence λ^t converging to the maximum of the corresponding dual objective. The first reason is the values μ^t themselves: for $\rho = 1$ they can be considered as approximate marginal probabilities [27, 22]. The second reason is an upper bound $E(\mu^t)$ or $\tilde{E}_\rho(\mu^t)$, which can be used for sound stopping criteria [15]. We will use the upper bound to construct a diminishing sequence of smoothing parameters ρ to get an ε-approximation for a maximum of the non-smooth function U via optimization of its smooth approximation $\tilde{U}_\rho$, see Section 4.2.

Primal LP Objective Estimation The simplest (and thus quite popular) way of computing upper bounds for maximum of (5) is rounding schemes [14]. It allows to estimate an integer solution $x \in \mathcal{X}$, from a dual iterate λ. However, the energy $E(\theta, x)$ corresponding to the integer solution x may not achieve the minimal value of objective (2) even in case the labeling was estimated based on the optimal value λ^* delivering the maximum of the dual objective U.

Alternatively we use the primal LP-bound construction introduced in [15]. We denote by $\mathbb{R}_+(\mathcal{G}) = \mathbb{R}_+^{\sum_{v\in\mathcal{V}}|\mathcal{X}_v|+\sum_{uv\in\mathcal{E}}|\mathcal{X}_{uv}|}$ a nonnegative linear half-space containing the local polytope $\mathcal{L}(\mathcal{G})$. Additionally we use the notation $\mathcal{L}_{uv}(\mu_u,\mu_v) \subset \mathbb{R}_+^{|\mathcal{X}_{uv}|}$ for the domain of pairwise marginals $\mu_{uv},\ uv \in \mathcal{X}_{uv}$ satisfying the constraints of (2) for given unary marginals μ_u, μ_v. We use the following reformulation of [15, Thm. 2]:

Theorem 2 *Let $\hat{\mu}_\rho(\lambda) \in \mathbb{R}_+(\mathcal{G})$ be computed as*

$$\hat{\mu}_\rho(\lambda)_{v,x_v} = \frac{\nu^1_\rho(\lambda)_{v,x_v} + \nu^2_\rho(\lambda)_{v,x_v}}{2}\,, \quad (14)$$

$$\hat{\mu}_\rho(\lambda)_{uv,x_{uv}} = \operatorname*{arg\,min}_{\mu_{uv,x_{uv}}\in\mathcal{L}_{uv}(\hat{\mu}_\rho(\lambda)_u,\hat{\mu}_\rho(\lambda)_v)} \langle\theta_{uv},\mu_{uv}\rangle\,. \quad (15)$$

Then for $\tilde{\lambda} = \arg\max_{\lambda\in\Lambda}\tilde{U}_\rho(\lambda)$ holds

$$U(\tilde{\lambda}) + 2\rho\log|\mathcal{X}| \geq E(\hat{\mu}_\rho(\tilde{\lambda})) \geq U(\lambda),\ \lambda\in\Lambda\,. \quad (16)$$

The inequality (16) basically states that the bound $E(\hat{\mu}_\rho(\tilde{\lambda}))$ becomes exact as ρ vanishes.

As noted in [15], evaluating (15) constitutes a *transportation problem* – a well-studied class in linear programing. Since the size of each individual problem is small, these can be easily solved by any appropriate method of linear programming. We found a specialization of the simplex method [1] to be quite efficient in our case.

Tree-Reweighted Free Energy Estimation. The upper bound for the minimum of the tree-reweighted energy (10) can be estimated in the same manner as in Theorem 2.

Theorem 3 *Let $\tilde{\mu}_\rho(\lambda) \in \mathbb{R}_+(\mathcal{G})$ be computed as*

$$\tilde{\mu}_\rho(\lambda)_{v,x_v} = \frac{\nu^1_\rho(\lambda)_{v,x_v} + \nu^2_\rho(\lambda)_{v,x_v}}{2}\,,$$

$$\tilde{\mu}_\rho(\lambda)_{uv,x_{uv}} = \operatorname*{arg\,min}_{\mu_{uv,x_{uv}}\in\mathcal{L}_{uv}(\tilde{\mu}_\rho(\lambda)_u,\tilde{\mu}_\rho(\lambda)_v)} \langle\theta_{uv},\mu_{uv}\rangle - \rho\sum_{x_{uv}\in\mathcal{X}_{uv}} N_{uv}\mu_{uv,x_{uv}}\log\frac{\mu_{uv,x_{uv}}}{\mu_{v,x_v}\mu_{u,x_u}}\,. \quad (17)$$

Then $\tilde{E}_\rho(\tilde{\mu}_\rho(\lambda)) \geq U(\lambda')$ *for any* $\lambda, \lambda' \in \Lambda$ *and for* $\tilde{\lambda} = \arg\max_{\lambda \in \Lambda} \tilde{U}_\rho(\lambda)$ *holds* $U(\tilde{\lambda}) = \tilde{E}_\rho(\tilde{\mu}_\rho(\tilde{\lambda}))$.

Please note that evaluating (17) constitutes a small-sized convex problem (an entropy maximization kind problem) with linear constraints. It can be efficiently solved e.g. by interior point methods.

As we already mentioned, Theorem 3 defines a method for estimating an upper bound for the optimal value of the tree-reweighted energy (10), which can be utilized to constructing a sound stopping criterion based on the duality gap for algorithms maximizing $\tilde{U}_\rho$.

4 SMOOTHING SCHEDULING

In this section we will discuss how to estimate an optimum of U via optimization of its smooth approximation $\tilde{U}_\rho$, i.e. how the smoothing parameter ρ can be selected or/and updated to guarantee the achievement of an ε-approximate solution of U.

We consider two approaches of controlling the smoothing gap $\Delta = \max_{\lambda \in \Lambda} U(\lambda) - \tilde{U}_\rho(\lambda)$: precision oriented and diminishing smoothing. Each of them will be considered in connection to both worst-case and adaptive estimation of the smoothing parameter ρ for a given value of Δ.

4.1 PRECISION-ORIENTED SMOOTHING

Precision-Oriented Worst-Case Smoothing Parameter Selection. The simplest way to select the parameter ρ is to fix it small enough. Let λ^* and $\tilde{\lambda}$ denote optimal points of U and $\tilde{U}_\rho$ respectively. By applying the inequality (9) to $\tilde{\lambda}$ and λ^* and taking into account that $\tilde{U}_\rho(\tilde{\lambda}) \geq \tilde{U}_\rho(\lambda^*)$ and $U(\lambda^*) \geq U(\tilde{\lambda})$ we obtain

$$\tilde{U}_\rho(\tilde{\lambda}) + \underbrace{2\rho \log |\mathcal{X}|}_{=:\Delta(\rho)} \geq U(\lambda^*) \geq \tilde{U}_\rho(\tilde{\lambda})\,. \tag{18}$$

Hence if the inequality

$$\Delta(\rho) < \varepsilon \tag{19}$$

holds, the optimal value $U(\lambda^*)$ can be estimated with precision ε by optimizing $\tilde{U}_\rho$. Inequality (19) can be transformed into a range of allowed values for the smoothing parameter: $0 \leq \rho < \frac{\varepsilon}{2 \log |\mathcal{X}|}$. However selecting an optimal value ρ from this range is not straightforward and depends on the algorithm used for the optimization of $\tilde{U}_\rho$. Lemma 3 in [15] claims that if the convergence rate of the optimization algorithm reads $O(\frac{1}{\rho})$ as a function of ρ, then the optimal value of ρ equals to a half of the upper bound of the range, i.e.

$$\rho = \frac{\varepsilon}{4 \log |\mathcal{X}|}\,. \tag{20}$$

However since little is known about the convergence rate of the considered S-TRWS algorithm, we employ this lemma as a hypothesis.

Precision-Oriented Adaptive Smoothing Parameter Selection. Indeed, as mentioned in [15], the estimation $2\rho \log |\mathcal{X}|$ for the smoothing gap is typically loose. Hence it was proposed by the authors to perform a local estimation the smoothing gap as

$$\hat{\Delta}^t(\rho) := \max_t U(\lambda^t) - \tilde{U}_\rho(\lambda^t) \tag{21}$$

for iterates λ^t of the optimization algorithm. The value ρ has to be selected such that $\hat{\Delta}^t(\rho) < \varepsilon$, even more, using Lemma 3 in [15] it was proposed to select ρ to satisfy

$$\hat{\Delta}^t(\rho) < \frac{\varepsilon}{2}\,. \tag{22}$$

Since this simple (and seemingly quite natural!) approach guarantees fulfillment of (19) only locally with respect to the sequence $\{\lambda^t\}$, it does not guarantee attainment of

(i) precision ε in general. Latter can be demonstrated by an example, when the difference $U(\tilde{\lambda}) - \tilde{U}_\rho(\tilde{\lambda})$ in an optimum $\tilde{\lambda}$ of the smoothed function $\tilde{U}_\rho$ is smaller than the one $U(\lambda^*) - \tilde{U}_\rho(\lambda^*)$ in the optimum λ^* of the non-smooth one, U, and $U(\lambda^*) - \tilde{U}_\rho(\tilde{\lambda}) > \varepsilon$. Starting an algorithm optimizing $\tilde{U}_\rho$ in the point $\tilde{\lambda}$ with

$$\hat{\Delta}^t(\rho) = U(\tilde{\lambda}) - \tilde{U}_\rho(\tilde{\lambda}) < \frac{\varepsilon}{2} \tag{23}$$

makes it stall in the starting point, since (a) the point $\tilde{\lambda}$ is already an optimum of $\tilde{U}_\rho$ and the algorithm can not get better with respect to $\tilde{U}_\rho$; (b) changing of the smoothing ρ is not performed since (22) holds.

(ii) the stopping condition $E(\mu_\rho(\lambda^t)) - U(\lambda^t) \leq \varepsilon$ even if (22) holds together with

$$U(\lambda^*) - \tilde{U}_\rho(\tilde{\lambda}) \leq \frac{\varepsilon}{2}\,. \tag{24}$$

Condition (24) guarantees only the attainment of the accuracy ε for the dual objective U, but not for the duality gap.

Even though we did not encounter such unfortunate cases in practice, there is a general cure for these. Due to Theorem 3 we can estimate the Lagrange dual function $\tilde{E}_\rho$ for $\tilde{U}_\rho$ and decrease ρ by a factor $\eta > 1$ together with restarting the algorithm from the current point as soon as

$$\tilde{E}_\rho(\tilde{\mu}_\rho(\lambda^t)) - \tilde{U}_\rho(\lambda^t) > \frac{\varepsilon}{2} \tag{25}$$

is not satisfied. In other words, one has to select the smoothing parameter ρ such that both conditions (22) and (25) hold. We prove this in the supplementary material because of the space limitations.

Algorithm 1 Generic diminishing smoothing algorithm.

- Given: target solution accuracy ε
- Initialize: $\lambda^0 \in \Lambda$, $\rho^0 > 0$, $E^0_{\min} = E(\hat{\mu}_{\rho^0}(\lambda^0))$.
- Iterate (for $t \geq 0$):
  1. **If** $E^t_{\min} - U(\lambda^t) < \varepsilon$ **Exit and return** λ^t.
  2. $\rho^{t+1} := \min\left\{\rho^t, F(E^t_{\min}, \tilde{U}_{\rho^t}(\lambda^t))\right\}$.
  3. $\lambda^{t+1} := \psi[\tilde{U}_{\rho^{t+1}}](\lambda^t)$.
  4. $E^{t+1}_{\min} := \min\{E^t_{\min}, E(\hat{\mu}_{\rho^{t+1}}(\lambda^{t+1}))\}$.
  5. $t := t + 1$. **Goto** 1.

4.2 DIMINISHING SMOOTHING

Instead of using a fixed value of ρ satisfying (19) or keeping it rarely changed when estimation $\hat{\Delta}(\rho)$ of the smoothing gap does not satisfy (22), we will construct a diminishing ρ-sequence depending on the duality gap. This idea is based on the assumption that maximization of $\tilde{U}_\rho$ is more efficient for larger values of ρ.

Let us consider a generic algorithm - Algorithm 1 - for the diminishing smoothing. Step 1 checks the stopping condition based on the duality gap. Minimization in steps 2 and 4 is required to ensure monotone decreasing of ρ and to keep the best upper bound $E^t_{\min}$ so far. A (for the moment) unknown mapping $F\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ determines how the smoothing parameter ρ should be updated. The mapping $\psi[\tilde{U}_\rho]$ corresponds to one or more iterates of the S-TRWS algorithm (see (11) and (13)) applied to the function $\tilde{U}_\rho$.

Our goal is to determine such mappings F, which at least guarantee convergence of Algorithm 1 iterates to the optimum of the non-smooth function U.

Definition 1 *A mapping $\psi[\tilde{U}_\cdot]\colon \mathbb{R}_+ \times \Lambda \to \Lambda$ is called* an optimizing mapping *if*
(i) for any $\rho > 0$ and any $\lambda^0 \in \Lambda$ the iterative process $\lambda^{t+1} = \psi[\tilde{U}_\rho](\lambda^t)$ converges to the set $\Lambda^*(\tilde{U}_\rho)$ of optimal solutions of $\tilde{U}_\rho$;
(ii) $\tilde{U}_\rho(\psi[\tilde{U}_\rho](\lambda)) \geq \tilde{U}_\rho(\lambda)$, moreover for $\lambda \notin \Lambda^*(\tilde{U}_\rho)$ holds $\tilde{U}_\rho(\psi[\tilde{U}_\rho](\lambda)) > \tilde{U}_\rho(\lambda)$,
(iii) $\psi[\tilde{U}_\rho](\lambda)$ is continuous w.r.t. ρ for any $\lambda \in \Lambda$.

Due to continuity of $\exp, \sum$ and $\log$, the iterates of the S-TRWS algorithm depend continuously on ρ. Theorem 6 in [25] proves that every iteration of the algorithm strictly increases the value of $\tilde{U}_\rho$ for a fixed ρ; the convergence of the S-TRWS algorithm to the optimum of the function $\tilde{U}_\rho$ for any fixed ρ is proved in [11]. Hence one or more iterations of the algorithm satisfy the conditions of Def. 1. We explain the practical importance of using more than a single iteration of the algorithm at the end of this paragraph.

Theorem 4 *Let $\psi[\tilde{U}_\cdot]\colon \mathbb{R}_+ \times \Lambda \to \Lambda$ be an optimizing mapping and let the mapping $F\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ satisfy the condition*

$$\Delta\left(F(E^t_{\min}, \tilde{U}_{\rho^t}(\lambda^t))\right) \leq \frac{E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t)}{2} \qquad (26)$$

where equality holds only when $E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t) = 0$. Then iterates λ^t of Algorithm 1 converge to the optimum of the function U.

Proof. From the first property of an optimizing mapping (see Def. 1) follows that it suffices to prove that $\rho^t \xrightarrow{t\to\infty} 0$ and $\rho^t > 0$, $\forall t$. Since $E(\mu) > \tilde{U}_\rho(\lambda)$ for any $\mu, \lambda \notin \Lambda^*(U)$ and $\rho \geq 0$, from step 2 follows that $\rho^t > 0$, $\forall t$. Sequence ρ^t is monotone and bounded by 0 from below, thus it converges to some $\rho^* \geq 0$. We will prove that $\rho^* = 0$. Let $\rho^* \neq 0$. From (26), which is strict while $E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t) \neq 0$, follows that $\tilde{U}_{\rho^t}(\lambda^t) + \Delta(\rho^{t+1}) < E(\hat{\mu}_\rho(\lambda^t)) - \Delta(\rho^{t+1})$. Taking the limit as t tends to infinity, denoting $\tilde{\lambda} = \arg\min_{\lambda\in\Lambda} \tilde{U}_{\rho^*}(\lambda)$ and using continuity in ρ and λ of $U(\lambda)$, $\tilde{U}_\rho(\lambda)$, $E(\hat{\mu}_\rho(\lambda))$ and ψ we obtain $\tilde{U}_{\rho^*}(\tilde{\lambda}) + \Delta(\rho^*) < E(\hat{\mu}_{\rho^*}(\tilde{\lambda})) - \Delta(\rho^*)$. Since $\Delta(\rho) = 2\log|\mathcal{X}|$, from (16) follows $E(\hat{\mu}_{\rho^*}(\tilde{\lambda})) - \Delta(\rho^*) \leq U(\tilde{\lambda})$ and thus $\tilde{U}_{\rho^*}(\tilde{\lambda}) + \Delta(\rho^*) < U(\tilde{\lambda})$. From (9) follows the contradiction, which implies $\rho^* = 0$. □

As follows from the proof nothing prevents to improve the upper bound $E^t_{\min}$ by using any available method of its estimation align with the one provided by Theorem 2. In our experiments we use rounding schemes [14], which sometimes allows to get better estimation of $E^t_{\min}$ and hence speed-up Algorithm 1.

Number of cycles of the S-TRWS algorithm, used for computing the optimizing mapping ψ, should be carefully selected because of the fact, that performing 1 cycle of the S-TRWS without prior change of the smoothing parameter is computationally 2 times cheaper than performing it with the change. This is caused by peculiarities of the S-TRWS algorithm. Namely, to compute values $\nu^i_\rho(\lambda)_{v,x_v}$ (see (11)) it has to perform both so-called *forward* and *backward* moves of a dynamic programming algorithm. And if after a given run no changes where made to smoothing, the result of the forward (backward) move do not need to be recomputed. It means, that computing n cycles of the S-TRWS algorithm with a *constant* smoothing costs only $n + 1$ oracle calls (additional one oracle call for the initial forward move) instead of $2n$.

Diminishing Worst-Case Smoothing Parameter Selection. Theorem 4 implies that for some $\gamma > 1$

$$F(E^t_{\min}, \tilde{U}_{\rho^t}(\lambda^t)) = \Delta^{-1}\left(\frac{E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t)}{2\gamma}\right), \qquad (27)$$

at the step 2 of Algorithm 1. Here γ is some constant (which indeed can differ from iteration to iteration)

and $\Delta^{-1}(\cdot)$ denotes the inverse mapping to $\Delta(\cdot)$. Since $\Delta(\rho) = 2\rho \log |\mathcal{X}|$ it reads $\Delta^{-1}(A) = \frac{A}{2 \log |\mathcal{X}|}$. Hence

$$F(E^t_{\min}, \tilde{U}_{\rho^t}(\lambda^t)) = \frac{E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t)}{4\gamma \log |\mathcal{X}|}\,,\ \gamma > 1\,, \quad (28)$$

and Algorithm 1 is defined up to a parameter γ. We refer to Section 5 for experimental evaluation of this parameter.

Diminishing Adaptive Smoothing Parameter Selection. As we already mentioned the estimate $2\rho \log |\mathcal{X}|$ for the smoothing gap is often too loose in practice, which can potentially slow down optimization algorithms. In this section we show how the adaptive estimate (21) can be used together with an appropriate approximation of the inverse mapping Δ^{-1} to speed up the algorithm. Additionally we will use a procedure similar to the one presented in (25) to prevent the algorithm from stalling.

The method is based on a local approximation of $\hat{\Delta}^t(\rho)$ near the current point λ with an affine function $\hat{\Delta}^t(\rho) := \delta \cdot \rho + \alpha$, where $\delta = \frac{\partial \hat{\Delta}^t}{\partial \rho} = \frac{\partial (U - \tilde{U}_\rho)}{\partial \rho} = -\frac{\partial \tilde{U}_\rho}{\partial \rho}$ and assigning $\hat{\Delta}^{-1}(d) := (d - \alpha)/\delta$. Hence step 2 of Algorithm 1 takes the form

$$\delta^t := -\frac{\partial \tilde{U}_{\rho^t}}{\partial \rho}(\lambda^t), \quad \alpha^t := \hat{\Delta}^t - \delta^t \rho^t, \quad (29)$$

$$\rho^{t+1} := \min\left\{\rho^t, \left(\frac{E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t)}{2\delta^t \gamma} - \frac{\alpha^t}{\delta^t}\right)\right\}. \quad (30)$$

Parameter α^t of this approximation depends on the estimate $\hat{\Delta}^t$ defined by (21).

To avoid stalling of the algorithm one should decrease ρ as soon as

$$\tilde{E}_\rho(\mu_\rho(\lambda^t)) - \tilde{U}_\rho(\lambda^t) \le \frac{E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t)}{2\gamma}\,. \quad (31)$$

The proof for that is based on comparing (23) and (25) to (27) and (31). Note that they become equivalent when $\varepsilon = (E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t))/\gamma$.

We sum up the above considerations in Algorithm 2. This algorithm additionally depends on the parameter η defining how much the smoothing parameter ρ should be decreased. In our experiments we use $\eta = 2$, but we observed the inequality (31) being fulfilled only for γ close to 1 (which we found to be non-optimal) and thus the value η does not influence results of our experiments.

It remains only to note that the derivative $\frac{\partial \tilde{U}_{\rho^t}(\lambda^t)}{\partial \rho}$ can be computed efficiently, since it is given by a ready-to-apply formula in the following proposition.

Proposition 1 *Let* $\nu^i_\rho(\lambda) \in \mathbb{R}^{\sum_{w \in \mathcal{V} \cup \mathcal{E}} |\mathcal{X}_w|}$ *be defined by (11). Then*

$$\frac{\partial \tilde{U}_\rho(\lambda)}{\partial \rho} = \frac{\tilde{U}_\rho(\lambda)}{\rho} - \frac{1}{\rho} \sum_{i=1}^{2} \sum_{w \in \mathcal{V}^i \cup \mathcal{E}^i} \sum_{x_w \in \mathcal{X}_w} \theta^i(\lambda)_{w,x_w} \nu^i_\rho(\lambda)_{w,x_w}\,.$$

Algorithm 2 Adaptive diminishing smoothing (ADS*al*)

- Given: a prescribed solution accuracy ε.
- Initialize: $\lambda^0 \in \Lambda$, $\rho^0 > 0$, $\gamma > 1$, $\eta > 1$, $E^0_{\min} = E(\hat{\mu}_{\rho^0}(\lambda^0))$.
- Iterate (for $t \ge 0$):
  1. **If** $E^t_{\min} - U(\lambda^t) < \varepsilon$ **Exit and return** λ^t.
  2. $\delta^t := -\frac{\partial \tilde{U}_{\rho^t}}{\partial \rho}(\lambda^t)$, $\alpha^t := \hat{\Delta}^t - \delta^t \rho^t$.
  3. $\rho^{t+1} := \min\left\{\rho^t, \left(\frac{E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t)}{2\delta^t \gamma} - \frac{\alpha^t}{\delta^t}\right)\right\}$.
  4. **If** $\tilde{E}_\rho(\tilde{\mu}_\rho(\lambda^t)) - \tilde{U}_\rho(\lambda^t) \le \frac{E^t_{\min} - \tilde{U}_{\rho^t}(\lambda^t)}{2\gamma}$ **then** $\rho^{t+1} := \frac{\rho^{t+1}}{\eta}$.
  5. $\lambda^{t+1} := \psi[\tilde{U}_{\rho^{t+1}}](\lambda^t)$.
  6. $E^{t+1}_{\min} := \min\{E^t_{\min}, E(\hat{\mu}_{\rho^{t+1}} \lambda^{t+1})\}$.
  7. $t := t + 1$. **Goto** 1.

5 EXPERIMENTAL EVALUATION

For an experimental evaluation we used datasets from the Middlebury MRF benchmark [20] (datasets *tsukuba, venus, family*) and a computer generated grid model of size 256×256 with 4 labels. The unary and pairwise factors of this model were randomly selected uniformly in the interval $[0, 1]$. This dataset will be denoted as *artificial*. Complete results of all our experiments are present in the supplementary material. In the main body of the paper we present only subset of plots and a summary Table 5 to save space.

We consider the following four smoothing-based algorithms described in the paper: (i) adaptive diminishing smoothing algorithm (**A-DSal**) – Algorithm 2; (ii) diminishing worst-case smoothing (**WC-DSal**) is defined by Algorithm 1, where the mapping F is specified by (28); (iii) precision oriented adaptive smoothing (**A-STRWS**), where the smoothing degree is controlled by (21), (22) and (25); (iv) precision oriented worst-case smoothing (**WC-STRWS**), where the smoothing degree is fixed and given by (20). In all cases we use the S-TRWS algorithm as an optimizing mapping.

We compare our approaches also to the original **TRW-S** code [8] and to the accelerated first-order Nesterov optimization scheme [15] (**NEST**), for which an implementation was kindly provided by the authors.

Additionally to the upper $\Delta(\rho)$-bound computed due to

Table 1: Results of algorithms evaluation. *Req. OC 0.1%* and *Req. OC 1* – number of oracle calls required to achieve the relative precision 0.1% and the absolute precision 1 respectively; *primal/dual bound*–the best upper/lower bounds achieved during the iterations: about 8000-10000 oracle calls for NEST, TRW-S and other methods. In case the best primal bound is an integer one we print it without decimal point. Please note that A-DSal turned out to be the most efficient solver for all datasets except *family* We did not mark "dual bounds winner" for *tsukuba* dataset, because in this case all algorithms except TRW-S were terminated as soon as the duality gap dropped below 1.

	Algorithm	A-DSal	WC-DSal	A-STRWS	WC-STRWS	NEST	TRWS
tsukuba	req. OC 0.1%	52	47	45	33	576	266
	req. OC 1	**107**	151	405	789	6244	>10000
	primal bound	**369218**	369218	369218	369218	369218	369252
	dual bound	369217.38	369217.12	369217.99	369217.99	369217.99	369217.58
venus	req. OC 0.1%	**111**	131	123	129	1746	266
	primal bound	**3047993.47**	3048546.82	3048411	3048534.23	3050376	3048098
	dual bound	**3047965.20**	3047920.27	3047936.20	3047934.25	3047938.84	3047929.95
family	req. OC 0.1%	>8516	>10006	>8013	>8001	>9023	**3012**
	primal bound	186636	184927	185144.02	185142.87	6136365	**184825**
	dual bound	184769.13	184742.11	184735.80	184734.96	145396.26	**184788.00**
artificial	req. OC 0.1%	**514**	>10004	639	>8001	1515	>10000
	primal bound	**56785.86**	56838.03	56956.08	57258.31	56815.01	81118
	dual bound	56779.98	56777.10	56764.87	56724.52	**56780.24**	56720.56

Theorem 2 we also estimate a so-called *integer* upper bound, which corresponds to integer values of variables μ in (2). The latter bound turns to be often better at the beginning of the optimization process, but contrary to the non-integer one it does not posses a crucial property of convergence to the optimum of the relaxed problem (2).

Since the performance of the precision-oriented smoothing algorithms (A-STRWS, WC-STRWS and NEST) depends on the target precision we run each algorithm on each dataset twice: with target *relative* precision 0.1% and with target *absolute* precision 1. Since potentials of all considered problem from the Middlebury benchmark are integers, getting an integer upper bound within the latter precision would mean solving the corresponding problem exactly in its original, non-relaxed formulation (1). Indeed it has happen for the *tsukuba* dataset – see below.

We use the notion of *oracle calls* for measuring the speed of the algorithms to eliminate the influence of different implementations. One oracle call corresponds to a single (either forward or backward) move of the S-TRWS algorithm. It requires approximately 1.5 seconds for the *tsukuba* dataset, 3.5 s for *venus*, 4.5 s for the *family* and less than 0.5 s for the *artificial* dataset, on a 2.5GHz machine. Such an oracle call approximately corresponds to the oracle call used in NEST, it is 5 times slower than a single (either forward or backward) move of the original non-smooth TRWS algorithm in our implementation and 10 times slower than the original implementation by Kolmogorov [8] for general-form pairwise potentials. The non-smooth TRWS is faster because of the evaluation of the exp-operation needed for smoothing. Preliminary experiments show that simply switching to GPU would already reduce this operation's cost to that of a simple multiplication operation. Hence we ignore the mentioned time difference and consider a single move of the TRW-S algorithm as an oracle call. Additionally we count computations needed to estimate a primal solution $\hat{\mu}_\rho(\lambda^t)$ according to (14)-(15) as an oracle call, since the time it requires is close to that of a single oracle call.

For different values of γ and different numbers of inner S-TRWS iterations we measured the number of oracle calls of the A-DSal and WC-DSal algorithms to attain precision 1%. We found $\gamma = 4$ and 3 inner iterations (4 oracle calls) in the optimizing mapping ψ to be optimal for both A-DSal and WC-DSal. Indeed we found the algorithms quite insensitive to the value of γ, unlike for instance the step-size selection in subgradient-based schemes. Our experiments show similar performance for the interval $2 \leq \gamma \leq 5$.

Tsukuba dataset. Fig. 1(bottom left) This dataset seems to be the easiest among investigated. The relative precision 0.1% was achieved by all algorithms within few dozens of oracle calls. Moreover, all algorithms except TRW-S attained the optimum of the non-relaxed problem (1). Number of oracle calls for that, i.e. to get the duality gap less than 1 is the smallest for A-DSal. Other algorithms, described in the paper perform not much worse and indeed much faster than NEST.

Venus dataset. Fig. 1 (top left) Contrary to *tsukuba* there remains a big ($\gg 1$) relaxation gap between obtained integer solutions and the lower (dual) bound. Simultaneously the upper (primal) bound for the relaxed LP problem demonstrates a fast convergence to the lower one, which makes A-DSal quite efficient. Other algorithms described

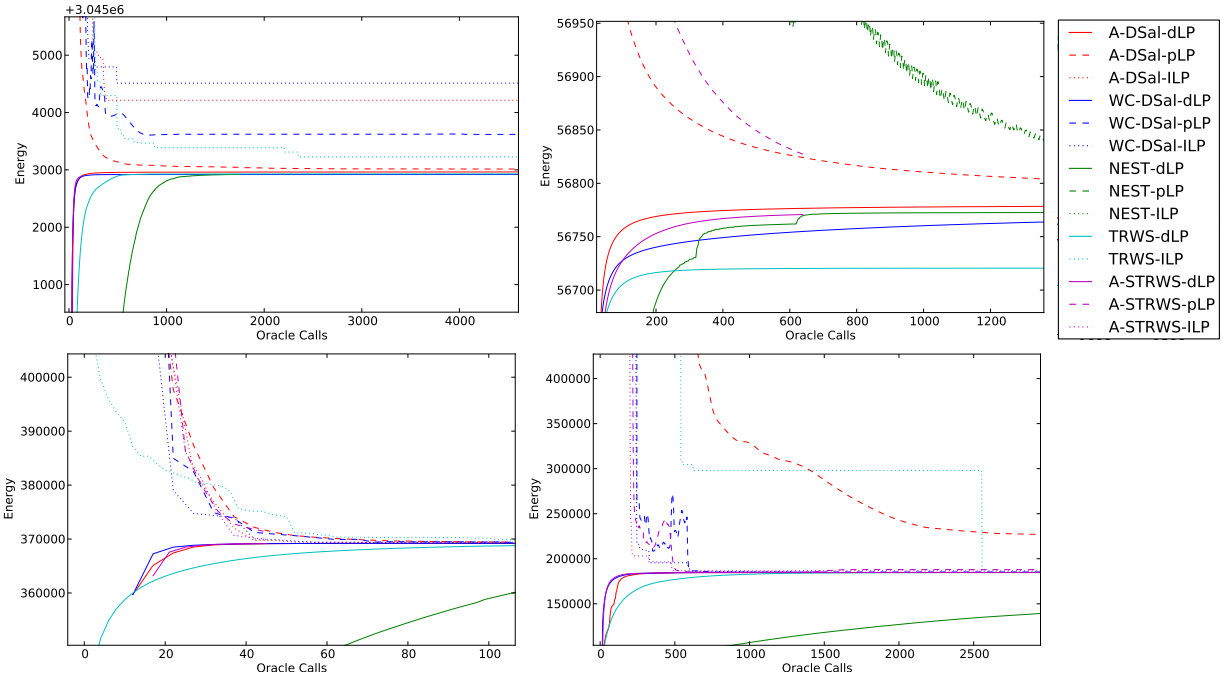


Figure 1: *Venus* (top left), *artificial* (top right), *tsukuba* (bottom left) and *family* (bottom right) datasets. Precision-oriented smoothing algorithms A-STRWS, WC_STRWS and NEST were run with a target precision 0.1%. The curves show primal (upper) (*pLP*) and dual (lower) (*dLP*) LP bounds, as well as the best integer primal bound (*ILP*) achieved. Colors correspond to different algorithms – see legend. For all datasets except *tsukua* the A-DSal demonstrates one of the best convergence rates. For the *family* the primal LP bound of A-DSal shows a very slow convergence, which makes the algorithm significantly less efficient.

in the paper show significantly worse performance. TRW-S is a bit slower at the beginning, but at the end shows the second good result after A-DSal in terms of both upper and lower bounds.

Family dataset. Fig. 1(bottom right) This dataset seems to be quite difficult for dual methods (methods considered in the paper and NEST, TRW-S) because the pairwise factors contain infinite (very large) numbers. This makes dual variables λ badly conditioned, i.e. a small change in their values corresponds to a big change of the estimated primal objective. This is why the estimations of the upper (primal) bound constructed according to Theorem 2 converge very slow and thus the smoothing parameter, selected according to the duality gap estimation, takes too large values. This influences A-DSal at most because it uses additionally adaptive smoothing parameter selection, which makes ρ even larger. However even for this setting the lower bound attained by A-DSal outperforms lower bound of all other approaches except TRW-S (see Table 5), which does not suffer from bad primal estimates.

Artificial dataset. Fig. 1 (top right) This dataset targets the situations when there is no local evidence in the data. This could be a typical setting when the inference is used in a loop of a learning algorithm, e.g. the structural SVM [2]. At the beginning of the learning process the potentials to be learned could take nearly random values, which makes the problem harder. This dataset clearly shows advantage of methods with an adaptive selection of the smoothing parameter: only those methods attained the precision 0.1%. The A-SDal again shows the best performance, whereas TRW-S – the worst one.

Summary. As our experiments show, the proposed adaptive diminishing smoothing algorithm (A-DSal) is competitive to or even outperforms the state-of-the-art methods, except in the case when the pairwise factors contain very large values. This issue has to be considered in future work.

6 CONCLUSIONS

We proposed an adaptive diminishing smoothing optimization algorithm for an LP relaxation of the energy minimization problem. The algorithm enjoys provable and fast convergence to the optimum, and often outperforms even one of the fastest state-of-the-art methods – TRWS, but contrary to that does never get stuck in non-optimal points.

Acknowledgement. This work has been supported by the German Research Foundation (DFG) within the program Spatio-/Temporal Graphical Models and Applications in Image Analysis, grant GRK 1653.

References

[1] M. S. Bazaraa and J. J. Jarvis. *Linear Programming and Network Flows*. Wiley, 1977.

[2] V. Franc and B. Savchynskyy. Discriminative learning of max-sum classifiers. *Journal of Machine Learning Research*, pages 67–104, Jan 2008.

[3] A. Globerson and T. Jaakkola. Convergent propagation algorithms via oriented trees. In *UAI*, 2007.

[4] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, 2007.

[5] J. Jancsary and G. Matz. Convergent decomposition solvers for tree-reweighted free energies. In *14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[6] J. K. Johnson, D. Malioutov, and A. S. Willsky. Lagrangian relaxation for MAP estimation in graphical models. In *45th Ann. Allerton Conf. on Comm., Control and Comp.*, 2007.

[7] V. Jojic, S. Gould, and D. Koller. Accelerated dual decomposition for MAP inference. In *ICML*, pages 503–510, 2010.

[8] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. on PAMI*, 28(10):1568–1583, 2006.

[9] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.

[10] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Trans. on PAMI*, 33(3):531 –552, march 2011.

[11] T. Meltzer, A. Globerson, and Y. Weiss. Convergent message passing algorithms - a unifying view. In *UAI 09*, 2009.

[12] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Prog.*, A(103):127–152, 2004.

[13] Y. Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM J. on Optimization*, 16:235–249, May 2005.

[14] P. Ravikumar, A. Agarwal, and M. Wainwright. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *JMLR*, 11:1043–1080, 2010.

[15] B. Savchynskyy, J. Kappes, S. Schmidt, and C. Schnörr. A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling. In *CVPR 2011*, 2011.

[16] M. Schlesinger. Syntactic analysis of two-dimensional visual signals in the presence of noise. *Kibernetika*, (4):113–130, 1976.

[17] M. Schlesinger and V. Giginyak. Solution to structural recognition (max,+)-problems by their equivalent transformations. In 2 Parts. *Control Systems and Computers*, (1-2), 2007.

[18] M. Schlesinger, E. Vodolazskiy, and N. Lopatka. Stop condition for subgradient minimization in dual relaxed (max,+) problem. In *EMMCVPR*, 2011.

[19] S. Schmidt, B. Savchynskyy, J. H. Kappes, and C. Schnörr. Evaluation of a first-order primal-dual algorithm for MRF energy minimization. In *EMMCVPR*, pages 89–103. Springer, 2011.

[20] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE Trans. PAMI.*, 30:1068–1080, June 2008.

[21] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on (hyper)trees: message passing and linear programming approaches. *IEEE Trans. on Inf. Th.*, 51(11), 2005.

[22] M. Wainwright, T. Jaakkola, and A. Willsky. A new class of upper bounds on the log partition function. *IEEE Trans. on Information Theory*, 51:2313–2335, 2005.

[23] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, 2008.

[24] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Trans. on PAMI*, 29(7), July 2007.

[25] T. Werner. Revisiting the decomposition approach to inference in exponential families and graphical models. Technical report, CMP, Czech TU, 2009.

[26] T. Werner. How to compute primal solution from dual one in MAP inference in MRF? *Control Systems and Computers*, (2), March-April 2011.

[27] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Trans. on Inf. Th.*, (7):2282–2312, 2005.

Predicting the behavior of interacting humans by fusing data from multiple sources

Erik J. Schlicht[1], **Ritchie Lee**[2], **David H. Wolpert**[3,4], **Mykel J. Kochenderfer**[1], **and Brendan Tracey**[5]
[1] Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA 02420
[2] Carnegie Mellon Silicon Valley, NASA Ames Research Park, Moffett Field, CA 94035
[3] Information Sciences Group, MS B256, Los Alamos National Laboratory, Los Alamos, NM 87545
[4] Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501
[5] Stanford University, Palo Alto, CA 94304

Abstract

Multi-fidelity methods combine inexpensive low-fidelity simulations with costly but high-fidelity simulations to produce an accurate model of a system of interest at minimal cost. They have proven useful in modeling physical systems and have been applied to engineering problems such as wing-design optimization. During human-in-the-loop experimentation, it has become increasingly common to use online platforms, like Mechanical Turk, to run low-fidelity experiments to gather human performance data in an efficient manner. One concern with these experiments is that the results obtained from the online environment generalize poorly to the actual domain of interest. To address this limitation, we extend traditional multi-fidelity approaches to allow us to combine fewer data points from high-fidelity human-in-the-loop experiments with plentiful but less accurate data from low-fidelity experiments to produce accurate models of how humans interact. We present both model-based and model-free methods, and summarize the predictive performance of each method under different conditions.

1 Introduction

The benefit of using high-fidelity simulations of a system of interest is that it produces results that closely match observations of the real-world system. Ideally, algorithmic design optimization would be performed using such accurate models. However, the cost of high-fidelity simulation (computational or financial) often prohibits running more than a few simulations during the optimization process. The high-fidelity models built from such a small number of high-fidelity data points tend not to generalize well. One approach to overcoming this limitation is to use multi-fidelity optimization, where both low-fidelity and high-fidelity data are used together during the optimization process (Robinson et al., 2006).

When the system of interest involves humans, traditionally human-in-the-loop (HITL) experimentation has been used to build models for design optimization. A common approach in HITL experimentation is to make the simulation environment match the true environment as closely as possible (Aponso et al., 2009). A major disadvantage of this approach is that designing a good high-fidelity HITL simulation requires substantial human and technological resources. Experimentation requires highly trained participants who need to physically visit the test location. As a consequence, collecting large amounts of high-fidelity data under different conditions is often infeasible.

In many cases, one could cheaply implement a lower fidelity simulation environment using test subjects who are not as extensively trained, allowing for the collection of large amounts of less accurate data. Indeed, this is the idea behind computational testbeds like Mechanical Turk (Kittur et al., 2008; Paolacci et al., 2010). The goal of this paper is to extend multi-fidelity concepts to HITL experimentation, allowing for the combination of plentiful low-fidelity data with more sparse high-fidelity data. This approach results in an inexpensive model of interacting humans that generalizes well to real-world scenarios.

In this paper, we begin by outlining the self-separation scenario used as a testbed to study multi-fidelity models of interacting humans. In Section 3 we introduce the multi-fidelity methods used in this paper. In Section 4, we present model-free approaches to multi-fidelity prediction, and in Section 5 we present model-

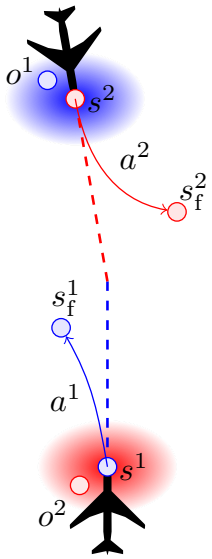
based methods. Results, summarized in Section 6, show the benefit of incorporating low-fidelity data to make high-fidelity predictions and that model-based methods out-perform model-free methods. Section 7 concludes with a summary of our findings and future directions for investigation.

2 Modeling Interacting Humans

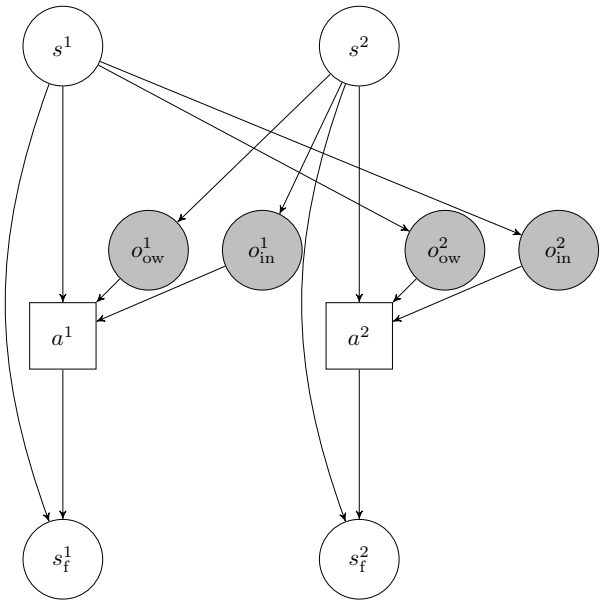
In order to ground our discussion of multi-fidelity methods in a concrete scenario, we will begin by presenting our testbed for studying the behavior of interacting humans. We model the interaction of two pilots whose aircraft are on a near-collision course. Both pilots want to avoid collision (Fig. 1a), but they also want to maintain their heading. For simplicity, we model the pilots as making a single decision in the encounter. Their decisions are based on their beliefs about the other aircraft's position and velocity, their degree of preference for maintaining course, and their prediction of the other pilot's decision.

We use a combination of Bayesian networks and game-theoretic concepts (Lee and Wolpert, 2012) to model this scenario. The structure of the model is shown in Fig. 1b. The state of aircraft i is a four-dimensional vector representing the horizontal position and velocity information. The initial distribution over states is described in Section 3. The action a^i taken by the pilot of aircraft i represents the angular change in heading, away from their current heading. Each pilot observes the state of the other aircraft based on their view out the window o^i_{ow} and their instrumentation o^i_{in}. Each pilot has a weight w^i in their utility function reflecting their relative desire to avoid collision versus maintaining heading. We consider the situation where each player knows their own type (i.e., the value of the weight term in their utility function) in addition to the type of the other player (Myerson, 1997).

As described in this section, the pilots chose their actions based on their observations and utility function. After both pilots select an action, those actions are executed for 5 s, bringing the aircraft to their final states s^i_{f}. Without loss of generality we describe the scenario from the perspective of one of the pilots, Player 1, with the other aircraft corresponding to Player 2. The model assumptions, detailed below, are symmetric between players. It should be noted that the approaches described in this paper are not limited to cases in which there are only two humans interacting, although scenarios with more humans would increase the amount of data required to make accurate predictions.



(a) Example geometry with model variables depicted.



(b) Model structure. Shaded circles are observable variables, white circles are unobservable variables, squares are decision nodes, and arrows represent conditional relationships between variables.

Figure 1: Visual self-separation game.

2.1 Visual inference

Player 1 infers the state of the intruder based on visual information out the window and instrumentation. Player 1 is not able to exactly infer the state of Player 2 as humans are not able to perfectly estimate position and velocity based on visual information (Graf et al., 2005; Schlicht and Schrater, 2007), and the instruments also have noise associated with their measurements (Fig. 1b).

We model the out-the-window visual observation as Gaussian with a mean of the true state of the intruder aircraft and covariance matrix given by $\text{diag}(900\,\text{ft}, 900\,\text{ft}, 318\,\text{ft/s}, 318\,\text{ft/s})$. The instrument observation is Gaussian with a mean of the true state of the intruder with covariance given by $\text{diag}(600\,\text{ft}, 600\,\text{ft}, 318\,\text{ft/s}, 318\,\text{ft/s})$. The out-the-window velocity uncertainties are derived from the visual psychophysics literature (Weis et al., 2002; Graf et al., 2005).

2.2 Decision-making

Theoretical models of multi-agent interaction offer a framework for formalizing decision-making of interacting humans. Past research has focused on game-theoretic approaches (Myerson, 1997) and their graphical representations (Koller and Milch, 2003; Lee and Wolpert, 2012) where agents make a single decision given a particular depth of reasoning between opponents. Such approaches have been successfully used for predicting human decision-making in competitive tasks (Camerer, 2003; Wright and Leyton-Brown, 2010, 2012).

In cases when it is important to capture how people reason across time, sequential models of human interaction can be used (Littman, 1994). Such models can reflect either cooperative settings where agents are attempting to maximize mutual utility (Amato et al., 2009) or competitive contexts where agents are attempting to maximize their own reward (Doshi and Gmytrasiewicz, 2005, 2006). Graphical forms of sequential decision processes have also been developed (Zeng and Xiang, 2010).

Since humans often reason over a relatively short time horizon (Sellitto et al., 2010), especially in sudden, stressful scenarios, we model our interacting pilots scenario as a one-shot game. Like Lee and Wolpert (2012), we use level-k relaxed strategies as a model for human decision-making. A level-0 strategy chooses actions uniformly at random. A level-k strategy assumes that the other players adopt a level-$(k-1)$ strategy (Costa-Gomes et al., 2001). In our work, we assume $k = 1$ because humans tend to use shallow depth-of-reasoning (Wright and Leyton-Brown, 2010), and increasing $k > 1$ had little effect on the predicted joint-decisions for our scenario.

The focus on bounded rationality stems from observing the limitations of human decision making. Humans are unable to evaluate the probability of all outcomes with sufficient precision and often make decisions based on adequacy rather than by finding the true optimum (Simon, 1956, 1982; Caplin et al., 2011). Because decision-makers lack the ability and resources to arrive at the optimal solution, they instead apply their reasoning only after having greatly simplified the choices available. This type of sampling-based, bounded-rational view of perceptual processing has been used in computational models of human visual tracking performance (Vul et al., 2010) and has been shown to predict human decision-making (Vul et al., 2009).

To align our model with a bounded-rational view of human performance, we assume that Player 1 makes decisions based on m' sampled intruder locations and m candidate actions sampled from a uniform distribution over ± 1 radian. In other words,

$$o^{(1)}, \ldots, o^{(m')} \sim p(o^1_{\text{ow}} \mid s^2) p(o^1_{\text{in}} \mid s^2) \quad (1)$$

$$a^{(1)}, \ldots, a^{(m)} \sim \mathcal{U}(-1, 1) \quad (2)$$

Player 1 selects the sampled action that maximizes the expected utility over the m' sampled states of the other aircraft:

$$a = \arg\max_i \sum_j w^1 d(s^1_{\text{f}|a^{(i)}}, s^2_{\text{f}|o^{(j)}}) - (1 - w^1)|a^{(i)}| \quad (3)$$

In the equation above,

- $s^1_{\text{f}|a^{(i)}}$ is the final state of Player 1 after performing action $a^{(i)}$,
- $s^2_{\text{f}|o^{(j)}}$ is the expected final state of Player 2 given observation $o^{(j)}$ and assuming a random heading change,
- $d(\cdot, \cdot)$ represents Euclidean distance, and
- w^1 is the relative desire of Player 1 to avoid the intruder versus maintaining their current heading.

The first term in the utility function rewards distance between aircraft, and the second term penalizes the magnitude of the heading change. Fig. 2 shows how joint-actions change across different w^1 and w^2 combinations. When weights are relatively low (e.g., $w^1 = 0.80$, $w^2 = 0.80$), joint-actions are centered around $(0, 0)$, indicating pilots tend to make very small heading change maneuvers. Conversely, when weights are relatively high (e.g., $w^1 = 0.98$, $w^2 = 0.98$), pilots tend to make large heading change maneuvers.

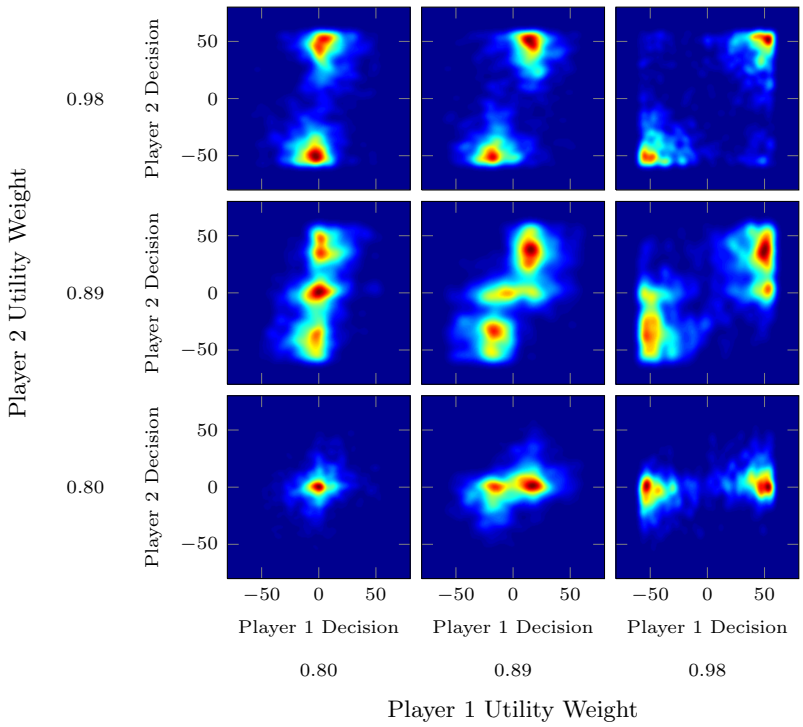


Figure 2: Influence of utility weights on joint-decision densities. Red regions depict areas with high probability joint-decisions (i.e., heading change, in degrees), while blue regions represent low probability joint-decisions.

2.3 Executing Actions

After choosing an appropriate action, the pilots then execute the maneuver. Although it is known that humans have uncertainty associated with their movements (Harris and Wolpert, 1998), we simulate the case in which pilots perfectly execute the action selected by level-k relaxed strategies, since uncertainty in action execution is not a focus of this study.

Recall that the action of the player is the change in heading angle of the aircraft, away from the current heading. From the initial state and the commanded heading change of the aircraft, the final state is obtained by simulating the aircraft for 5 seconds assuming point-mass kinematics, no acceleration, and instantaneous heading change.

3 Multi-fidelity Prediction

There are two ways one could distinguish a low-fidelity and high-fidelity model of this encounter. The first is using "low-fidelity humans" versus "high-fidelity humans," where low-fidelity humans would represent people who have been trained in flight simulators but who are not commercial pilots. The second way is to have both a low-fidelity and high-fidelity simulation environment, say having low-fidelity environment being a flight simulator that does not perfectly match the conditions of an actual commercial jet. For this paper, our low-fidelity simulators are the same as the high-fidelity simulation except there is no instrument panel. The low-fidelity simulation has the o_{in} nodes removed in Fig. 1b.

We used separate training and testing encounters, all sampled with Player 2 being in Player 1's field of view, and a heading such that a collision occurs if an avoidance maneuver is not performed (dashed lines in Fig. 1a). Training encounters were initialized with Player 2 randomly approaching at $-45°$, $0°$, and $+45$. Test encounters were initialized with Player 2 randomly approaching at $-22.5°$ and $+22.5°$. Initial headings for both train and test encounters were sampled from a Gaussian distribution with a mean around a heading direction that would result in a collision with a standard deviation of $5°$.

The goal of multi-fidelity prediction is to combine the data in the high and low-fidelity training encounters to predict the joint-decisions of pilots in the high-fidelity game. The next two sections describe different approaches to achieving this type of prediction, and the notation is summarized in Table 1.

Table 1: Notation used for multi-fidelity prediction

	Description
A	Observed joint-actions (a^1, a^2)
$\hat{A}$	Predicted joint-actions $(\hat{a}^1, \hat{a}^2)$
S	Encounter geometry (s^1, s^2)
N	Number of encounters
w	Joint utility-weights (w^1, w^2)
z	Regression weight
$(.)^{tr}$	Superscript indicating training encounters
$(.)^{te}$	Superscript indicating test encounters
$(.)_l$	Subscript indicating low-fidelity game
$(.)_h$	Subscript indicating high-fidelity game

4 Model-Free Prediction Approaches

A model-free approach is one where we do not use any knowledge of the underlying game or the decision-making process. This section discusses a traditional model-free approach to predicting joint-actions as well as a model-free multi-fidelity method.

4.1 Locally Weighted, High-fidelity

The simplest approach to model-free prediction is to use the state and joint-action information from the high-fidelity training data to predict the joint-actions given the states in the testing data. Using only the high-fidelity training data, we trained a high-fidelity predictor R_h that predicts joint-actions A_h^{tr} given the training encounters S_h^{tr}. Then, the test encounters were used as inputs to the predictor, to predict the joint-actions at the new states $\hat{A}^{te} = R_h(S^{te})$. The regression model we considered was locally weighted (LW) regression, where the predicted high-fidelity joint-decisions for a particular test encounter is the weighted combination of high-fidelity joint-decisions from the training encounters. The weights are determined by the distance between the training encounters and the test encounter:

$$\hat{A}_i^{te} = \sum_j^{N_h^{tr}} z_{i,j} A_{h,j}^{tr} \tag{4}$$

$$z_{i,j} = \frac{e^{-d_{i,j}}}{\sum_j e^{-d_{i,j}}} \tag{5}$$

where $d_{i,j}$ is the standardized Euclidean distance between the state in the ith testing encounter and the jth training encounter. This approach does not use any of the low-fidelity data to make predictions, so the quality of the predictions is strictly a function of the amount of high-fidelity training data. This method represents how well one might do without taking advantage of multi-fidelity data, and serves as a baseline for comparison against the multi-fidelity approach.

4.2 Locally Weighted, Multi-fidelity

Our model-free multi-fidelity method uses an approach similar to the high-fidelity method, but it also takes advantage of the low-fidelity data. We use LW regression on the low-fidelity training data to obtain a low-fidelity predictor R_l that predicts joint-actions $\hat{A}_l^{tr}$ for the training encounters S_l^{tr}.

We use the the low-fidelity predictor to augment the high-fidelity training data. The augmented input $(S_h^{tr}, R_l(S_h^{tr}))$ is used to train our augmented high-fidelity predictor $R_{h'}$ that predicts joint-actions based on the high-fidelity data. The test encounters S^{te} were used as inputs to the augmented high-fidelity predictor to predict the joint-action for the test encounters as $\hat{A}^{te} = R_{h'}(S^{te}, R_l(S^{te}))$. In essence, we are using the predictions from the low-fidelity predictor as features for training the high-fidelity predictor.

5 Model-Based Prediction Approaches

The model-free methods described above ignore what is known about how the pilots make their decisions. In many cases, we have a model of the process with some number of free parameters. In our case, we model the players interacting in a level-k environment, and treat the utility weights w_h as unknown parameters that can be learned from data, allowing the prediction of behavior in new situations. Such an approach is used in the inverse reinforcement learning literature (Baker et al., 2009; Ng and Russell, 2000), where the goal is to learn parameters of the utility function of a single agent from experimental data. This section outlines three model-based approaches where the data comes from experiments of varying fidelities. The first two are based on *maximum a posteriori* (MAP) parameter estimates, and the third is a Bayesian approach.

5.1 MAP High-fidelity

To demonstrate the benefit of multi-fidelity model-based approaches, we use a baseline method that exclusively relies on high-fidelity data to make predictions. Since we have a model of interacting humans, we can simply use the test encounters S^{te} as inputs into the model and use the resulting joint-actions as predictions. However, to make accurate predictions, we must estimate the utility weights used by the pilots. To find the MAP utility weights w_h^*, we first need to estimate $p(A \mid S, w)$, which is the probability of the joint-actions A given encounters S and weight w.

As an approximation, we estimate $p(A \mid S, w)$ by simulating the high-fidelity game using a set of 1000 novel encounters (S^n), under different utility weight combinations. We stored the resulting joint-actions for the jth utility weight combination, and then estimated $p(A_h^n \mid w_h^j) = \sum_S p(A_h^n \mid S^n, w_h^j)$ using kernel density estimation via diffusion (Botev et al., 2010). Fig. 2 shows how $p(A_h^n \mid w_h)$ changes across a subset of w_h settings.

We can find the MAP utility weight combination by using the joint-actions A_h^{tr} we observe for our training encounters S_h^{tr}. For each joint-action from the training data, we find the nearest neighbor joint-action in A_h^n. We sum the log-likelihood of the nearest-neighbor joint-actions for the training encounters, under each utility weight combination. The MAP utility weight combination is defined by

$$w_h^* = \arg\max_{w_h} \left[\ln p(w_h) + \sum_n \ln p(A_h^n \mid w_h)\right] \quad (6)$$

where $p(w_h)$ is the prior of weight vector w_h. In our experiments, we assume $p(w_h)$ is uniform over utility weight combinations. Once we have estimated the MAP utility weights, we can simulate our high-fidelity game using the test encounters and the MAP utility weights to obtain the predicted joint-action. The predicted joint-action is obtained by generating $N_s = 10$ samples from $p(A_h^{te} \mid S_i^{te}, w_h^*)$ and averaging them together:

$$A^{(1)}, \ldots, A^{(N_s)} \sim p(A_h^{te} \mid S_i^{te}, w_h^*) \quad (7)$$

$$\hat{A}_{h,i}^{te} = \frac{1}{N_s} \sum_{\ell=1}^{N_s} A^{(\ell)} \quad (8)$$

The amount of high-fidelity data impacts the quality of the utility-weight estimate (w_h^*), thereby limiting its effectiveness in situations when there is little data. The method described next overcomes this limitation by fusing both high- and low-fidelity data to find the utility weights to use for prediction.

5.2 MAP Multi-fidelity

In situations when there is little high-fidelity data to estimate the utility weights, it is beneficial to fuse both low- and high-fidelity data to increase the reliability of the estimate. In this approach, the predicted high-fidelity joint-actions for test encounters ($\hat{A}^{te}$) are being computed according to Eq. (8), and our MAP utility weight is still found by maximizing Eq. (6). However, instead of only using the high-fidelity data to estimate the utility weights, we use both the low-fidelity and the high-fidelity data to estimate the weights. Specifically, we find the log-likelihood of the nearest-neighbor joint-action A^n for both the low- and high-fidelity joint-actions A_h^{tr} and A_l^{tr}. Once we have estimates for the utility weights in the decision model, we can use the test encounters S^{te} as inputs to the high-fidelity model to obtain predictions $\hat{A}^{te}$, similar to what was done in Eq. (8).

Although this is a straightforward way to use both low- and high-fidelity training data, it assumes that the decision makers in the low-fidelity and high-fidelity encounters are identical. While it is plausible that the decision makers have similar utility functions to make this a valid assumption, it could also be the case that highly trained humans have very different utility functions from lesser-trained ones, or that changes in the simulation environment also cause changes in the utility function (for example, due to the decreased sense of immersion). Therefore, we would like a method that relaxes the equal utility assumption, allowing us to make predictions for games that are different for low- and high-fidelity.

5.3 Bayesian Multi-fidelity

The approaches described above use the test encounters as inputs into the high-fidelity model and sample joint-actions from $p(A_h^{te} \mid S_h^{te}, w_h^*)$ using the MAP utility weight estimate w_h^* to compute the predicted joint-actions $\hat{A}_h^{te}$. In contrast, a Bayesian approach to multi-fidelity makes its predictions by sampling joint-actions from $p(A_h^{te} \mid S^{te}, w_h^j)$ under *each* of the j high-fidelity utility weight combinations, and then weighting the predicted joint-actions by the probability of the utility weights, given the low- and high-fidelity joint-actions observed from the training encounters:

$$\hat{A}_{h,i}^{te} = \sum_j p(A_{h,k}^{te} \mid S_i^{te}, w_h^j) p(w_h^j \mid A_l^{tr}, A_h^{tr}) \quad (9)$$

where

$$p(w_h^j \mid A_l^{tr}, A_h^{tr}) = p(w_h^j \mid A_h^{tr}) \sum_k p(w_l^k \mid A_l^{tr}) p(w_l^k, w_h^j)$$

The probability distribution $p(w_h^j \mid A_h^{tr})$ was estimated using the method outlined in Section 5.1. The low-fidelity version of this distribution $p(w_l^k \mid A_l^{tr})$ was estimated in a similar manner for each of the k low-fidelity utility weight combinations by simulating the low-fidelity game using the same novel encounters S^n and kernel density estimation via diffusion methods.

Finally, $p(w_l, w_h) = p(w_l^1, w_l^2, w_h^1, w_h^2)$ is a prior probability over low- and high-fidelity utility weights, and can be selected based on the degree of similarity between the high-fidelity and low-fidelity simulations. For our case, we assume a Gaussian prior with a mean

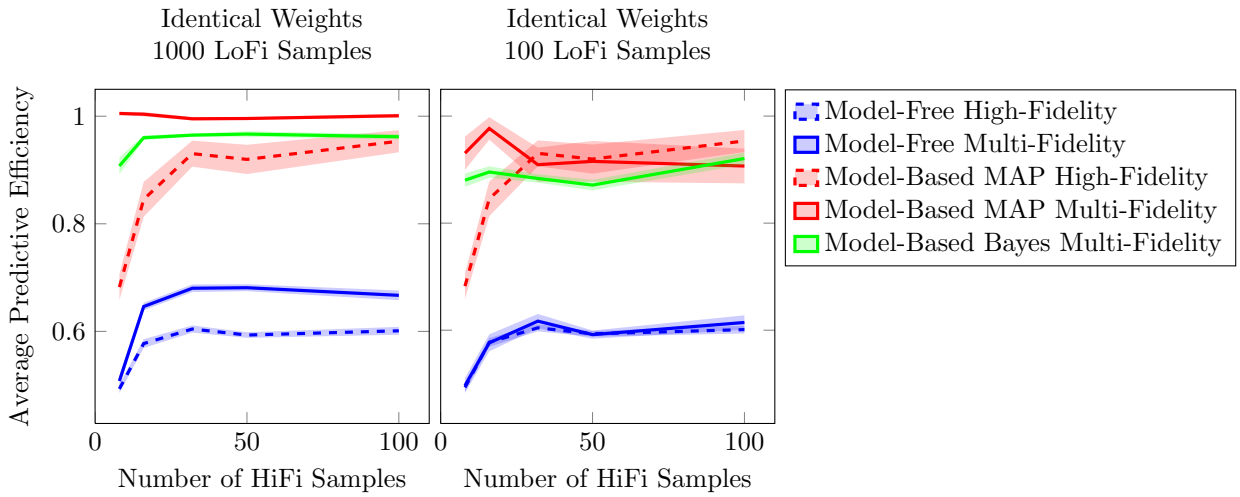


Figure 3: Predictive performance curves for various methods across different numbers of high-fidelity training samples. Lines depict mean predictive efficiency across the 10 simulations per sample-size condition, and the shaded-areas represent standard error. Results are shown for conditions in which the low- and high-fidelity utility weights are identical.

corresponding to the ground-truth utility weights (Section 6) and covariance given by:

$$\begin{bmatrix} .0017 & 0 & .0013 & 0 \\ 0 & .0017 & 0 & .0013 \\ .0013 & 0 & .0017 & 0 \\ 0 & .0013 & 0 & .0017 \end{bmatrix}$$

This covariance assumes that there is variation among players at the same fidelity, but also that the weights of the players are similar (but not identical) across fidelities. However, it assumes that the utilities of the two players within any specific encounter are independent.

A Bayesian approach to multi-fidelity allows for predictions to be made that account for the uncertainty in how well the utility weights account for the joint-actions observed in training. Such an approach also relaxes the assumption of identical utility weights across games through a prior distribution that encodes the coupling of these parameters.

6 Results

In order to assess the performance of both model-free and model-based approaches to multi-fidelity prediction, we evaluated each method with different amounts of high-fidelity training data. We assessed the benefit of incorporating an additional 100 or 1000 low-fidelity training examples, depending on the condition.

Since novices may employ different strategies than domain experts, we also simulated the scenario under different ground-truth utility weight relationships between the low- and high-fidelity games. For the scenario where experts and novices employ similar behavior, the ground-truth utility weights were set to:

$$w_l^{gt} = \{w^1 = 0.89, w^2 = 0.90\} \quad (10)$$

$$w_h^{gt} = \{w^1 = 0.89, w^2 = 0.90\} \quad (11)$$

For the scenario where there is a small difference in how novices act, the ground-truth low-fidelity utility weights were changed to:

$$w_l^{gt} = \{w_l^1 = 0.88, w_l^2 = 0.89\} \quad (12)$$

For the scenario where novices perform much differently than experts, the low-fidelity ground-truth utility weights were:

$$w_l^{gt} = \{w_l^1 = 0.80, w_l^2 = 0.81\} \quad (13)$$

Performance is measured in terms of "predictive efficiency," which is a number typically between 0 and 1. It is possible to obtain a number above 1 because the normalization factor is an estimated lower-bound of the test-set error. The test-set error is given by

$$D = \sum_i d(\hat{A}_i, A_i^{te}), \quad (14)$$

where $d(\cdot,\cdot)$ is the Euclidean distance, $\hat{A}_i$ is the predicted action for the ith encounter, and A_i^{te} is th-e joint-action of the ith test encounter. A lower-bound on the test-set error D^{lb} can be approximated by estimating the error when using the ground-truth model in conjunction with Eq. (8). The predictive efficiency is given by D^{lb}/D.

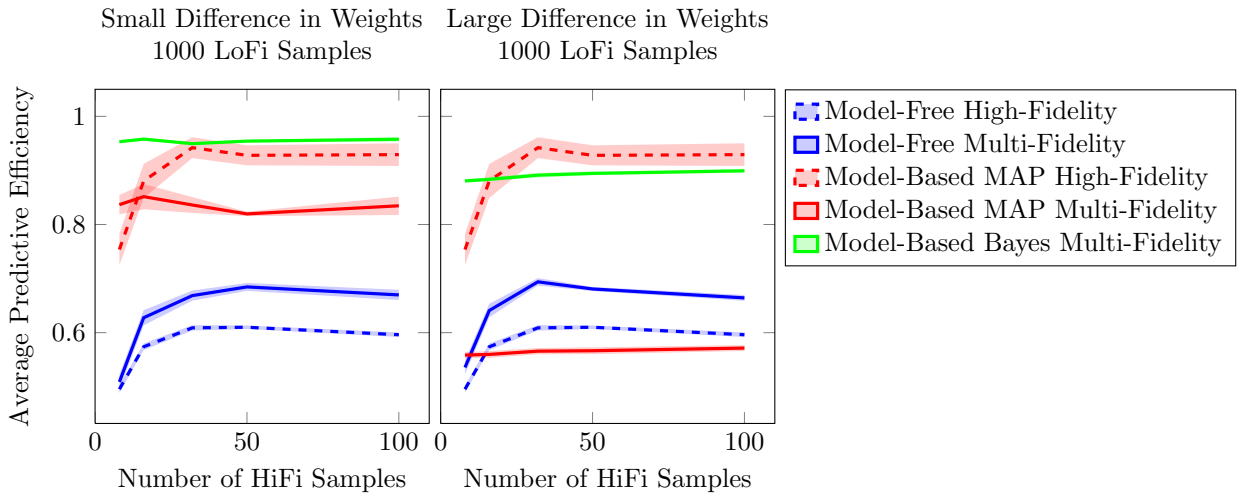


Figure 4: Predictive performance curves for various methods across different numbers of high-fidelity training samples. Lines depict mean predictive efficiency across the 10 simulations per sample-size condition, and the shaded-areas represent standard error. Results are shown for conditions in which the low- and high-fidelity utility weights have small (left) and large (right) differences between them.

Figure 3 shows how average predictive efficiency changes as a function of the amount of low- and high-fidelity training data for the identical utility weight condition. Figure 4 shows how average predictive efficiency is impacted by the level of discrepancy between low- and high-fidelity utility weights.

Multi-fidelity model-free approaches (blue solid lines) predicted better than methods that used only high-fidelity data (blue dashed lines), except for the case in which there was little (100 samples) low-fidelity available to train the model (Figure 3).

Model-based multi-fidelity approaches had better predictive performance (red and green solid lines) than high-fidelity methods (red dashed line) in cases when there was a large amount of low-fidelity data (1000 samples), and there was little or no difference in the utility weights used by the low-fidelity and high-fidelity humans in the task (Figures 3 and 4).

7 Conclusions and Further Work

We developed a multi-fidelity method for predicting the decisions of interacting humans. We investigated the conditions under which these methods produce better performance than exclusively relying on high-fidelity data. In general, our results suggest that multi-fidelity methods provide benefit if there is a sufficient amount of low-fidelity training data available and the differences between expert and novice behavior is not too large. Future effort will investigate these distinctions in greater detail.

This approach can also be extended to many domains beyond aviation. For example, work is currently being conducted that uses a 'power-grid game' to produce low-fidelity data to train algorithms that detect attacks on the system. This data could be combined with relatively sparse high-fidelity data (e.g., historical data involving known attacks) to increase the predictive performance of the model.

Acknowledgments

The Lincoln Laboratory portion of this work was sponsored by the Federal Aviation Administration under Air Force Contract #FA8721-05-C-0002. The NASA portion of this work was also sponsored by the Federal Aviation Administration. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government. We would like to thank James Chryssanthacopoulos of MIT Lincoln Laboratory for his comments on early versions of this work.

References

Amato, C., Bernstein, D. S., and Zilberstein, S. (2009). Optimizing fixed-size stochastic controllers for POMDPs and Decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21:293–320.

Aponso, B., Beard, S., and Schroeder, J. (2009). The NASA Ames vertical motion simulator—a facility engineered for realism. In *Royal Aeronautical Society Spring Flight Simulation Conference*.

Baker, C., Saxe, R., and Tenenbaum, J. B. (2009). Action understanding as inverse planning. *Cognition*, 113:329–349.

Botev, Z., Grotowski, J., and Kroese, D. (2010). Kernel density estimation via diffusion. *The Annals of Statistics*, 38:2916–2957.

Camerer, C. (2003). *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press.

Caplin, A., Dean, M., and Martin, D. (2011). Search and satisficing. *American Economic Review*, 101(7):2899–2922.

Costa-Gomes, M., Crawford, V., and Broseta, B. (2001). Cognition and behavior in normal-form games: An experimental study. *Econometrica*, 69(5):1193–1235.

Doshi, P. and Gmytrasiewicz, P. (2005). A particle filtering based approach to approximating interactive POMDPs. In *AAAI Conference on Artificial Intelligence*.

Doshi, P. and Gmytrasiewicz, P. (2006). On the difficulty of achieving equilibria in interactive POMDPs. In *AAAI Conference on Artificial Intelligence*.

Graf, E. W., Warren, P. A., and Maloney, L. T. (2005). Explicit estimation of visual uncertainty in human motion processing. *Vision Research*, 45:3050–3059.

Harris, C. and Wolpert, D. (1998). Signal-dependent noise determines motor planning. *Nature*, 394:780–784.

Kittur, A., Chi, E., and Suh, B. (2008). Crowdsourcing user studies with Mechanical Turk. In *ACM Conference on Human Factors in Computing Systems*.

Koller, D. and Milch, B. (2003). Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45:181–221.

Lee, R. and Wolpert, D. H. (2012). Game theoretic modeling of pilot behavior during mid-air encounters. In *Decision Making with Imperfect Decision Makers*. Springer.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning*.

Myerson, R. B. (1997). *Game Theory: Analysis of Conflict*. Harvard University Press.

Ng, A. and Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*.

Paolacci, G., Chandler, J., and Ipeirotis, P. G. (2010). Running experiments on Amazon Mechanical Turk. *Judgement and Decision Making*, 5:411–419.

Robinson, T., Willcox, K., Eldred, M., and Haimes, R. (2006). Multifidelity optimization for variable-complexity design. In *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*.

Schlicht, E. J. and Schrater, P. R. (2007). Effects of visual uncertainty on grasping movements. *Experimental Brain Research*, 182:47–57.

Sellitto, M., Ciaramelli, E., and di Pellegrino, G. (2010). Myopic discounting of future rewards after medial orbitofrontal damage in humans. *Journal of Neuroscience*, 30:16429 16436.

Simon, H. (1956). Rational choice and the structure of the environment. *Psychological Review*, 63:129–138.

Simon, H. (1982). *Models of bounded rationality*. MIT Press.

Vul, E., Frank, M., Alverez, G., and Tenenbaum, J. B. (2010). Explaining human multiple object tracking as resource-constrained approximate inference in a dynamic probabilistic model. In *Advances in Neural Information Processing Systems*.

Vul, E., Goodman, N., Griffiths, T. L., and Tenenbaum, J. B. (2009). One and done? Optimal decisions from very few samples. In *COGSCI Annual Meeting of the Cognitive Science Society*.

Weis, Y., Simoncelli, E. P., and Adelson, E. H. (2002). Motion illusions as optimal percepts. *Nature Neuroscience*, 5:598–604.

Wright, J. R. and Leyton-Brown, K. (2010). Beyond equilibrium: Predicting human behavior in normal-form games. In *AAAI Conference on Artificial Intelligence*, Atlanta, Georgia.

Wright, J. R. and Leyton-Brown, K. (2012). Behavioral game-theoretic models: A bayesian framework for parameter analysis. In *International Conference on Autonomous Agents and Multiagent Systems*.

Zeng, Y. and Xiang, Y. (2010). Time-critical decision making in interactive dynamic influence diagram. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*.

Latent Composite Likelihood Learning for the Structured Canonical Correlation Model

Ricardo Silva
Department of Statistical Science and CSML
University College London
RICARDO@STATS.UCL.AC.UK

Abstract

Latent variable models are used to estimate variables of interest – quantities which are observable only up to some measurement error. In many studies, such variables are known but not precisely quantifiable (such as "job satisfaction" in social sciences and marketing, "analytical ability" in educational testing, or "inflation" in economics). This leads to the development of measurement instruments to record noisy indirect evidence for such unobserved variables such as surveys, tests and price indexes. In such problems, there are postulated latent variables and a given measurement model. At the same time, other unantecipated latent variables can add further unmeasured confounding to the observed variables. The problem is how to deal with unantecipated latents variables. In this paper, we provide a method loosely inspired by canonical correlation that makes use of background information concerning the "known" latent variables. Given a partially specified structure, it provides a structure learning approach to detect "unknown unknowns," the confounding effect of potentially infinitely many other latent variables. This is done without explicitly modeling such extra latent factors. Because of the special structure of the problem, we are able to exploit a new variation of composite likelihood fitting to efficiently learn this structure. Validation is provided with experiments in synthetic data and the analysis of a large survey done with a sample of over 100,000 staff members of the National Health Service of the United Kingdom.

1 CONTRIBUTION

We present a method for learning the structure of a latent variable model, where latent variables are divided into two categories: i. latent variables which we would like to estimate, as in any smoothing task (e.g., to generate latent representations of data points for visualization, clustering, and ranking, among other tasks); ii. all other latent variables, which we are not interested in estimating but which can add further confounding among observed variables and as such cannot be ignored. They are nuisance variables.

This setup is motivated by many practical problems in the applied sciences where target latent variables are chosen upfront, with observed variables designed to measure the unobservable variables of interest. Consider the simple illustrative example of Figure 1.

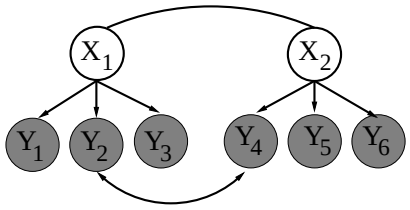


Figure 1: A latent variable model with a mixed graphical representation (that is, more than one type of edge).

Here, each X_i is a latent variable and each Y_i an observed variable. For instance, X_1 could represent a latent variable quantifying the level of job satisfaction, postulated to be measured by answers to questions such as $Y_1 \equiv$ "I can choose my own method of working" and $Y_2 \equiv$ "I have strong support from my manager". X_2 could represent a latent level of job responsibility, measured by questions such as $Y_4 \equiv$ "I know what my responsibilities are" and $Y_5 \equiv$ "I am regularly consulted by my team".

Other nuisance latent factors might correlate, say, the employee support from her manager and the level of interaction with her team. That is, associations not accounted by X_1 and X_2. This is represented graphically in Figure 1 by a bi-directed edge between two observed items, a notation widely used in the structural equation model literature (Bollen, 1989) and futher formalized by Richardson and Spirtes (2002). In a linear Gaussian model, for in-

stance, this could be parameterized by the measurement model equation $Y_i = \mathbf{X}^\mathsf{T}\beta + \epsilon_i$, where the covariance of the error terms for Y_2 and Y_4 is allowed to be non-zero. More details will be given in the sequel.

Conditioned on the background knowledge of a model where we specify the "known unknowns" – the latent features we would like to estimate by a postulated relationship to observations – we would like to identify the remaining dependence structure given by all remaining factors we did not specify in advance – the "unknown unknowns." This provides a meeting point between the common practice of designing measurement models for the practical goal of quantifying specific latent factors (Bollen, 1989), and purely data-driven approaches for discovering latent structure (Elidan et al., 2000; Silva et al., 2006).

The structure of the paper is as follows. A formal problem definition is given in Section 2, where we specifiy precisely the goals, classes of models and assumptions embedded in our procedure. Algorithms for structure learning are described in detail in Section 3, based on variations of the composite likelihood method integrated within a expectation-maximization framework. Further context and related work is discussed in Section 4. Experiments with synthetic and real data are discussed in Section 5, followed by a Conclusion in Section 6.

2 PROBLEM SPECIFICATION AND MODEL SPACE

In this section, we first define the space of graphical models which is assumed to contain the data generating process of interest. We then present further details on the parametric assumptions and a formulation based on *cumulative distribution networks* (Huang and Frey, 2008; Silva et al., 2011).

2.1 Structural Conditions

In our setup, we assume the following:

1. that the data is generated according to $\mathcal{G}$, an underlying directed acyclic graph (DAG) (Koller and Friedman, 2009) composed of latent and observed variables. In $\mathcal{G}$, observed variables are not parents of any other variable, as in typical models of factor analysis (Bartholomew and Knott, 1999);
2. that an expert provides a partition of the set of observed variables $\mathbf{Y}$, such that each set $\mathcal{S}_i \subset \mathbf{Y}$ in this partition corresponds to the observed children of a latent variable X_i. For example, in Figure 1, the partition is given by sets $\mathcal{S}_1 \equiv \{Y_1, Y_2, Y_3\}$ and $\mathcal{S}_2 \equiv \{Y_4, Y_5, Y_6\}$;
3. let $\mathbf{X}_\mathcal{S}$ be this set of latent variables associated with the partition (e.g., $\{X_1, X_2\}$ in Figure 1). We also allow for a (possibly infinite) set of latent variables $\mathbf{X}_\infty$ that is disjoint of $\mathbf{X}_\mathcal{S}$;
4. elements of $\mathbf{X}_\mathcal{S}$ are arbitrarily connected to each other in $\mathcal{G}$, and elements of $\mathbf{X}_\infty$ are also arbitrarily connected to each other. However, any $X_i \in \mathbf{X}_\mathcal{S}$ is marginally independent of any $X_j \in \mathbf{X}_\infty$ in $\mathcal{G}$.

2.2 Problem Specification

We want to account for any association among elements of $\mathbf{Y}$ that is due to $\mathbf{X}_\infty$, so that functionals of the conditional distribution $\mathcal{P}(\mathbf{X}_\mathcal{S} \mid \mathbf{Y})$ can be correctly estimated from data.

Instead of modeling the number of elements in $\mathbf{X}_\infty$ and how they relate to each other as in a nonparametric latent variable model formulation (e.g. Wood et al., 2006), we will treat the existence of $\mathbf{X}_\infty$ as a black-box. We model directly the dependencies that arise from its existence within a *mixed graph* formulation (Richardson and Spirtes, 2002). A mixed graph is a graph with more than one type of edge.

Let $\mathcal{G}_m$ be a graph with vertices $\mathbf{Y} \cup \mathbf{X}_\mathcal{S}$, and directed edges from each X_i to each element of $\mathcal{S}_i$. For simplicity, the structure among elements of $\mathbf{X}_\mathcal{S}$ is assumed to be a fully connected network of undirected edges. If two observed variables Y_i and Y_j have a common ancestor in $\mathbf{X}_\infty$ in $\mathcal{G}$, add a bi-directed edge $Y_i \leftrightarrow Y_j$ to $\mathcal{G}_m$. We say that $\mathcal{G}_m$ is the mixed graph induced by $\mathcal{G}$ and $\mathbf{X}_\mathcal{S}$. The conditional independence constraints entailed by $\mathcal{G}_m$ all hold in $\mathcal{G}$ (Richardson and Spirtes, 2002). Hence, identifiability issues aside, fitting a model based on a graph different from $\mathcal{G}_m$ should in general result in a misspecified distribution and give a inadequate model for $\mathcal{P}(\mathbf{X}_\mathcal{S} \mid \mathbf{Y})$.

Our goal can then be summarized as: *learn the structure of $\mathcal{G}_m$ by finding the correct bi-directed substructure.* In the example of Figure 1, this means returning the information that $Y_2 \leftrightarrow Y_4$ is the only bi-directed edge in $\mathcal{G}_m$.

The *practical* assumption here is that the bi-directed component of $\mathcal{G}_m$ is sparse. Sparsity in a bi-directed component corresponds to a marginal independence constraint among elements of $\mathbf{X}_\infty$: the lack of an edge $Y_i \leftrightarrow Y_j$ corresponds to the parents of Y_i in $\mathbf{X}_\infty$ being marginally independent of the parents of Y_j in $\mathbf{X}_\infty$. The extent to which this assumption is valid will depend on how well the dependence between elements of $\mathbf{Y}$ is captured by $\mathbf{X}_\mathcal{S}$ and it is verified empirically in Section 5. This complements Wood et al. (2006), which makes fewer assumptions but has to deal with a much harder problem.

2.3 Comments

The setup where observed variables are partitioned into sets corresponding to particular semantic groups is a key idea behind *canonical correlation analysis* (CCA). The standard CCA corresponds to a partition into two sets. For rank-one CCA, this corresponds to a graphical model with a single

latent variable being a parent of all observations, variables in each partition connected to each other by bi-directed edges. There are no bi-directed connections across variables in different sets of the partition, but CCA in general allows for several independent latent variables being common parents of all observations. A general latent variable model interpretation of canonical correlation analysis was originally introduced in a series of reports by Wegelin et al. (2001, 2002), and by Bach and Jordan (2005). In contrast, we are assuming that a single latent variable is (explicitly) a parent of each given group of variables, that such latent variables can be mutually dependent, but that extra factors are allowed. Common between our model space and CCA is the idea of having a partition of the observed variables. As such, we refer to the problem of searching for a structure in the space of graphs with a known partition as the *structured canonical correlation analysis* problem. We once again emphasize that a major motivation for this type of analysis is a domain-dependent assumption about which latent variables are being measured, and that such variables have a prior interpretation.

2.4 Parametric Assumptions

For a fixed level of $\mathbf{X}_{\mathcal{S}}$, observed variables $\mathbf{Y}$ should have marginal independence constraints as implied by the corresponding bi-directed structure. In a Gaussian parameterization, for instance, this would mean that the covariance of Y_i and Y_j given $\mathbf{X}_{\mathcal{S}}$ is constrained to be zero if Y_i is not adjacent to Y_j, and free otherwise (Richardson and Spirtes, 2002).

For this paper, we will model binary data. Latent variables $\mathbf{X}_{\mathcal{S}}$ are assumed to follow a zero mean Gausssian with an unknown covariance matrix Σ. For the rest of this section, we discuss models for the conditional distribution of observed variables given $\mathbf{X}_{\mathcal{S}}$. We will assume a linear model for each univariate dependence $\mathcal{P}(Y_i = 0 \mid X_j = x_j) \equiv \Phi(0; \beta_{i1} x_j + \beta_{i0}, 1)$ for $Y_i \in \mathcal{S}_j$. Function $\Phi(0; \mu, \sigma^2)$ is the probability of a Gaussian of mean μ and variance σ^2 being negative. Coefficients $\{\beta_{i1}, \beta_{i0}\}$ are unknown.

Given these univariate conditionals, a conditional joint is needed. Models of marginal independence for binary data can be obtained from a Gaussian parameterization, as discussed by Silva and Ghahramani (2009). However, in our context we will be also interested in performing Bayesian model selection. Although priors for sparse covariance matrices exist, performing model selection in a large space is computationally problematic: we are particularly motivated by the modeling of surveys with a large sample size. More details on that are discussed in Section 4.

2.4.1 Brief review of CDNs

Instead, we will adopt the cumulative distribution network framework (CDN) of Huang and Frey (2008, 2011). Given a symmetric graph, a CDN model defines the cumulative distribution function $F(\mathbf{y})$ of a multivariate distribution by a product of f factors. Each factor $F_i(\cdot)$ has as arguments the variables in a clique $\mathbf{Y}_{f_i}$ of the graph:

$$F(\mathbf{y}) \equiv \prod_{i=1}^{f} F_i(\mathbf{y}_{f_i})$$

As shown by Silva et al. (2011), this can be extended to accommodate conditional distributions and integrated with a copula modeling framework (Nelsen, 2007) [1]. For a fixed instantiation $\mathbf{x}$ of the parents $\mathbf{X}$ of a set of random variables $\mathbf{Y}$, its conditional CDF $\mathcal{P}(\mathbf{Y} \leq \mathbf{y} \mid \mathbf{X} = \mathbf{x}) \equiv F(\mathbf{y} \mid \mathbf{x})$ can be parameterized as

$$F(\mathbf{y} \mid \mathbf{x}) \equiv \prod_{i=1}^{f} C_i(u_{\mathbf{x}}(y_{f_i;1})^{e_{f_i;1}}, \ldots, u_{\mathbf{x}}(y_{f_i;n(i)})^{e_{f_i;n(i)}}) \tag{1}$$

where $C_i(\cdot)$ is a copula function, $y_{f_i;j}$ denotes the instantiation of the j-th variable in clique f_i, exponent $e_{f_i;j}$ is a non-negative parameter, and $n(i)$ the number of elements in clique f_i. Moreover,

$$u_{\mathbf{x}}(y_i) \equiv \mathcal{P}(Y_i \leq y_i \mid \mathbf{x}) \tag{2}$$

Also, for each Y_i, the sum of its respective exponents $e_\star$, across all factors containing Y_i, is equal to 1. For simplicity, we fix each exponent associated with Y_i to be $1/h(i)$, with $h(i)$ being the number of factors containing Y_i.

A powerful property of CDNs is its simple marginalization procedure: calculation of the marginal of a CDF by marginalizing a subset $\mathbf{Y}_{marg}$ consists of evaluating the CDF where $\mathbf{Y}_{marg} = \infty$. For instance, $\mathcal{P}(Y_1 \leq y_1, Y_2 \leq y_2) = \mathcal{P}(Y_1 \leq y_1, Y_2 \leq y_2, Y_3 \leq \infty, \ldots, Y_N \leq \infty)$.

In a graph such as $Y_1 \leftrightarrow Y_2 \leftrightarrow Y_3$, the CDF is given by a CDN model $F_1(y_1, y_2)F_2(y_2, y_3)$ and the corresponding marginal for Y_1 and Y_3 is given by $F_1(y_1, \infty)F_2(\infty, y_3) \equiv g(y_1)h(y_3)$, corroborating the fact that Y_1 and Y_3 should be marginally independent. In the case of a model

[1] A full description of copula models is beyond the scope of this paper. Nelsen (2007) provides further details, and Huang and Frey (2011), Silva et al. (2011) provide examples within the context of CDNs. For the purposes of this paper, it suffices to understand that a copula is nothing but a d-dimensional CDF with (continuous) uniform marginals in $[0, 1]^d$. Its motivation can then be understood: it allows for the construction of a joint distribution where marginals can be parameterized separately. By defining each univariate marginal $\mathcal{P}(y_i \mid \mathbf{x})$ separately, the transformation $u_{\mathbf{x}}(y_i) = \mathcal{P}(Y_i \leq y_i \mid \mathbf{x})$ gives an uniform $[0, 1]$ random variable. These transformed variables can then be plugged into a joint distribution with uniform marginals, which becomes a distribution with arbitrary marginals in the original space. This, among other uses, allows for the use of plug-in estimates for marginals to be combined with other estimators for joints. In this case, no further bias will be introduced into the marginal models even if the joint is misspecified or if the copula estimator is biased.

for $F(y_1, y_2, y_3 \mid \mathbf{x})$, according to (1), we have that $F(y_1, y_2, y_3 \mid \mathbf{x})$ is given by

$$C_1(u_{\mathbf{x}}(y_1), u_{\mathbf{x}}(y_2)^{1/2}) C_2(u_{\mathbf{x}}(y_2)^{1/2}, u_{\mathbf{x}}(y_3))$$

Recall that, if $C(u, v)$ is a copula function, then $C(u, 1) = u$ (since copulas are CDFs with $U(0, 1)$ marginals). One can then verify that $F(y_1, y_3 \mid \mathbf{x}) = \mathcal{P}(Y_1 \leq y_1 \mid \mathbf{x})\mathcal{P}(Y_3 \leq y_3 \mid \mathbf{x})$ and $F(y_2 \mid \mathbf{x}) = \mathcal{P}(Y_2 \leq y_2 \mid \mathbf{x})$ as desired.

Another advantage of the CDN formulation is that parameters across factors are functionally independent. This is in contrast with, for instance, the sparse covariance models of Richardson and Spirtes (2002) and Silva and Ghahramani (2009), where a positive definite constraint ties all parameters. Having no constraints across different factors will be a fundamental property to be exploited in our learning procedure. We call this property *parameter modularity*.

The difficulty with the CDN formulation is that in order to calculate the likelihood function, as required for any likelihood-based learning procedure, one has to convert the conditional CDFs into probability mass functions (PMFs) (Huang and Frey, 2011). This can potentially take an exponential amount of time on the number of variables in the graph. However, the marginalization property and the parameter modularity property of the CDNs leads to an attractive way of performing efficient learning, as we will see in the next section.

3 A LATENT COMPOSITE LIKELIHOOD APPROACH

Let $\mathcal{D} = \{\mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(N)}\}$ be our data, a set of binary measurements of dimension p. For a given graphical structure $\mathcal{G}_m$ and fixed coefficient parameters $\{\beta_{i1}, \beta_{i0}\}$ and latent covariance matrix Σ, the marginal likelihood of $\{\mathcal{G}_m, \{\beta_{i1}, \beta_{i0}\}, \Sigma\}$ is

$$\mathcal{P}(\mathcal{D} \mid \mathcal{G}_m, \{\beta_{i1}, \beta_{i0}\}, \Sigma) =$$

$$\int \mathcal{P}(\mathcal{D}, \mathbf{X}^{1:N}, \theta \mid \mathcal{G}_m, \beta, \Sigma)\, d\mathbf{X}^{1:N}\, d\theta \qquad (3)$$

where $\mathbf{X}^{1:N}$ is a shorthand notation for $\{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(N)}\}$, β is a shorthand notation for $\{\beta_{1,1}, \beta_{1,0}, \ldots, \beta_{p,1}, \beta_{p,0}\}$, and parameter set θ describes parameters associated with copula functions.

One approach for model selection is to assign a prior $\pi(\mathcal{G}_m)$ over possible graph structures and then choose $\mathcal{G}_m$ that maximizes $\mathcal{P}(\mathcal{D} \mid \mathcal{G}_m, \{\beta_{i1}, \beta_{i0}\}, \Sigma)\pi(\mathcal{G}_m)$. Since $\{\{\beta_{i1}, \beta_{i0}\}, \Sigma\}$ are also unknown, we could marginalize then away. However, such parameters do not affect the complexity of the model, and with the goal of having an efficient computational procedure, we will treat these parameters also as nuisance parameters and maximize with respect to them along with $\mathcal{G}_m$. In this paper, we will assume that individual pairwise factors are associated with each bi-directed edge in $\mathcal{G}_m$. Our prior for θ is independent[2] of $\mathcal{G}_m$ and factorizes as

$$\pi(\theta \mid \mathcal{G}_m) = \pi(\theta) = \prod_{1 \leq i < j \leq p} \pi(\theta_{ij}) \qquad (4)$$

Maximizing any expression that depends on (3) poses a formidable computational problem. Moreover, it depends on conditional probability expressions $\mathcal{P}(\mathbf{Y} \mid \mathbf{x})$. Such conditional mass functions need to be obtained from the canonical CDF $F(\cdot)$ to PMF $\mathcal{P}(\cdot)$ transformation (Joe, 1997), which for binary data boils down to:

$$\mathcal{P}(\mathbf{Y} = \mathbf{y}) = \sum_{z_1=0}^{1} \cdots \sum_{z_p=0}^{1} (-1)^{\sum_{i=1}^{p} z_i} F(\mathbf{y} - \mathbf{z}) \qquad (5)$$

This is of course exponential in p, but if the corresponding bi-directed component is a symmetric graph of low tree-width, the expression can be calculated efficiently by dynamic programming and used by any learning method that requires the likelihood function. The original derivation by Huang et al. (2010) is quite complex and provides insights on how approximate methods should behave. However, in the sequel we will make use of exact methods only, and as such we provide in the Supplementary Material a straighforward reduction of (5) to a standard inference problem in factor graphs – making the method easier to implement.

Integrating away θ and $\mathbf{X}$ is harder in general and large tree-width graphs are still a possibility. As such we avoid methods that attempt to maximize (3). The core procedure is based on structural composite likelihood learning. It is described in Section 3.1, and refined in Section 3.2. Further implementation details are given in Section 3.3. An alternative to these methods is to emulate a constraint-based approach (Spirtes et al., 2000) for structure discovery, providing a non-iterative procedure to identify which bi-directed edges are needed. This is done in Section 3.4. A brief discussion on model identification is provided in the Supplementary Material.

3.1 Basic Structural Composite Likelihood

A composite likelihood function (Varin et al., 2011) for a parameter of interest θ is defined as

$$CL(\theta; \mathcal{D}) = \prod_{k \in K} \mathcal{L}_k(\theta; \mathcal{D})^{w_k}$$

where $\mathcal{L}_k(\cdot)$ is the likelihood function resulting from the conditional probability (or density) function of a subset of

[2]Parameter θ_{ij} will not, of course, affect the likelihood function if $Y_i \leftrightarrow Y_j$ is not in $\mathcal{G}_m$. One could otherwise interpret θ_{ij} as simply not existing in this case, but this way of interpreting θ_{ij} will make the presentation easier without being less precise.

$\mathbf{Y}$ given another subset, and w_k are user-specified weight parameters. It is intuitive to understand why a composite likelihood function can provide consistent estimates of θ: if an unique model is identifiable from the marginal conditional densities used in each $\mathcal{L}_k(\cdot)$, then maximizing the log-composite likelihood is equivalent to minimizing the KL-divergence between each marginal conditional and the empirical distribution. By matching each marginal as the divergence goes to zero, one recovers the parameters.

Joreskog and Moustaki (2001) applied this concept in the context of latent variable models, by fitting a probit model using univariate and bivariate marginals with equal weight. A key fact is that bivariate likelihoods can be integrated numerically, since only two underlying latent variables are present. Although still computationally demanding, this approach does not require any Markov chain Monte Carlo within a stochastic EM procedure, nor requires biased approximations such as mean-field methods. We initially propose a similar idea, where our (penalized) composite likelihood function is given by

$$PCL(\mathcal{G}_m, \beta, \Sigma) \equiv \mathcal{F}_{\mathcal{G}_m}^{(\beta,\Sigma)} + \log \pi(\mathcal{G}_m), \qquad (6)$$

$$\mathcal{F}_{\mathcal{G}_m}^{(\beta,\Sigma)} \equiv \sum_{i<j} \log \mathcal{P}(\mathbf{Y}_i^{1:N}, \mathbf{Y}_j^{1:N} \mid \mathcal{G}_m, \beta, \Sigma)$$

When using one-parameter copula functions (Nelsen, 2007), each bivariate term $\mathcal{P}(\mathbf{Y}_i^{1:N}, \mathbf{Y}_j^{1:N} \mid \mathcal{G}_m, \beta, \Sigma)$ requires the numerical integration of at most three terms: the copula parameter θ_{ij} corresponding to the edge $Y_i \leftrightarrow Y_j$ and up to two latent variables per data point configuration, as explained in Section 3.3.

A greedy procedure for optimizing $PCL(\cdot,\cdot,\cdot)$ is outlined in table Algorithm 1. It takes as input a partition over observed variables ($\mathcal{S}$) and a dataset $\mathcal{D}$. Algorithm 1 alternates between optimizing the objective function with respect to the continuous parameters (Step 5), and optimizing structure (Steps 6). Optimization in Step 5 is done with respect to the coefficient space Ω_β and correlation matrix space Ω_Σ, as explained in further detail in Section 3.3. The structural update is a standard greedy algorithm that picks the best choice of graph within the graph space $\mathcal{G}_m^{+/-}$: the space that includes $\mathcal{G}_m$ and all mixed graphs that differ from $\mathcal{G}_m$ by exactly one bi-directed edge. Finally, procedure GETDAG($\mathcal{S}$) at the beginning of Algorithm 1 just returns the variable space ($\mathbf{Y}$, $\mathbf{X}_\mathcal{S}$) and the initial mixed graph without any bi-directed edges. The initialization of parameters is discussed in the Supplementary Material.

Algorithm 1 Pairwise Structured CCA Learning

1: **procedure** LEARNSTRUCTUREDCCA-I($\mathcal{S}$,$\mathcal{D}$)
2: $\{\mathbf{Y}, \mathbf{X}_\mathcal{S}, \mathcal{G}_m\} \leftarrow$ GETDAG($\mathcal{S}$)
3: $\{\beta, \Sigma\} \leftarrow$ INITPARAMETERS($\mathcal{G}_m, \mathcal{D}$)
4: **repeat**
5: $\{\beta, \Sigma\} \leftarrow \arg\max_{(\Omega_\beta, \Omega_\Sigma)} PCL(\mathcal{G}_m, \beta, \Sigma)$
6: $\mathcal{G}_m \leftarrow \arg\max_{(\mathcal{G}_m^{+/-})} PCL(\mathcal{G}_m, \beta, \Sigma)$
7: **until** $\mathcal{G}_m$ has not changed.
8: **return** $\mathcal{G}_m$
9: **end procedure**

3.2 Learning via Distributed EM Bounds

Even for problems with large sample sizes, one might be concerned that using only pairwise regions might imply low statistical efficiency. Higher (statistical) efficiency can be obtained by using components with more than two observed variables. This is undesirable, as it might require sophisticated integration methods, including MCMC. We present a different way of sharing statistical power among different pairwise likelihood functions without requiring an integration procedure on dimensions higher than in the pairwise procedure of the previous Section. It is based on ideas adapted from the expectation-maximization (EM) family of optimization methods (Dempster et al., 1977).

Using the standard Jensen bound for convex combinations, but applied independently to different terms in a summation, it is possible to lower-bound $\mathcal{F}_{\mathcal{G}_m}^{(\beta,\Sigma)}$ as

$$\sum_{i<j} \int q_{ij}(\theta_{ij}) \log \frac{\mathcal{P}_{ij}(\mathbf{Y}_i^{1:N}, \mathbf{Y}_j^{1:N}, \theta_{ij} \mid \mathcal{G}_m, \beta, \Sigma)}{q_{ij}(\theta_{ij})} d\,\theta_{ij} \qquad (7)$$

The bound holds for any choice of $q_{ij}(\cdot)$ and it is well-known to be maximized by choosing $q_{ij}(\cdot)$ to be $\mathcal{P}(\theta \mid \mathbf{Y}_i^{1:N}, \mathbf{Y}_j^{1:N}, \mathcal{G}_m, \beta, \Sigma)$ (Neal and Hinton, 1998).

We will further modify the idea in (7) with a different matching between functionals $q_{ij}(\cdot)$ and log-likelihood functions. Let $X_i \in \mathbf{X}_\mathcal{S}$ and recall $\mathcal{S}_i$ are the observed children of X_i in $\mathbf{Y}$. Let $|\mathcal{S}|$ be the number of latent variables. The trick is to first rewrite $\mathcal{F}_{\mathcal{G}_m}^{(\beta,\Sigma)}$ as in Equation (8) displayed in Table 1. Using an arbitrary set of distributions $\{q_{mn}(\cdot)\}$, $1 \le m < n \le |\mathcal{S}|$, $PCL(\mathcal{G}_m, \beta, \Sigma)$ can then be rewritten and elementwise bounded as

$$PCL(\mathcal{G}_m, \beta, \Sigma) \ge \mathcal{Q}_{\mathcal{G}_m}^{(\beta,\Sigma,\{q_{mn}(\cdot)\})} + \log \pi(\mathcal{G}_m) + \kappa \qquad (10)$$

where $\mathcal{Q}_{\mathcal{G}_m}^{(\beta,\Sigma)}$ is defined as Equation (9) in Table 1, and constant κ does not depend on the free parameters $\{\mathcal{G}_m, \beta, \Sigma\}$.

Given this setup, we finally define $q_{mn}(\Theta_{mn})$ to be the joint distribution $\mathcal{P}(\Theta_{mn} \mid \mathbf{Y}_{mn}^{1:N}, \mathcal{G}_m, \beta, \Sigma)$, where $\mathbf{Y}_{mn}$ are the *joint children* of X_m and X_n, and Θ_{mn} are the copula parameters used in the corresponding marginal model for $\mathbf{Y}_{mn}$. Function $q_{mn}(\theta_{ij})$ is then the corresponding (univariate) marginal of $q_{mn}(\Theta_{mn})$.

The desirable property of (10) is that, while it still requires only a three-dimensional integration (details in Section 3.3), information from Y_k, not in $\{Y_i, Y_j\}$, can be

Table 1: Components of a Pairwise Composite Likelihood Score Function

$$\mathcal{F}_{\mathcal{G}_m}^{(\beta,\Sigma)} = \sum_{m<n} \sum_{Y_i \in \mathcal{S}_m} \sum_{Y_j \in \mathcal{S}_n} \log \mathcal{P}(\mathbf{Y}_i^{1:N}, \mathbf{Y}_j^{1:N} \mid \mathcal{G}_m, \beta, \Sigma) + \sum_{m=1}^{|\mathcal{S}|} \frac{1}{|\mathcal{S}|-1} \sum_{n=1}^{|\mathcal{S}|-1} \sum_{\{Y_i,Y_j\} \subset \mathcal{S}_m} \log \mathcal{P}(\mathbf{Y}_i^{1:N}, \mathbf{Y}_j^{1:N} \mid \mathcal{G}_m, \beta, \Sigma) \tag{8}$$

$$\begin{aligned} \mathcal{Q}_{\mathcal{G}_m}^{(\beta,\Sigma,\{q_{mn}(\cdot)\})} \quad = \quad & \sum_{m<n} \sum_{Y_i \in \mathcal{S}_m} \sum_{Y_j \in \mathcal{S}_n} \int q_{mn}(\theta_{ij}) \log \mathcal{P}(\mathbf{Y}_i^{1:N}, \mathbf{Y}_j^{1:N} \mid \mathcal{G}_m, \beta, \Sigma, \theta_{ij}) d\,\theta_{ij} + \\ & \frac{1}{|\mathcal{S}|-1} \sum_{m=1}^{|\mathcal{S}|} \sum_{n \neq m} \sum_{\{Y_i,Y_j\} \subset \mathcal{S}_m} \int q_{mn}(\theta_{ij}) \log \mathcal{P}(\mathbf{Y}_i^{1:N}, \mathbf{Y}_j^{1:N} \mid \mathcal{G}_m, \beta, \Sigma, \theta_{ij}) d\,\theta_{ij} \end{aligned} \tag{9}$$

Algorithm 2 Modified Pairwise Structured CCA Learning

1: **procedure** LEARNSTRUCTUREDCCA-II($\mathcal{S}$,$\mathcal{D}$)
2: $\{\mathbf{Y}, \mathbf{X}_\mathcal{S}, \mathcal{G}_m\} \leftarrow$ GETDAG($\mathcal{S}$)
3: $\{\beta, \Sigma\} \leftarrow$ INITPARAMETERS($\mathcal{G}_m, \mathcal{D}$)
4: **repeat**
5: **for** $1 \leq m < n \leq |\mathcal{S}|$ **do**
6: $q_{mn}(\cdot) \leftarrow \mathcal{P}(\Theta_{mn} \mid \mathbf{Y}_{mn}^{1:N}, \beta, \Sigma, \mathcal{G}_m)$
7: **end for**
8: $\{\beta, \Sigma\} \leftarrow \arg\max_{(\Omega_\beta, \Omega_\Sigma)} Q_{\mathcal{G}_m}^{(\beta,\Sigma,\{q_{mn}(\cdot)\})}$
9: $\mathcal{G}_m \leftarrow \arg\max_{(\mathcal{G}_m^{+/-})} PCL(\mathcal{G}_m, \beta, \Sigma)$
10: **until** $\mathcal{G}_m$ has not changed.
11: **return** $\mathcal{G}_m$
12: **end procedure**

passed around. Consider the graph in Figure 1 again. There is a d-connecting path between Y_2 and Y_6 given Y_4 (or "m-connecting," using the nomenclature from Richardson and Spirtes, 2002). Hence, there is a three-way interaction between Y_2, Y_4 and Y_6 that varies according to the copula between $Y_2 \leftrightarrow Y_4$, and information from $\mathbf{Y}_6$ is propagated to the estimation of $\beta_{2;1}$ and $\beta_{4;1}$ via the distribution of $\theta_{2;4}$.

A modification of Algorithm 1 is shown in Algorithm 2. The parameter fitting procedure is now expanded into Steps 5-8, with the structure learning step unmodified. The posterior $\mathcal{P}(\Theta_{mn} \mid \mathbf{Y}_{mn}^{1:N}, \mathcal{G}_m, \beta, \Sigma)$ needs to be calculated efficiently and it has to lead to efficient integration in Step 8. This is discussed in Section 3.3. Moreover, it might be surprising that in Step 9 we do not optimize structure with respect to the same function of Step 8. Optimizing the bound of Equation (10) with respect to the graph is equivalent to a composite likelihood Structural EM procedure (Friedman, 1998). While Structural EM procedures are definitely useful, we prefer to not apply it uncritically to our special situation. EM is a coordinate ascent method (Neal and Hinton, 1998), and as such it is more prone to get stuck in a local maxima compared to optimizing marginal likelihoods directly. This is particularly true when optimizing with respect to a discrete structure – in this case, it is common to prematurely converge to a structure with fewer bi-directed edges than expected if we optimize the Structural EM lower bound starting with an empty bi-directed component. Since for each data point we have no more than two latent variables per log-likelihood term in $PCL(\cdot,\cdot,\cdot)$, we avoid maximizing a lower bound in Step 9. Convergence issues are discussed in the Supplementary Material.

3.3 Implementation Details

For Algorithm 2, we obtain the posterior distribution $q_{mn}(\Theta_{mn})$ using the Laplace approximation (MacKay, 2003). Our goal is to use Algorithms 1 and 2 in problems with large sample sizes, and hence such an approximation can provide a reasonably accurate replacement for the exact posterior – which has no closed form and would require a numerical method in any case.

Even the Laplace approximation procedure requires another inner approximation. Recall that in order to obtain the mean vector and covariance matrix required to approximate the distribution of Θ_{mn} with a Gaussian, we need to maximize $\log \mathcal{P}(\mathbf{Y}_{mn}^{1:N} \mid \mathcal{G}_m, \beta, \Sigma, \Theta_{ij}) + \log \pi(\Theta_{mn})$. Let P be the total number of children between X_i and X_j. Then $\mathcal{P}(\mathbf{Y}_{mn}^{1:N} \mid \Theta_{mn}, \mathcal{G}_m, \beta, \Sigma)$ can be rewritten as

$$\prod_{\mathbf{y}_{mn} \in \{0,1\}^P} \left(\int \mathcal{P}(\mathbf{y}_{mn}, x_m, x_n \mid \Theta_{mn}, \dots) d\,x_m d\,x_n\right)^{N_{mn}}$$

which takes constant time in sample size[3] given precomputed sufficient statistics N_{mn}, the count of events $\mathbf{Y}_{mn} = \mathbf{y}_{mn}$ in dataset $\mathcal{D}$, and under an approximation to each of the two-dimensional integrals.

In our implementation, we use an approximation for the

[3] Assuming that the marginal of $\mathcal{G}_m$ over $\mathbf{Y}_{mn}$ has a bounded tree-width, this expression is tractable in the dimensionality of the problem using the CDN inference method to obtain the likelihood function (Huang and Frey, 2011, see also the Supplementary Material). The full graph $\mathcal{G}_m$ might have a large tree-width, as long as the subgraph given by $\mathbf{Y}_{mn}$ does not.

integral of $\mathcal{P}(\mathbf{y}_{mn}, x_m, x_n \mid \Theta_{mn}, \beta, \Sigma, \mathcal{G}_m)$ of the form

$$\approx \sum_k w_k(\sigma_{mn}) \mathcal{P}(\mathbf{y}_{mn}, x_m^{(k)}, x_n^{(k)} \mid \Theta_{mn}, \beta, \Sigma, \mathcal{G}_m) \tag{11}$$

for a fixed set of grid points $\{(x_m^{(k)}, x_n^{(k)})\}$ regardless of the values of m and n, but a function of the covariance $\sigma_{mn} \equiv (\Sigma)_{mn}$. Any quadrature method could be used.

We opted for an admittely crude but simple implementation which uses an arguably excessive number of points compared to adaptive quadrature methods, but which is relatively amortized taken into account such integrations need to be done several thousand times – we need to calculate our weights w_k only once. Without loss of generality, Σ can be assumed to be a correlation matrix. For each marginal $\{x_m, x_n\}$, we naively space the grid points uniformly between the 0.01 and 0.99 quantiles of the standard Gaussian. The result is a division of the space into squares centered at each $\{(x_m^{(k)}, x_n^{(k)})\}$ which does not depend on m and n. We define $w_k(\sigma_{mn})$ to be the probability mass of each square. This can be pre-computed only once in the whole procedure, also using the following simplification of Σ: we allow each entry σ_{mn} to lie only within the discrete set of choices $\{-0.99, -0.97, \ldots, 0.97, 0.99\}$ and cache $w_k(\sigma_{mn})$ for every k and possible value of σ_{mn}. Since they correspond to bivariate Gaussian integrals, recomputing these weights $\mathcal{O}(p^2)$ times at each iteration would otherwise be very costly.

Equations (8) and (9) require integrals over θ_{mn}. Given the approximation for the bivariate likelihood of each pairwise data point configuration, we introduced a discrete approximation for the prior $\pi(\theta_{mn})$ (Algorithm 1) or univariate Gaussian marginal $q_{mn}(\theta_{mn})$ that results from the Laplace approximation (Algorithm 2). The grid points are naively a number of quantiles corresponding to uniformly spread cumulative probabilities in $[0, 1]$. These quantiles are recomputed at every iteration, since this is relatively cheap.

When optimizing parameters, we optimize β for a fixed Σ by gradient methods[4]. We then optimize Σ given $\{\beta_{i1}, \beta_{i0}\}$ without enforcing positive definiteness: the evaluation of $\mathcal{F}$ and $\mathcal{Q}$ does not require this condition. The objective functions decouple over the entries of Σ and hence each entry σ_{mn} can be optimized separately – by searching over the discretized space $\{-0.99, -0.97, \ldots, 0.97, 0.99\}$, as required to allow for the caching of $w_k(\sigma_{mn})$.

3.4 Bivariate Residual Search

An alternative to the expensive iterative methods from the previous section is to do a single hypothesis test for each bi-directed edge. If one knew parameters $\{\beta, \Sigma\}$, a possibility is to do a χ^2-like measure of fitness of the implied bivariate contingency table for each $\{Y_i, Y_j\}$, and add the corresponding bi-directed edge if there is evidence of misfit. Without knowledge of $\{\beta, \Sigma\}$, a possibility is to fit the model without bi-directed edges as a surrogate, in the hope that such estimates are good enough to detect misfit. In Algorithm 3, we outline a procedure where the test of misfit is to choose between two models with the single edge $Y_i \leftrightarrow Y_j$ and none (our definition of the space $\mathcal{G}_m^{ij(+/-)}$) based on $PCL(\cdot, \cdot, \cdot)$ – essentially a Bayesian test for the edge in the bivariate model $\mathcal{P}(Y_i, Y_j \mid \beta, \Sigma)$. The final graph is given by the union of all edges that were selected by the bivariate tests. The shortcoming of course is that the initial parameter estimate might be bad if there are reasonably strong dependencies due to the unaccounted latent variables. Iterating the procedure, however, gives a procedure that is essentially Algorithm 1, albeit with a "parallel" testing of the edges. For simplicity, we do not consider the continuum between sequential and parallel tests (e.g., methods where initial iterations would modify one edge only per parameter update, allowing multiple edge modifications later on) and evaluate Algorithm 3 only.

Algorithm 3 Single-shot Structured CCA Learning

procedure LEARNSTRUCTUREDCCA-0($\mathcal{S}$,$\mathcal{D}$)
  $\{\mathbf{Y}, \mathbf{X}_\mathcal{S}, \mathcal{G}_m\} \leftarrow$ GETDAG($\mathcal{S}$)
  $\{\beta, \Sigma\} \leftarrow \arg\max_{(\Omega_\beta, \Omega_\Sigma)} PCL(\mathcal{G}_m, \beta, \Sigma)$
  for $1 \leq i < j \leq |\mathbf{Y}|$ **do**
    $\mathcal{G}_m^{ij} \leftarrow \arg\max_{(\mathcal{G}_m^{ij(+/-)})} PCL(\mathcal{G}_m, \beta, \Sigma)$
  end for
  return $\cup_{ij} \mathcal{G}_m^{ij}$
end procedure

4 RELATED WORK

More conventional uses of EM in the context of composite likelihod have been discussed by Varin et al. (2011) and Gao and Song (2011). CCA has been applied in other contexts in machine learning, such as analysing text data under different languages (Hardoon et al., 2004). This setup and usage is very different from our motivation, which focus on applications with several small sets of measurements, and where the special structure of the latent space (a one-to-one association between latent variables and groups of observable variables) is motivated by particular applications in measurement error problems (Bollen, 1989). Concerning model selection in structured latent spaces, Silva and Ghahramani (2009) provides an approach based on Gaussian distributions, and a Gaussian copula method can be readily applied to the binary modeling case. However, we want to avoid a full likelihood approach for scalability reasons. While Silva and Ghahramani (2009) provides priors over sparse covariance matrices, it is not clear what

[4]In order to reduce the number of parameters, we calculate each intercept β_{i0} as a function of the slope β_{i1} such that the marginal probability $\mathcal{P}(Y_i = 0 \mid \beta, \Sigma)$ matches the empirical probability. Hence the only free parameters are the slopes $\{\beta_{i1}\}$.

the marginals of such priors are over subsets of the observations: sparse inverse Wishart priors do not have a closed form for their marginals. This makes the Bayesian Gaussian approach seemingly hard to apply in the composite likelihood scenario. The use of composite likelihood methods in model selection (Varin and Vidoni, 2005) has been increasing in recent years, including variations where parameters across different likelihood components do not need to be constrained to be the same: a postprocessing method can be used to combine estimates from different regions into a single point estimate (e.g., Meinshausen and Buehlmann, 2006). Further consequences of ignoring association due to unspecified factors in a latent variable model are discussed by Westfall et al. (2012). It is also important to mention that the method here introduced generalizes the learning methods of Huang and Frey (2011), since the CDN is a special case. It is also an alternative to maximum likelihood estimation for networks of large tree-width.

5 EVALUATION

We evaluate the approach with a synthetic and a real-world experiment. The prior over graphical structures is given by an independent probability of 0.1 for each bi-directed edge. We use Frank copulas for the dependence structure (Nelsen, 2007). The prior for parameter θ_{ij} is defined by first sampling a standard Gaussian $Z \sim \mathcal{N}(0,1)$ and squashing it as $\theta_{ij} = [1/(1+e^{-z})] \times 50 - 25$, which generates θ_{ij} in the interval $[-25, 25]$ – beyond which the Frank copula gives numerically unstable results in our code.

5.1 Synthetic Studies

We simulate[5] 20 networks with 4 latent variables and 4 children per latent variable. Bi-directed edges were added independently with probability 0.2. The network is then pruned randomly so that each latent variable d-separates at least three children, and that each observed variable has no more than 3 adjacent bi-directed edges. The average number of resulting edges was approximately 18. Results for three evaluation measures are shown in Figure 2. We compare the three methods of Section 3, doing a comparison at three different sample sizes (1000, 5000 and 10000). We show the average paired difference over 20 trials between LEARNSTRUCTUREDCCA-II (LSC-II) and LSC-I, and between LSC-II and LSC-0. The first criterion is the root mean squared error of the average slope coefficients $\{\beta_1\}$ with respect to the true model; the second criterion is "edge omission," the number of incorrectly removed edges divided by the total number of edges; the third criterion is "edge commission," incorrect addition of an edge divided by the total number of possible additions. The number of times where the difference is positive with the corresponding p-values for a Wilcoxon signed rank test are given below (stars indicate numbers less than 0.05):

	1000		5000		10000	
Slope	I	0	I	0	I	0
number	13	6	17	15	15	13
p-value	0.22	0.25	*	*	*	0.06
Omission	I	0	I	0	I	0
number	11	18	6	14	6	9
p-value	0.17	*	0.82	*	0.62	0.22
Commision	I	0	I	0	I	0
number	5	2	15	16	16	18
p-value	0.28	*	*	*	*	*

The upshot is that Algorithm 2 does at least as well as the others concerning edge omission, but with substantially fewer false positives – a problem that particularly affects the single-shot algorithm. Algorithm 2 is also more robust concerning parameter estimation. It has to be said, however, that all methods do badly concerning edge omission at sample size 1000: in more than 10 trials, Algorithm 2 had edge omission rates over 0.5. At sample size 5000, this substantially decreased (17 trials under 0.2, 10 under 0.15). Algorithm 2 commission errors are typically low (< 0.05) for all datasets. It is also relevant that Algorithm 2 typically converges faster than Algorithm 1 despite the extra step on approximating posteriors with the Laplace approximation.

5.2 Analysis of the NHS Staff Survey

We present a simple application to the modeling of response patterns in the 2009 National NHS Staff Survey (Care Quality Comission and Aston University, 2010). We now emphasize less an evaluation of the structure and more the initial goal described in Section 2.2 on showing how finding bi-directed structures helps in estimating functionals of $\mathcal{P}(\mathbf{X}_{\mathcal{S}} \mid \mathbf{Y})$. The NHS (National Health Service) is the public healthcare system of the United Kingdom. A survey concerning different aspects of job satisfaction, work conditions, training and other factors takes place regularly. We provide an analysis of the 2009 survey, which was returned by 156,951 respondents. Under the assumption that sections in the questionnaire translate to particular trait factors, we evaluate how much is gained by allowing a latent variable model to be adaptive to external factors not originally included in the model. Responses to questions are either binary Yes/No questions or encoded in an ordinal scale (varying in 5 points from strong disagreement to strong agreement). We en-

[5]Other details about the simulation: a "signal factor" (the ratio between the variance β_{i1}^2 implied by the Gaussian X_m, and $\beta_{i1}^2 + 1$ given by the added variance of the probit model) is the sampled uniformly within the interval $[0.2, 0.6]$ for each variable Y_i. Slope β_{i1} is then set accordingly as a function of the signal factor, with its sign chosen with probability 0.5. A "marginal factor" (the marginal probability $\mathcal{P}(Y_i = 0)$) is uniformly sampled within the interval $[0.1, 0.9]$, and intercept β_{i0} is set according to the marginal factor and the sampled value of β_{i1}. We sample each copula parameter independently, uniformly in the interval $[10, 15]$. Finally, we generate Σ by rescaling as a correlation matrix the result of $\sum_{k=1}^{4} z_k z_k^{\mathsf{T}}$, where $Z_K \sim \mathcal{N}(0,1)$.

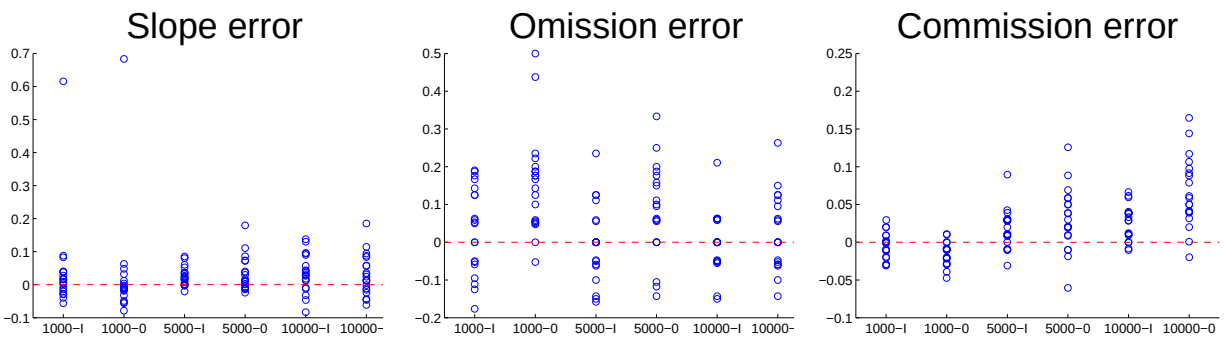


Figure 2: Differences between LEARNSTRUCTUREDCCA-II (LSC-II) and LSC-I are labeled, at three different samples sizes, as 1000-I, 5000-I and 10000-I. Differences between LSC-II and LSC-0 are labeled analogously.

coded ordinal data as binary, assigning the value of 1 to "Agree" and "Strongly agree" responses, and 0 otherwise (including missing or not applicable responses). We generated 9 factors out of the questionnaire with a total of 50 observed variables. The selection and partitioning process is described in detail in the Supplementary Material. We randomly selected 100,000 respondents as a training set and fit a model using LEARNSTRUCTUREDCCA-II. A total of about 40 bi-directed edges was generated. We also fit the model without any bi-directed edges. Since we cannot calculate marginal likelihoods easily as a way of comparing the models, we resort to evaluating the predictive ability of the latent representations. Given these two models, we generate latent embeddings of the observations[6] in the test set by maximizing $\sum_{ij} \log \mathcal{P}(Y_i^{(d)}, Y_j^{(d)}, X_i^{(d)}, X_j^{(d)} \mid \beta, \Sigma, \mathcal{G}_m)$, for each data point d, with respect to the latent variables. Here X_i and X_j are the (possibly unique) latent parents of Y_i and Y_j. This results in over 50,000 points in a 9-dimensional space. Half of these points were used to build 11 logistic regression models to predict answers to questions not included in the model[7]. We calculated the area under the curve for predictions done in the test set of approximately 28,000 points. Results are shown in the Table 2, which illustrates that the mixed model does at least as well or better at generating a latent representation of the observations, while preserving the interpretability of the model.

Table 2: Comparison of the mixed graph CCA model (MCCA) against the standard, fixed structure, model (SCCA) in 11 binary classification tasks, as measured by the area under the curve.

	MCCA	SCCA		MCCA	SCCA
Q1	0.71	0.71	Q7	0.80	0.79
Q2	0.75	0.75	Q8	0.82	0.81
Q3	0.86	0.82	Q9	0.86	0.83
Q4	0.90	0.82	Q10	0.69	0.69
Q5	0.79	0.80	Q11	0.78	0.75
Q6	0.73	0.72			

6 CONCLUSION

We summarize the main features of our method:

- it searchs only over a projection of an infinite dimensional latent space into a mixed graph structure, instead of explicitly adding latent variables. Conditioned on $\mathbf{X}_S$, the composite likelihood function can be calculated analytically. This is possible due to the *implicit latent variable representation* of the CDN;
- assuming no missing data, it requires only the sufficient statistics for regions of bounded size, computable with a single pass through the data, possible due to the *marginalization property* of the CDN;
- optimization is unconstrained, thanks to the *parameter modularity* property of the copula formulation;
- it allows for the use of simple deterministic integration methods, while still providing a mechanism to propagate information beyond simple pairs of observed variables – a *novel variation of composite likelihood estimation*, to the best of our knowledge.

Although the approach scales well in the sample size, the parameter fitting step is still particularly expensive in high-dimensions, as also noted by Joreskog and Moustaki (2001). A more efficient algorithm can be achieved by a smarter implementation of the gradient optimization method, or by relaxing the restriction that parameter estimates need to be the same across different likelihood components (Meinshausen and Buehlmann, 2006). Moreover, having an adaptive structure for the partition of the random variables is also an important direction.

[6] After fitting both models, 5 of the observed variables were given extreme coefficients in their measurement equations (in both models). In order to make the embedding process numerically stable, we removed these 5 variables from the testing.

[7] Questions concerned job satisfaction. See Supp. Material.

Acknowledgements

The author would like to thank the three anonymous reviewers for very many helpful comments that inspired improvements in the presentation of the method and results.

References

F. Bach and M. I. Jordan. A probabilistic interpretation of canonical correlation analysis. *Technical Report 688, Department of Statistics, University of California, Berkeley*, 2005.

D. Bartholomew and M. Knott. *Latent Variable Models and Factor Analysis*. Arnold Publishers, 1999.

K. Bollen. *Structural Equations with Latent Variables*. John Wiley & Sons, 1989.

Care Quality Commission and Aston University. Aston Business School, National Health Service National Staff Survey, 2009 [computer file]. *Colchester, Essex: UK Data Archive [distributor], October 2010. Available at* HTTPS://WWW.ESDS.AC.UK, *SN: 6570*, 2010.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–37, 1977.

G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: a structure-based approach. *Neural Information Processing Systems*, 13:479–485, 2000.

N. Friedman. The Bayesian structural EM algorithm. *Proceedings of 14th Conference on Uncertainty in Artificial Intelligence (UAI '98)*, 1998.

X. Gao and P.X-K. Song. Composite likelihood EM algorithm with applications to multivariate hidden Markov model. *Statistica Sinica*, 21:165–186, 2011.

D. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16:2639–2664, 2004.

J. Huang and B. Frey. Cumulative distribution networks and the derivative-sum-product algorithm. *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, 2008.

J. Huang and B. Frey. Cumulative distribution networks and the derivative-sum-product algorithm: Models and inference for cumulative distribution functions on graphs. *Journal of Machine Learning Research*, 12: 301–348, 2011.

J. Huang, N. Jojic, and C. Meek. Exact inference and learning for cumulative distribution functions on loopy graphs. *Advances in Neural Information Processing Systems*, 23, 2010.

H. Joe. *Multivariate Models and Dependence Concepts*. Chapman-Hall, 1997.

K. Joreskog and I. Moustaki. Factor analysis for ordinal variables: a comparison of three approaches. *Multivariate Behavioural Research*, 36:347–387, 2001.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

N. Meinshausen and P. Buehlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34:1436–1462, 2006.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. *In M. Jordan (Ed.), Learning in Graphical Models*, pages 355–368, 1998.

R. Nelsen. *An Introduction to Copulas*. Springer-Verlag, 2007.

T. Richardson and P. Spirtes. Ancestral graph Markov models. *Annals of Statistics*, 30:962–1030, 2002.

R. Silva and Z. Ghahramani. The hidden life of latent variables: Bayesian learning with mixed graph models. *Journal of Machine Learning Research*, 10:1187–1238, 2009.

R. Silva, R. Scheines, C. Glymour, and P. Spirtes. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7:191–246, 2006.

R. Silva, C. Blundell, and Y. W. Teh. Mixed cumulative distribution networks. *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS '11)*, 2011.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Cambridge University Press, 2000.

C. Varin and P. Vidoni. A note on composite likelihood inference and model selection. *Biometrika*, 92(3):519–528, 2005.

C. Varin, N. Reid, and D. Firth. An overview of composite likelihood methods. *Statistica Sinica*, 21:5–42, 2011.

J. Wegelin, T. Richardson, and D. Ragozin. Rank-one latent models for cross-covariance. *Technical Report no. 391, Department of Statistics, University of Washington*, 2001.

J. Wegelin, A. Packer, and T. Richardson. Rank-r latent models for cross-covariance. *Technical Report no. 411, Department of Statistics, University of Washington*, 2002.

P. Westfall, K. Henning, and R. Howell. The effect of error correlation on interfactor correlation in psychometric measurement. *Structural Equation Modeling: A Multidisciplinary Journal*, 19:99–117, 2012.

F. Wood, T. Griffiths, and Z. Ghahramani. A nonparametric Bayesian method for inferring hidden causes. *Uncertainty in Artificial Intelligence*, 2006.

Spectrum Identification using a Dynamic Bayesian Network Model of Tandem Mass Spectra

Ajit P. Singh[†] **John Halloran**[†] **Jeff A. Bilmes**[†] **Katrin Kirchoff**[†] **William S. Noble**[‡]
[†]Department of Electrical Engineering
University of Washington
Seattle, WA 98195
[‡]Department of Genome Sciences
University of Washington
Seattle, WA 98195

Abstract

Shotgun proteomics is a high-throughput technology used to identify unknown proteins in a complex mixture. At the heart of this process is a prediction task, the *spectrum identification problem*, in which each fragmentation spectrum produced by a shotgun proteomics experiment must be mapped to the peptide (protein subsequence) which generated the spectrum. We propose a new algorithm for spectrum identification, based on dynamic Bayesian networks, which significantly outperforms the de-facto standard tools for this task: SEQUEST and Mascot.

1 Introduction

Shotgun proteomics is the dominant technology used to identify which proteins are present in a cell or tissue sample, and is widely used in the biological sciences (Marcotte, 2007; Steen and Mann, 2004). At the heart of shotgun proteomics is a machine learning problem. Proteins are broken down into small fragments, called peptides. Although shotgun proteomics cannot directly determine the sequence of these peptides, the technology can rapidly generate indirect information about peptide sequences, called fragmentation spectra. The task of identifying the peptide string responsible for generating an observed fragmentation spectrum is known as the *spectrum identification problem*. This problem is similar in form to speech recognition, in which a spoken utterance (fragmentation spectrum) is mapped to its corresponding natural language string (peptide sequence).

This paper applies dynamic Bayesian networks (DBNs) to the spectrum identification problem. We draw on ideas from the graphical models community to build an algorithm for spectrum identification which is significantly more accurate than the most popular tools in wet-lab use, SEQUEST (Eng et al., 1994) and Mascot (Perkins et al., 1999), as well as other representative tools which have been proposed for spectrum identification. We call our algorithm Didea[1].

2 Background

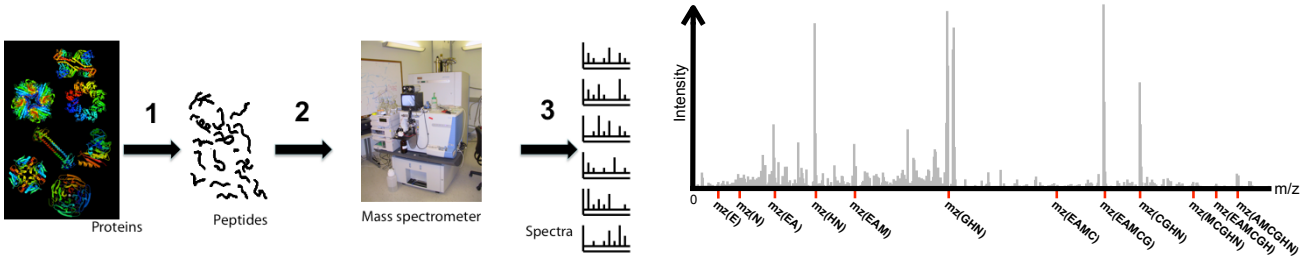
Spectrum identification is a machine learning problem, but like many problems in computational biology, one which has a high barrier to entry for computer scientists. Therefore, we present a self-contained introduction to shotgun proteomics, explaining how peptides generate fragmentation spectra (Section 2.1). One of the reasons for using a dynamic Bayesian network is that it allows us to use qualitative knowledge about the physics of peptide fragmentation (in the structure of the DBN) and a small number of trainable parameters to improve predictive power.

We review the literature on spectrum identification, which falls into two broad categories: database search (Section 2.2) and *de novo* identification (Section 2.3). Database search uses additional biological information about the sample being analyzed to constrain the statistical complexity of spectrum identification. Database search is more accurate than *de novo*, and is the preferred technique in practice (Kim et al., 2010). The two most popular tools in wet-lab practice, SEQUEST and Mascot, are both database search tools, as is Didea.

2.1 Shotgun Proteomics

A typical shotgun proteomics experiment proceeds in three steps, as illustrated in Figure 1(a). The input to the experiment is a collection of proteins, which have been isolated from a complex mixture. Each protein can be represented as a string of amino acids, where the alphabet is size 20 and the proteins range in length from 50–1500 amino acids. A typical complex mixture may

[1] A portmanteau of the words 'Dynamic', 'Peptide', and 'Algorithm'.



(a) The three steps—(1) cleaving proteins into peptides, (2) separation of the peptides using liquid chromatography, and (3) tandem mass spectrometry analysis.

(b) Red ticks indicate where we expect to detect ions from the fragmentation of peptide EAMCGHN.

Figure 1: The schematic for a typical shotgun proteomics experiment (a), with an example of a real fragmentation spectrum annotated by the theoretical spectrum, in red, of a candidate peptide EAMCGHN (b).

contain a few thousand proteins, ranging in abundance from tens to hundreds of thousands of copies.

In the first experimental step, the proteins are digested into peptides, or protein subsequences, using a molecular agent called trypsin. On average, the length of the peptides for the model organisms we consider is 14-15, with no peptides being longer than 50 amino acids. Digestion is necessary because whole proteins are too massive to be subject to direct mass spectrometry analysis without using very expensive equipment. In the second experimental step, peptides are subjected to a process called *liquid chromatography*, in which the peptides pass through a thin glass column that separates the mixture of peptides based on a particular chemical property (e.g., hydrophobicity). This separation step reduces the complexity of the mixtures of peptides going into a mass spectrometer. The third experimental step, which occurs inside the mass spectrometer, involves two rounds of mass spectrometry. Approximately every second, the device analyzes the population of approximately 20,000 intact peptides that most recently exited from the liquid chromatography column. Then, based on this initial analysis, the machine selects five distinct peptide species for fragmentation. Each of these fragmented species is subjected to a second round of mass spectrometry analysis. The resulting "fragmentation spectra" are the primary output of a shotgun proteomics experiment.

A fragmentation spectrum is shown in Figure 1(b). We explain how such a spectrum is generated using a concrete example. Assume that, from the population of 20,000 intact peptides, we isolate a distinct peptide species: a collection of peptide molecules, each with sequence EAMCGHN. Each peptide molecule has extra protons, which allows it to be isolated and accelerated using magnetic fields. Molecules with extra protons are called ions; the number of extra protons, its charge. These isolated peptide molecules are collided into a neutral gas, which causes each molecule to break, typically along one of the amino acid bonds at random. For a peptide of length n, there are $n-1$ bonds which can be broken, each yielding a (prefix, suffix) pair: e.g., (E, AMCGHN), (EA, MCGHN), ..., (EAMCGH, N). When an amino acid bond is broken, the extra protons migrate to either the prefix or suffix, at random. The charged prefix is called a b-ion; the charged suffix is called a y-ion. Collectively, the ionized products of fragmentation are called product ions. We assume all product ions are either b- or y-ions. The sequence of peptides or product ions cannot be directly measured, but we can measure how often product ions with a particular mass-to-charge (m/z) ratio are detected. These measurements are represented in a plot with m/z (measured in Daltons) on the horizontal axis and intensity (unitless, but roughly proportional to ion abundance) on the vertical axis.

Real fragmentation spectra are noisy. The isolation of peptide species is imperfect, so fragmentation spectra can contain product ions from peptides with different sequences (chimeric spectra). Mass analyzers can measure subatomic mass differences, so even the smallest unfiltered contaminant adds noise to the fragmentation spectrum. Product ions without charge cannot be detected, and we do not always know how many protons are adopted by a product ion—i.e., variable charge. There are a host of secondary fragmentations and degradations of product ions, which are measured. Finally, intensity measurements are quite noisy.

The input to the spectrum identification problem is a fragmentation spectrum, along with the observed (approximate) mass of the intact peptide whose fragmentation produced the spectrum. The output is the sequence of the unknown peptide, the fragmentation of which generated the spectrum. A peptide paired with a spectrum is called a peptide-spectrum match (PSM).

2.2 Database Search

A tandem mass spectrometry experiment produces a set of fragmentation spectra $S = \{s_1, \ldots, s_r\}$. Each spectrum is also associated with a measurement of the mass of the unknown peptide which generated the spectrum, $m(s_i)$. Let U be the universe of all peptides. We assume as input a database of possible peptides, $P \subset U$. The key assumptions behind database search are that (i) we know the organisms from which the proteins came from, and (ii) we have a set of all known proteins for the organism, which can be computed given the organism's genome. We need only search over peptides in P, not all peptides in U. Formally, spectrum identification is the task of assigning a peptide $p_i \in P$ to each spectrum s_i. Let $\Psi : S \times P \to \mathbb{R}$ denote a scoring function, where a higher score corresponds to a higher confidence that a peptide-spectrum match is correct (i.e., the peptide generated the spectrum).

Since we have measured $m(s_i)$, we can further constrain the search space over peptides to those whose mass is close to $m(s_i)$, a set of candidate peptides,

$$C(s_i, P, \delta) = \{p : p \in P, |m(p) - m(s_i)| < \delta\}, \quad (1)$$

where $m(p)$ is the mass of the peptide p. In our experiments, we use $\delta = 3.0$ Daltons. The database search itself involves scoring all candidate peptides, returning the highest scoring one for each spectrum. For $i = 1 \ldots r$,

$$p_i = \underset{p \in C(s_i, P, \delta)}{\operatorname{argmax}} \Psi(s_i, p). \quad (2)$$

The first computer program to use a database search procedure to identify fragmentation spectra was SEQUEST (Eng et al., 1994), whose scoring function is defined in terms of inner products between the quantized observed spectrum and a theoretical spectrum $\phi(p)$ generated from a simple model of peptide fragmentation. Observed and theoretical spectra are vectors of real numbers, so a peptide can be compared to a spectrum using inner products,

$$\begin{aligned} \text{XCorr}(s, \phi(p)) &= \alpha_x - \beta_x, \quad \text{where} \\ \alpha_x = \langle s, \phi(p) \rangle, \;\; \beta_x &= \frac{1}{150} \sum_{\tau=-75}^{75} \langle s, \phi_\tau(p) \rangle, \end{aligned} \quad (3)$$

and where $\phi_\tau(p)$ is just $\phi(p)$ shifted by $|\tau|$ units to the right ($\tau > 0$) or left ($\tau < 0$). Intuitively, a peptide is a good match if the observed spectrum s is highly correlated to the theoretical spectrum (α_x high), but not to shifted versions of the theoretical spectrum (β_x low). Didea's scoring function has an analogous $\alpha - \beta$ representation, where the analogues of correlation (α) and cross-correlation (β) are calculated using graphical model inference (Section 3.2).

There are a number of different tools for database search: e.g., SEQUEST (Eng et al., 1994), Mascot (Perkins et al., 1999), OMMSA (Geer et al., 2004), X!Tandem (Craig and Beavis, 2004), PepHMM (Wan et al., 2006), Riptide (Klammer et al., 2008), Andromeda (Cox et al., 2011), InsPecT (Tanner et al., 2005) and MS-GFDB (Kim et al., 2010). Separate from these approaches are post-processors, such as PeptideProphet (Keller et al., 2002) and Percolator (Käll et al., 2007), which refine the scores of a set of prescored peptide-spectrum matches, instead of scoring spectra in a streaming fashion. We note that statistical tools also exist for identifying proteins from peptide identifications, which can also refine peptide scores like a post-processor (e.g., (Li et al., 2010)). Some of the tools cited do not have implementations available for benchmarking (e.g., Riptide). Indeed, each of the above papers supports its predictive power claims by benchmarking against a representative subset of prior work. We have opted for a large set of competitors, on a diverse panel of data sets: SEQUEST, Mascot, MS-GFDB, OMMSA. SEQUEST and Mascot are the two most popular tools in wet-lab practice: e.g., each has been cited over 3000 times in Google Scholar, and both are commercial products.[2] MS-GFDB is a new scoring tool; the authors have demonstrated superior performance on the same kinds of fragmentation spectra that we study (collision-induced dissociation of tryptic peptides) against many of the tools listed above. OMMSA is open-source, and readily available. We do not compare against post-processors, as they use information not available to the other tools.

2.3 De Novo vs. Database Search

Early work on spectrum identification framed the problem as an exhaustive search over all possible peptides (Sakurai et al., 1984), an approach which came to be known as *de novo* spectrum identification, a.k.a. de novo peptide sequencing (Bafna and Edwards, 2003; Bandeira et al., 2008; Bartels, 1990; Bern and Goldberg, 2005; Bhatia et al., 2011; Dancik et al., 1999a,b; Datta and Bern, 2008; Fischer et al., 2004; Frank and Pevzner, 2005; Frank et al., 2005; Jeong et al., 2010). De novo tools select all candidate peptides that are within a mass tolerance of $m(s_i)$, not just those peptides that occur in a per-organism peptide database P.

De novo and database search methods cannot be equitably compared. Database search uses knowledge about the organism from which the protein sample is drawn, as well as knowledge of the proteome of that organism. De novo does not assume that such information is available. De novo searches over pep-

[2]Mascot is a black box, the details of its spectrum identification engine, as currently implemented, are proprietary.

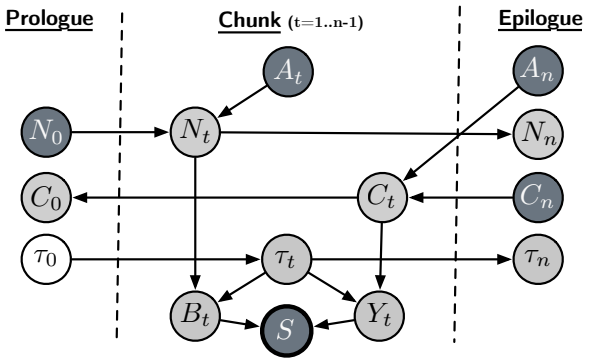


Figure 2: Basic Didea model. Dark gray nodes are observed random variables, light gray nodes are variables whose value is a deterministic function of its parents, and unshaded nodes are hidden random variables. Bold circles indicate vectors of random variables.

tides in $C(s_i, U, \delta)$; database search searches over peptides in $C(s_i, P, \delta)$, where for any reasonable value of δ, $|C(s_i, P, \delta)| \ll |C(s_i, U, \delta)|$.

3 Generative Model of Peptide Fragmentation

Didea encodes much of the physical process of peptide fragmentation by collision-induced dissociation within a dynamic Bayesian network. To maximize predictive performance, we take advantage of the idea of spectrum shifting (Equation 3) within our model, to derive a scoring function for peptide-spectrum matches based on inference within the DBN. The resulting inference-based scoring function can be viewed as an $\alpha - \beta$ score, where instead of dot products, we use the DBN to induce analogues of correlation and cross-correlation.

Figure 2 shows the basic Didea model; Sections 3.1–3.3 describe all the conditional probability distributions. Many of the conditional probability distributions are deterministic, which allows the scoring function to be computed in polynomial time.

An advantage of Didea is that adding more knowledge to the model (e.g., Sections 3.4-3.5) yields a new scoring function without any changes to the code. We just compute the same posterior on a new model.

3.1 Mapping the peptide to theoretical product ions

A DBN is a structured probability distribution over variable-length sequences of random variables, and is a strict generalization of the more familiar hidden Markov model (HMM). In Didea, a peptide containing n amino acids is represented by a sequence of random variables $(A_t : t = 1 \dots n)$. Each variable A_t takes on the value of one of the 20 standard amino acids. In database search the peptide is given, so the amino acid variables are observed: $\mathbf{p} = (A_t = a_t : t = 1 \dots n)$. To simplify the main text, we equivalently refer to a peptide as a sequence of amino acids $p = a_1 a_2 \dots a_n$.

In collision-induced dissociation, most of the product ions are either b- or y-ions. If the peptide p is cleaved between a_t and a_{t+1}, then the resulting (respectively b- and y-) product ions are referred to as $b_t = a_1 \dots a_t$ and $y_t = a_{t+1} \dots a_n$. We use the function $c(\cdot)$ to denote the charge of a peptide, $c(p)$; the charge of the product ions, $c(b_t)$ and $c(y_t)$; and the observed charge(s) of the fragmentation spectrum, $c(s) \subseteq \{+1, +2, +3\}$.[3] For now, we assume that $c(p) = +2$ (two adopted protons) and that the product ions each adopt one proton from the peptide during fragmentation, i.e., $\forall t, c(b_t) = c(y_t) = +1$. These charge assumptions simplify exposition, and are relaxed in Section 3.5.

The structure of the Didea model is shown in Figure 2. Random variables are grouped into *frames*, indexed by $t = 0 \dots n$. The first frame (frame 0) is referred to as the prologue; the last frame (frame n), the epilogue. The variables in the prologue and epilogue do not themselves correspond to b- or y-ions, but do contain variables needed to define recursions used to compute the mass of product ions. For a peptide of length n, the chunk frame is repeated once for $a_1, \dots, a_{n-1}$, in the same way that variables are repeated in an HMM to accommodate variable length sequences.

A product ion (a.k.a. fragmentation) spectrum s is a collection of peaks $s = \{(x_j, h_j)\}_j$, where x_j is a point on the mass-to-charge, or m/z, axis (x-axis), and h_j is the corresponding intensity. To check whether the b_t or y_t ion appears in the observed spectrum, we need the mass of each product ion. The neutral masses of the b_t and y_t ions are represented by random variables $N_t \equiv m(A_1 \dots A_t)$ and $C_t \equiv m(A_{t+1} \dots A_n)$, where $m(\cdot)$ returns the neutral masses of a sequence of amino acids. In Didea, the mass of each amino acid is rounded to the nearest whole Dalton, so that N_t and C_t are multinomial random variables over the neutral mass of the b_t and y_t ions—$N_t, C_t \in \{0, \dots, D\} \subset \mathbb{Z}_{++}$. By definition, $N_0 = 0$ and $C_n = 0$, so that the masses of the product ions can be computed recursively: $N_t = N_{t-1} + m(A_t)$, $C_t = C_{t+1} + m(A_t)$. Note that $C_0 = N_n = m(A_1 \dots A_n)$. Unlike HMMs, DBNs can have both left-to-right and right-to-left arcs (as long as there are no directed cycles), which allows us to implement the two recursions in the same model. Since $\{A_t\}_t$ are observed, both N_t and C_t become deterministic

[3] In our experiments, the mass spectrometer operates in positive-ion mode, and the reported charge of each spectrum is either $c(s) = +1$, $c(s) = +2$, or $c(s) \in \{+2, +3\}$.

random variables: $p(N_t = m(a_1 \dots a_t)) = 1$ and $p(C_t = m(a_{t+1} \dots a_n)) = 1$.

3.2 Mapping theoretical product ions to the fragmentation spectrum

We now have the mass of each product ion, and we have assumed that the charge of each product ion is +1. To finish the basic Didea model, we need to *(i)* encode the spectrum as observed random variables, *(ii)* introduce random variables which map from mass and charge to a specific m/z region in the spectrum, and *(iii)* define a conditional probability distribution which measures the value of finding a peak of intensity h in the m/z region where a product ion is expected to be.

From the settings used to collect our spectra, we know that $x_j \in [0, 2000]$ m/z units. For simplicity, we quantize the m/z scanning range into bins that are 1Da wide. In Figure 2, the bins correspond to a vector of random variables $\mathbf{S} = (S_i : i = 1 \dots B = 2000)$. A spectrum is an instantiation of the random variables $\mathbf{s} = (S_i = s_i : i = 1 \dots B)$, where each $s_i \geq 0$.

Spectra may differ by orders of magnitude in both total intensity ($\sum_j h_j$) and maximum intensity ($\max\{h_j\}$). To control for intensity variation, we rank-normalize each spectrum: peaks are sorted in order of increasing intensity, and the i^{th} peak is assigned intensity $i/|s|$, so $\max\{h_j\} = 1.0$. If bin i contains no peak, then $s_i = 0$. Otherwise, s_i is the highest rank-normalized peak, so $s_i \in [0, 1]$. We represent the vector of random variables $\mathbf{S}$ using a single, bold-edged node in Figure 2.

Each repetition of the chunk frame represents one pair of b- and y-ions, denoted (b_t, y_t). The random variables $B_t, Y_t \in \{1, \dots B\}$ represent, respectively, the bin where the b_t and y_t ions are to be expected.[4] We want to be able to shift the theoretical spectrum, in much the same fashion as Equation 3. We do so using the discrete random variables $\tau_t \in [-M \dots + M]$, for some $M \in \{1 \dots B\}$ (we use $M = 37$). We expect to see a peak corresponding to the b_t ion in S_{B_t}, and a peak corresponding to the y_t ion in S_{Y_t}. The mass-to-m/z mappings are $B_t = \text{round}(N_t + 1)$ and $Y_t = \text{round}(C_t + 19)$, where 1 is the mass of a proton, and 19 the mass of a proton plus a water molecule. τ_t shifts all the theoretical product ion peaks. To ensure that $B_t, Y_y \in [0, \dots B]$, values less than 1 are changed to 1; values greater than B are changed to B. To ensure that all the product ion peaks are shifted by the same amount, we copy the value of τ_0 from the prologue frame into each subsequent one: $\forall t = 1 \dots n$, $\tau_t = \tau_{t-1}$. As will be seen in Equation 6, to mimic the arithmetic average over shifts in the background term in XCorr, we assume that τ_0 is uniformly distributed over all shifts, $p(\tau_0) = (2M + 1)^{-1}$.

Most of the conditional probability distributions in Figure 2 are deterministic, which leads to a concise form for the joint distribution. For now, we refer to the contribution of a spectrum peak to the likelihood as non-negative function g:

$$p(\tau_0, \mathbf{s}, \mathbf{p}) = p(\tau_0) \prod_{t=1}^{n-1} \prod_{i=1}^{B} g(S, i, b_t, y_t, \tau_t), \tag{4}$$

$$g(\cdots) \triangleq P(S_i \,|\, b_t, y_t, \tau_t)^{\mathbf{1}(i=b_t+\tau_t \vee i=y_t+\tau_t)}. \tag{5}$$

Equation 4 is the likelihood of the model in Figure 2.

The values of B_t and Y_t, for each t, are deterministically determined from the peptide sequence. Assignment $\{B_t = b_t\}_t$ forces the model to assign some score, based on the intensity of the peaks found in $\{S_{b_t}\}_t$ (likewise with $\{Y_t = y_t\}_t$). The choice of $P(S_i \,|\, b_t, y_t, \tau_t)$ determines how the spectrum influences the score; details are deferred to Section 3.3.

One way to use the proposed model to score a PSM would be to set $\tau_0 = 0$ and use Equation 4, which would be analogous to dropping the β_x term in XCorr. Instead, to achieve an analogue to XCorr, we propose a score that can be interpreted as the difference between the Didea analogues of spectrum correlation (α_d) and cross-correlation (β_d):

$$\begin{aligned} \theta(\mathbf{s}, \mathbf{p}) &\triangleq \log p(\tau_0 = 0 \,|\, \mathbf{p}, \mathbf{s}) = \alpha_d - \beta_d, \\ \alpha_d &= \log p(\tau_0 = 0, \mathbf{p}, \mathbf{s}), \\ \beta_d &= \log p(\tau_0) \sum_{\tau_0} p(\mathbf{p}, \mathbf{s} \,|\, \tau_0). \end{aligned} \tag{6}$$

Both α_x and α_d are biased, assigning higher scores to spectra that tend to be close matches to many peptides (e.g., spectra with more non-zero peaks will tend to have higher α, regardless of which peptide is evaluated against it). The β_d term, like the β_x term, penalizes spectra which tend to match many peptides, which is why $\alpha - \beta$ scores tend to be more discriminative than α. Equation 6 is a (posterior) probability; Equation 3 is not probabilistic.

3.3 Modeling spectrum peak intensity

Here, we define Equation 5, which determines the influence of the spectrum on the likelihood, and thus the PSM score $\theta(\mathbf{s}, \mathbf{p})$. The cost of inference is proportional to the number of times $P(S_i \,|\, b_t, y_t, \tau_t)$ is evaluated. Since the spectrum is fixed across all $t = 1 \dots n - 1$, the values of $P(S_i \,|\, b_t, y_t, \tau_t)$, for all $(S_i = s_i, b_t, y_t, \tau_t)$, can be precomputed and stored in a lookup table:

[4] Since there is no ambiguity, we also use b_t and y_t to represent assignments to random variables, $B_t = b_t$ and $Y_t = y_t$.

$w : \{1 \dots B\} \to \mathbb{R}_+$, a transformation of the spectrum into non-negative weights. When a predicted product ion matches peak s_i in the spectrum, $w(i)$ is the contribution of that peak match to the PSM score.

In a graphical model, any conditional probability distribution on discrete variables can be reparameterized into a lookup table using virtual evidence (Pearl, 1998), allowing us to re-express Equation 4 as

$$p(\tau_0, \mathbf{p}, \mathbf{s}) \propto p(\tau_0) \prod_{t=1}^{n-1} w(B_t + \tau_t) w(Y_t + \tau_t), \qquad (7)$$

where $w(i) = f(s_i)$, for user-chosen transformation $f > 0$. Logically, we prefer to match theoretical product ions to higher intensity peaks in the observed spectrum. Our experiments with f focused on scaled and shifted exponential distributions, restricted to $[0, 1]$, since $s_i \in [0, 1]$. Based on these experiments, a class of f which performs well on a wide variety of data sets is

$$f_\lambda(S) = 1 - \lambda e^{-\lambda} + \lambda e^{-\lambda(1-S)}. \qquad (8)$$

The parameter, $\lambda > 0$, dictates the value placed on matching higher intensity peaks in the scoring function. The larger λ is, the higher the preference for matching b- and y-ions to high intensity peaks. There is no simple threshold for determining peak relevance based only on intensity; a PSM can have a high score even if most of the peaks identified are of low intensity. The unusual form of f_λ came out of an attempt to use exponential distributions to model peak intensity. Since $f_\lambda(S)$ is nearly log-linear on $[0, 1]$, one can instead use $f'_{\lambda_2}(S) = \exp(\lambda_2 S)$, where $\lambda_2 > 0$ is chosen to make $f'_{\lambda_2} \approx f_\lambda$. We get slightly better results using f_λ.

To optimize λ with respect to $\theta(\mathbf{s}, \mathbf{p})$, we use a grid search on a development set of 1000 yeast spectra; $\lambda = 0.5$ yielded the best results. λ acts as a tuning parameter for soft peak filtering. Both SEQUEST and MS-GFDB have to decide whether a peak is relevant before scoring, which can affect which PSM is selected.

3.4 Modeling fragmentation of charge +3 peptide ions

Thus far, we have assumed that the precursor ion has two protons, which are divided equally between product ions. However, electrospray ionization frequently produces peptide precursor ions with more than two protons (Trauger et al., 2002). That is, the charge of the spectrum is higher than +2. In this section, we extend Didea to model the fragmentation of charge +3 peptides.[5]

Here, we have that $c(p) = +3$, with $c(b_t)$ and $c(y_t)$ denoting the charges of the product ions. Since charge is conserved in fragmentation we have $\forall t,\ c(p) = c(b_t) + c(y_t)$. The charge of b_t is modeled as a multinomial $\xi_t \in \{0, +1, +2, +3\}$. It is unusual for a product ion to consume all the protons, and uncharged products are not detectable. In the absence of prior knowledge, we assume that the remaining choices are equally probable, $p(\xi_t = +1) = p(\xi_t = +2) = 0.5$. Figure 3(a) is the model for charge +3 peptide ions.

The definition of the scoring function, Equation 6, remains unchanged. The inference still uses $\log p(\tau_n = 0 \mid \mathbf{p}, \mathbf{s})$ as the score of a PSM. However, the values of α_d and β_d change, because the joint distribution over PSMs has changed:

$$p_{\text{ch3}}(\tau_0, \mathbf{p}, \mathbf{s}) \propto p(\tau_0) \prod_{t=1}^{n-1} \sum_{\xi_t} p(\xi_t) \Omega(t), \qquad (9)$$

$$\Omega(t) = \left[w(B_t(N_t, \xi_t) + \tau_t) w(Y_t(C_t, \xi_t) + \tau_t) \right],$$

$$B_t(N_t, \xi_t) = \text{round}\left((N_t + \xi_t)/\xi_t \right), \qquad (10)$$

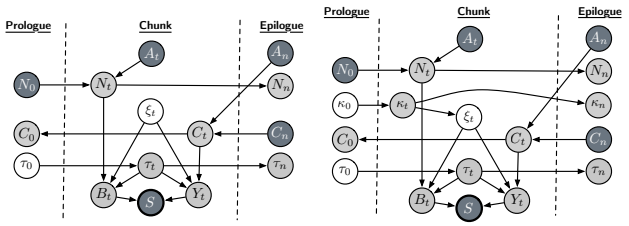
$$Y_t(C_t, \xi_t) = \text{round}\left((C_t + 18 + \xi_t)/\xi_t \right). \qquad (11)$$

Equations 10–11 are the mass to quantized m/z mappings for b- and y-ions. For each pair of ions (b_t, y_t), inference integrates over all divisions of the charge.

SEQUEST has a very different treatment of charge-variants of product ions. The theoretical spectrum in SEQUEST consists of the union of peaks where $c(b_t) = +1$ and $c(b_t) = +2$. As a thought experiment, consider what peaks need to be in the observed spectrum to maximize the SEQUEST score. The α_x term is maximized when all the theoretical peaks appear in the spectrum; the β_x term is minimized when no shifted version of the spectrum matches the theoretical peaks. To maximize the SEQUEST score, the observed spectrum would have to contain all peaks corresponding to all charge variants of the product ions, except those that would contribute significantly to the β_x term. There can be an exponentially large gap between the number of peaks in the SEQUEST theoretical spectrum, and the observed spectrum. In contrast, our approach averages the score over all 2^{n-1} assignments to $\xi = (\xi_1 \dots \xi_{n-1})$.[6] Each assignment to ξ corresponds to a particular combination of product ion peaks. Maximizing the score with respect to a peptide does not make the assumption that all charge-variants of the

[5] It is possible for electrospray ionization of tryptic peptides to produce peptide ions with a higher charge than +3, though doing so depends both on the mass spectrometer, and properties of the peptide (Kinter and Sherman, 2000, p. 69). Our data does not include such spectra, but it is straightforward to extend the model described here to charge $> +3$ spectra.

[6] Because of conditional independencies in Figure 3(a), $p(\xi)$ factors such that integrating over ξ is done in $O(n)$ time.



(a) Didea for database search (Section 3.4)

(b) Didea for multiply-charged spectra (Section 3.5)

Figure 3: Didea models which are used to handle multiply-charged spectra *(a)* ξ_n is the charge of the b-ion. *(b)* κ_0 is a Bernoulli random variable over spectrum charge $\{+2, +3\}$, and is copied in each frame, $\forall t, \kappa_t = \kappa_0$.

product ions exist in the spectrum. In our approach, if many possible peaks are missing from the spectrum, then $p(\xi)$ will get smaller as the disagreement between ξ and the spectrum increases. It is possible for $p(\xi)$ to be infinitesimally small for all but a few values of ξ. That is, a high score can be achieved even if only a few ξ have significant probability.

3.5 Multiply-charged spectra

Electrospray ionization routinely produces peptide precursor ions where charge $c(s) \in \{+1, +2, +3\}$. When $c(s) = +1$, the spectrum is referred to as *singly-charged*; when $c(s) = +2$, the spectrum is referred to as *doubly-charged*; when $c(s)$ has more than one possible value, we are unable to distinguish which charge with certainty, and the spectrum is referred to as *multiply-charged*.[7]

Existing database search algorithms analyze multiply-charged spectra by doubling the amount of computation (Eng et al., 1994). Each peptide-spectrum match is searched at $c(s) = +2$ and again at $c(s) = +3$, and the higher scoring match is used. If Ψ_c is the search algorithm specialized to charge c, replace Equation 2 with

$$p^* = \operatorname*{argmax}_{p \in C(m,P,\delta)} \max_{c \in \{+2,+3\}} \Psi_c(s, p). \qquad (12)$$

We could do the same search over charges with Didea. Evaluating $\theta(\cdot, \cdot)$ using Figure 2 corresponds to Ψ_{+2}; using Figure 3(a) corresponds to Ψ_{+3}. However, a concern with Equation 12 is that it assumes that the per-charge scoring functions are calibrated,

$$\mathbf{E}_{s \sim S, p \sim P} \left[\Psi_{+2}(s, p) \right] = \mathbf{E}_{s \sim S, p \sim P} \left[\Psi_{+3}(s, p) \right]. \qquad (13)$$

[7]The only multiply-charged spectra in our data have $c(s) \in \{+2, +3\}$. Extending the material of this section to other forms of multiply-charged spectra is straightforward.

A multiply-charged spectrum is an expression of uncertainty as to the charge state of the precursor. The mass window in the MS1 step is such that selecting the same peptide, at different charges, in the same MS2 scan, is almost impossible. It is not plausible that a single spectrum represents a mixture of the same peptide at different charges. So while taking the max of a set of charge-specific scoring function $\{\Psi_c\}_c$ allows one to make identification decisions, it provides no measure of uncertainty as to a spectrum's true charge—either charge +2 or charge +3 is selected, but no distribution over spectrum charge state can be produced.

Thus, rather than taking a maximum over scores for different charge states, we alter Didea to model uncertainty as to the value of $c(s)$. In Figure 3(b) we introduce a Bernoulli random variable $\kappa_0 \in \{+2, +3\}$, which is copied into each chunk frame. Moreover,

$$p(\xi_t \mid \kappa_t) = \begin{cases} p(\xi_t = +1) = 1.0 & \kappa_t = +2, \\ p(\xi_t = +1) = 0.5 & \kappa_t = +3, \end{cases} \qquad (14)$$

We assume no prior knowledge about the distribution of $c(s)$, $p(\kappa_0 = +2) = 0.5$. The rest of the parameters remain unchanged from Section 3.4. When $\kappa_0 = +2$, the model is identical to Figure 2. When $\kappa_0 = +3$, the model is identical to Figure 3(a). Since κ_0 is uncertain, when $\theta(\cdot, \cdot)$ is computed the score will be an average of both models. Note, however, that the resulting score is not the same as averaging the scores output by the +2 and +3 model. In each frame $t > 0$, the scoring inference is using an estimate of the spectrum charge based on the product ions considered—i.e., $p(\kappa_t)$ differs at each t. We conjecture that $p(\kappa_n \mid a_1, \ldots, a_n)$ can be used as an estimate of the uncertainty in the charge of the peptide-spectrum match chosen by Didea.

4 Experiments

Evaluating spectrum identification algorithms is complicated by the lack of test data. One cannot collect realistic spectra from known peptides. Section 4.1 describes how to estimate *absolute ranking curves*, which are the standard approach for comparing spectrum identification algorithms. We benchmark Didea against a panel of spectra from three different organisms (Section 4.2), presenting results in Section 4.3.

4.1 Evaluation Without Ground Truth

A correct peptide-spectrum match (PSM) (p_i, s_i) is one where s_i is the result of the fragmentation of peptide p_i, and an incorrect PSM is any match that is not correct. Clearly, database search scores vastly more incorrect PSMs than correct ones. Moreover, evaluation is greatly complicated by the lack of ground truth. Con-

ceptually, ground truth for a spectrum identification could be generated by feeding a purified peptide p_* into the tandem mass spectrometer. Unfortunately, there is no way of purifying p_* to the point where contaminants are undetectable. Moreover, even if one could run a pure peptide sample through the mass spectrometer, the resulting spectrum s_* would exhibit an unrealistically low level of noise; the test data would not reflect a real shotgun proteomics experiment. The standard solution in shotgun proteomics is to perform an evaluation without ground truth: measure the number of matches at a bound on the false discovery rate (Elias and Gygi, 2007; Käll et al., 2008).

For any peptide-spectrum match (p_i, s_i) with score v_i, there are two possibilities: either the peptide p_i generated spectrum s_i, or it did not. We refer to these events as hypotheses H_1 and H_0, respectively. If the scoring function is any good, then we expect the likelihood of v_i to be low under H_0, if (p_i, s_i) is the correct match. We can quantify what "low" means here using a hypothesis test,

$$H_0 : v_i \leq c, \qquad H_1 : v_i > c, \tag{15}$$

where the choice of $c \in \mathbb{R}$ determines the stringency of the test. Informally, the null hypothesis (H_0) is that the match is incorrect; the alternate (H_1) is that the match is correct.

Characterizing the accuracy at different values of c is greatly simplified by converting the scores to p-values, the probability of obtaining a score at least as large as c under the null hypothesis. Denote the cumulative distribution of scores under the null $G_0(c) \triangleq \mathbb{P}(T \leq c \,|\, H_0)$, where T is a random variable representing the PSM score. Given PSM scores $\{v_i\}_i$ the corresponding p-values are $p_i = 1 - G_0(v_i)$. Computing p-values for a wide range of scores v_i is tantamount to estimating the null distribution. Estimated p-values $\{\hat{p}_i\}_i$ are generated by replacing the exact (unknown) null distribution with an estimate $\hat{G}_0(c)$.

For spectrum identification, a widespread approach for estimating the null distribution is *target-decoy search* (Balgley et al., 2007; Elias and Gygi, 2007; Käll et al., 2008), in which a second search is performed against each spectrum using a set of decoy peptides $d \in D$, each of which could not have generated the spectrum. The original peptides $t \in P$ are referred to as targets. Under the constraint that $P \cap D = \emptyset$, decoys are used to generate a sample from the null distribution

$$\bar{v}_i = \max_{d \in D} \Psi(s_i, d), \quad \forall i = 1 \ldots r. \tag{16}$$

The samples $\{\bar{v}_i\}_i$ are then used in a Monte Carlo estimate of the p-values: $\hat{p}_i$ is just the fraction of sampled decoy scores that exceed v_i. In our experiments, decoy peptides are generated by randomly permuting the reference proteome, which induces a set of random peptides under a trypsin digest.

Since there are r hypothesis tests (spectra), we can measure the tradeoff between the number of PSMs that are accepted and the stringency of the threshold on the score using false discovery rates. Visually, the tradeoff is represented using an *absolute ranking plot* (Figure 4), where each point on the x-axis is a q value $\tilde{q} \in [0, 1]$, a measure of the false discovery rate (Storey, 2002), and the corresponding value on the y-axis is the number of PSMs accepted at that q-value. At $q = 1$, all r identifications are accepted. Because peptide-spectrum matches are often used as input to another estimation task (e.g., protein identification or quantification) our concern is with maximizing performance at small q-values, so we only plot $q \in [0, 0.1]$. One method dominates another if its absolute ranking curve is strictly above the absolute ranking curve for the other method.

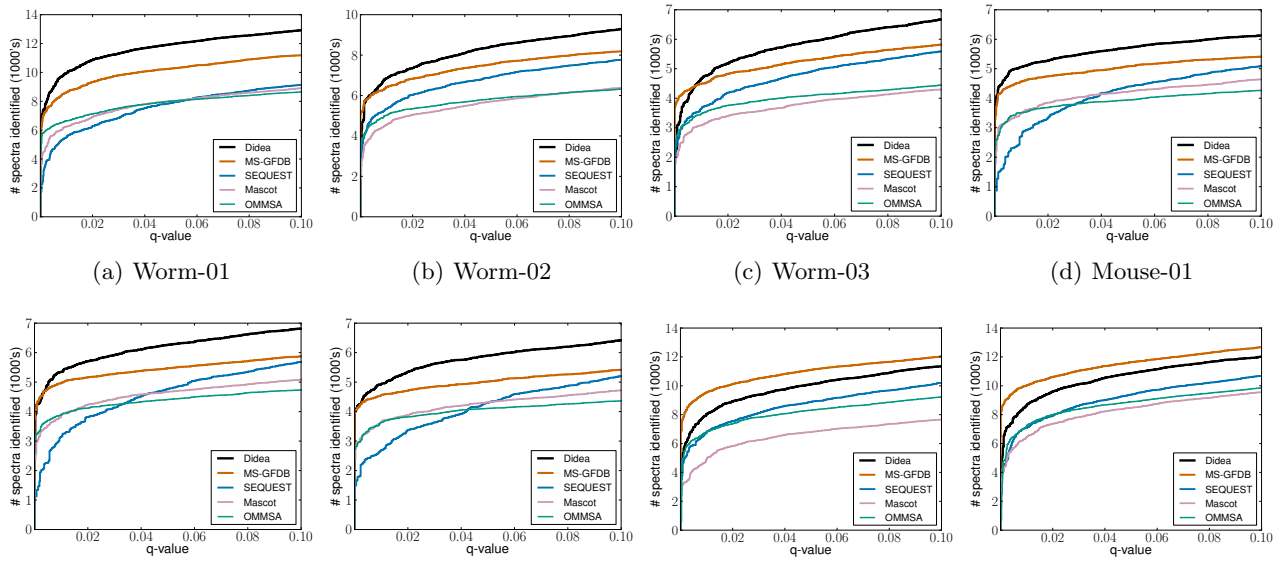
Absolute ranking measures what the end-user cares about: maximizing the number of spectra identified at their chosen false discovery rate tolerance.

4.2 Data sets

The spectra we study are generated from complex proteins samples drawn from three organisms: tryptic digests of a whole-cell lysate of *Saccharomyces cerevisiae* (yeast), whole-cell lysate of *Caenorhabditis elegans* (worm), and liver tissue from *Mus musculus* (mouse). The yeast and worm data are freely available at `http://noble.gs.washington.edu/proj/percolator`, with a description of the sample preparation procedure, liquid chromatography protocol, and mass spectrometer settings in Käll et al. (2007).

4.3 Results

Our primary claim is that Didea outperforms competing systems under absolute ranking, the metric that end-users actually care about. To support our empirical claim, we benchmark Didea against competitors. All algorithms have access to the same peptide database, created by a fully-tryptic *in silico* digest of reference yeast, worm, and mouse proteomes. Candidate peptides are chosen using a mass tolerance of $\delta = 3.0$ Daltons. A static modification is applied to account for cysteine carbidomethylation in all the algorithms. No variable modifications to amino acids are permitted. All spectra are either singly-charged or multiply-charged. All Didea-type scoring functions preprocess the spectra using Equation 8 and use exact inference to compute the PSM score. For Didea, we use the version of the model which integrates over uncertainty in the precursor charge (Section 3.5).



(a) Worm-01 (b) Worm-02 (c) Worm-03 (d) Mouse-01

(e) Mouse-02 (f) Mouse-03 (g) Yeast-01 (h) Yeast-02

Figure 4: Absolute ranking curves on eight different data sets. Didea significantly outperforms all competitors in (a-f). Didea significantly outperforms all competitors, save MS-GFDB (g-h). If $q = 0.05$, then the expected fraction of false discoveries (incorrect PSMs) in the matches accepted is 0.05. The goal is to maximize the number of spectra identified for $q \in [0, 0.1]$.

The absolute ranking plots for the benchmark are in Figure 4. At an estimated 1% false discovery rate, Didea outperforms MS-GFDB. While we often outperform MS-GFDB, there are cases where it beats Didea (Figures 4(g)-4(h)). Surprisingly, Didea is able to beat MS-GFDB even though it models less of the complexity of peptide fragmentation physics: we use no a-ions, nor neutral losses, nor do we account for errors in the m/z measurements in a fragmentation spectrum, like MS-GFDB does. Even more surprising is that Didea achieves high predictive power with only one trained parameter, λ. Even using a coarse grid search, the chosen value of λ works well across a variety of data sets. Training λ on a per-organism basis does not significantly increase the absolute ranking curve for Didea. Furthermore, we filter no peaks from the spectrum, even though real spectra are extremely noisy. In contrast, both SEQUEST and MS-GFDB require extensive preprocessing of the spectrum to work well.

All spectrum identification tools make an empirical claim, that their system identifies more spectra at a given q-value, or range of q-values. We compare favorably on such evaluations, against a wide range of competitors, using a relatively simple model and probabilistic inference.

5 Summary and Future Work

We have formulated a scoring function for spectrum identification based on a polynomial-time inference in a dynamic Bayesian network. Didea is significantly more accurate than both SEQUEST and Mascot, the primary tools used for this task in real-world use. Moreover, we outperform even recently developed competitors, like MS-GFDB, on many spectra. We anticipate that adding additional information, e.g., a-ions, neutral losses, relationships between ion yield and ion composition, will further improve the accuracy of Didea.

We believe that Didea is suggestive of the promise of applying techniques popular in speech recognition, such as dynamic Bayesian networks, to spectrum identification. For example, natural language processing often uses a compressed representation of a natural language corpora, which can be seen as analogous to compressed representations of a peptide database. Especially exciting would be a variant of Didea which replaces database search with direct Viterbi decoding of the peptide sequence.

Acknowledgements: This work was funded by NIH awards R01 GM096306 and P41 GM103533.

References

V. Bafna and N. Edwards. On de novo interpretation of tandem mass spectra for peptide identification. In *RECOMB*, pages 9–18, 2003.

B. M. Balgley, T. Laudeman, L. Yang, T. Song, and C. S. Lee. Comparative evaluation of tandem ms search algorithms using a target-decoy search strategy. *Mol Cell Proteomics*, 6(9):1599–1608, 2007.

N. Bandeira, J. Ng, D. Meluzzi, R. Linington, P. Dorrestein, and P. Pevzner. De novo sequencing of nonribosomal peptides. In *Res Comp Mol Bio*, volume 4955, pages 181–195. 2008.

C. Bartels. Fast algorithm for peptide sequencing by mass spectroscopy. *Biomed Env Mass Spec*, 19:363–368, 1990.

M. Bern and D. Goldberg. EigenMS: De novo analysis of peptide tandem mass spectra by spectral graph partitioning. In *Res Comp Mol Bio*, volume 3500, pages 995–995. 2005.

S. Bhatia, Y. Kil, B. Ueberheide, B. Chait, L. Tayo, L. Cruz, B. Lu, J. Yates, and M. Bern. Constrained de novo sequencing of peptides with application to conotoxins. In *Res Comp Mol Bio*, volume 6577, pages 16–30. 2011.

J. Cox, N. Neuhauser, A. Michalski, R. Scheltema, J. Olsen, and M. Mann. Andromeda—a peptide search engine integrated into the MaxQuant environment. *J Proteome Res*, 10(4):1794–1805, 2011.

R. Craig and R. C. Beavis. Tandem: matching proteins with tandem mass spectra. *Bioinformatics*, 20:1466–1467, 2004.

V. Dancik, T. Addona, K. Clauser, J. Vath, and P. Pevzner. De novo peptide sequencing via tandem mass spectrometry. *J Comp Bio*, 6(3-4):327–342, 1999a.

V. Dancik, T. A. Addona, K. R. Clauser, and J. E. Vath. De novo peptide sequencing via tandem mass spectrometry: a graph-theoretical approach. In *RECOMB*, pages 135–144, 1999b.

R. Datta and M. Bern. Spectrum fusion: Using multiple mass spectra for de novo peptide sequencing. In *Res Comp Mol Bio*, volume 4955, pages 140–153. 2008.

J. E. Elias and S. P. Gygi. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nature Methods*, 4 (3):207–214, 2007.

J. K. Eng, A. L. McCormack, and J. R. Yates, III. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *J Am Soc Mass Spec*, 5:976–989, 1994.

B. Fischer, V. Roth, J. Buhmann, J. Grossmann, S. Baginsky, W. Gruissem, F. Roos, and P. Widmayer. A hidden markov model for de novo peptide sequencing. In *NIPS*, 2004.

A. Frank and P. Pevzner. Pepnovo: de novo peptide sequencing via probabilistic network modeling. *Analytical Chemistry*, 77:964–973, 2005.

A. Frank, S. Tanner, and P. Pevzner. Peptide sequence tags for fast database search in mass-spectrometry. In *RECOMB*, pages 326–341, 2005.

L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant. Open mass spectrometry search algorithm. *J Proteome Res*, 3(5):958–964, 2004.

K. Jeong, S. Kim, N. Bandeira, and P. Pevzner. Gapped spectral dictionaries and their applications for database searches of tandem mass spectra. In *Res Comp Mol Bio*, volume 6044, pages 208–232. 2010.

L. Käll, J. Canterbury, J. Weston, W. S. Noble, and M. J. MacCoss. A semi-supervised machine learning technique for peptide identification from shotgun proteomics datasets. *Nature Methods*, 4:923–25, 2007.

L. Käll, J. D. Storey, M. J. MacCoss, and W. S. Noble. Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. *J Proteome Res*, 7(1):29–34, 2008.

A. Keller, A. I. Nesvizhskii, E. Kolker, and R. Aebersold. Empirical statistical model to estimate the accuracy of peptide identification made by MS/MS and database search. *Analytical Chemistry*, 74:5383–5392, 2002.

S. Kim, N. Mischerikow, N. Bandeira, J. D. Navarro, L. Wich, S. Mohammed, A. J. R. Heck, and P. A. Pevzner. The generating function of CID, ETD, and CID/ETD pairs of tandem mass spectra: Applications to database search. *Mol Cell Proteomics*, 9(12): 2840–2852, 2010.

M. Kinter and N. E. Sherman. *Protein sequencing and identification using tandem mass spectrometry.* Wiley-Interscience, 2000.

A. A. Klammer, S. R. Reynolds, M. Hoopmann, M. J. MacCoss, J. Bilmes, and W. S. Noble. Modeling peptide fragmentation with dynamic Bayesian networks yields improved tandem mass spectrum identification. *Bioinformatics*, 24(13):i348–i356, 2008.

Q. Li, M. J. MacCoss, and M. Stephens. A nested mixture model for protein identification using mass spectrometry. *Ann App Stats*, 4(2):962–987, 2010.

E. M. Marcotte. How do shotgun proteomics algorithms identify proteins? *Nature Biotech*, 25:755–757, 2007.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1998.

D. N. Perkins, D. J. C. Pappin, D. M. Creasy, and J. S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20:3551–3567, 1999.

T. Sakurai, T. Matsuo, H. Matsuda, and I. Katakuse. PAAS3: A compute program to determine probable sequence of peptides from mass spectrometric data. *Biomed Mass Spectrom*, 11(8):396–399, 1984.

H. Steen and M. Mann. The ABC's (and XYZ's) of peptide sequencing. *Nature Reviews Molecular Cellular Biology*, 5:699–711, 2004.

J. D. Storey. A direct approach to false discovery rates. *J Royal Stat Soc*, 64:479–498, 2002.

S. Tanner, H. Shu, A. Frank, L.-C. Wang, E. Zandi, M. Mumby, P. A. Pevzner, and V. Bafna. InsPecT: Identification of posttranslationally modified peptides from tandem mass spectra. *Analytical Chemistry*, 77:4626–4639, 2005.

S. A. Trauger, W. Webb, and G. Siuzdak. Peptide and protein analysis with mass spectrometry. *Spectroscopy*, 16(16–28), 2002.

Y. Wan, A. Yang, and T. Chen. PepHMM: A hidden Markov model based scoring function for mass spectrometry database search. *Analytical Chemistry*, 78 (2):432–437, 2006.

Detecting Change-Points in Time Series by Maximum Mean Discrepancy of Ordinal Pattern Distributions

Mathieu Sinn
IBM Research
Dublin, Ireland

Ali Ghodsi
Department of Statistics and
Actuarial Science, University of Waterloo
Waterloo, Ontario, Canada

Karsten Keller
Institute of Mathematics
University of Lübeck
Lübeck, Germany

Abstract

As a new method for detecting change-points in high-resolution time series, we apply Maximum Mean Discrepancy to the distributions of ordinal patterns in different parts of a time series. The main advantage of this approach is its computational simplicity and robustness with respect to (non-linear) monotonic transformations, which makes it particularly well-suited for the analysis of long biophysical time series where the exact calibration of measurement devices is unknown or varies with time. We establish consistency of the method and evaluate its performance in simulation studies. Furthermore, we demonstrate the application to the analysis of electroencephalography (EEG) and electrocardiography (ECG) recordings.

1 INTRODUCTION

Detecting changes in the temporal evolution of a system (biological, physical, mechanical, etc.) is of major importance, e.g. in medical diagnostics, industrial quality control, or the analysis of financial markets. In statistics, the problem of inferring the time point of a change from a sequence of observations is known as change-point detection. There is a vast literature on parametric and non-parametric methods for testing the presence of change-points and for estimating their locations. For an overview, we refer to the monographs of Chen and Gupta (2000) and Brodsky and Darkhovsky (1993, 2000). Recently, Harchaoui et al. (2009) have introduced an estimator for change-points based on kernels.

In this paper, we propose to detect change-points by comparing the distributions of ordinal patterns in different parts of a time series. In order to capture differences among the distributions automatically, we apply the Maximum Mean Discrepancy (MMD) criterion recently introduced by Gretton et al. (2007a, 2007b). The proposed method is computationally fast and robust with respect to (non-linear) monotonic transformations of the observations, which makes it particularly attractive for the exploration of high-resolution biophysical time series where the exact calibration of measurement devices is unknown or varies with time. In real-life applications, such changes in the calibration occur, e.g., for electroencephalography (EEG) recordings where small displacements of electrodes result in changes of the conductivity.

Detecting changes in the dynamics of a time series by looking at ordinal pattern distributions has first been proposed by Bandt and Pompe (2002). More specifically, they consider permutation entropy, which is the Shannon entropy of ordinal pattern distributions, as a measure for detecting changes in the complexity of time series. Permutation entropy has been applied to the analysis of epileptic activity in EEG data (Li et al., 2007; Bruzzo et al., 2008) and to measuring anaesthetic drug effects (Li et al., 2008; Olofsen et al., 2008). In methodological studies, Keller et al. (2007a, 2007b) define further statistics of ordinal pattern distributions besides permutation entropy and investigate properties of the distributions themselves. Bandt and Shiha (2007) derive formulas for ordinal pattern probabilities in stochastic processes; Sinn and Keller (2011) study statistical properties of estimators of these probabilities.

This paper is organized as follows: In Section 2, we formalize the change-point problem and review the definition of ordinal pattern distributions. Section 3 shows how MMD can be used to automatically detect change-points and discusses related work. In Section 4, we evaluate the performance of our methods in simulation studies and demonstrate the application to real-life time series. Section 5 concludes the paper.

2 OUTLINE OF THE METHOD

In this paper, we consider the problem of inferring the time point of a change in the distribution of an observed time series. The basic idea is to look at the order structure in different parts of the time series; if we find a time point with a clear difference between the order structure before and afterwards, we conclude that it is a change-point.

Formally, we describe the problem as follows: Let $\mathbb{N}$ denote the set of natural numbers and $\mathbb{Z}$ the set of integers. Consider a real-valued stochastic process $\mathbf{X} = (X_t)_{t\in\mathbb{Z}}$ on some probability space $(\Omega, \mathcal{A}, \mathbb{P})$, where Ω denotes the sample space, $\mathcal{A}$ the set of measurable events and $\mathbb{P}$ the probability measure. Let $\mathbf{Y} = (Y_t)_{t\in\mathbb{Z}}$ be the process of increments given by $Y_t := X_t - X_{t-1}$ for $t \in \mathbb{Z}$. Throughout this paper, we always assume the following holds:

(A1) The process $\mathbf{Y}$ is non-degenerate, i.e., all finite-dimensional distributions of $\mathbf{Y}$ are absolutely continuous with respect to the Lebesgue measure.

(A2) The processes $(Y_0, Y_{-1}, \ldots)$ and $(Y_1, Y_2, \ldots)$ are strictly stationary and ergodic.

Note that as a consequence of (A1), the values of $\mathbf{X}$ are pairwise distinct $\mathbb{P}$-almost surely, that is,

$$\mathbb{P}(X_{t_1} \neq X_{t_2}) \quad = \quad 1 \tag{1}$$

for all $t_1, t_2 \in \mathbb{Z}$ with $t_1 \neq t_2$.

Now let $m, n \in \mathbb{N}$. Suppose that we observe a time series of length $m+n$ with a change in the underlying distribution after the first m time points, and m, n are unknown (only the length $m+n$ is observable). Then finding the location of the change-point is equivalent to estimating the pair (m, n). We model this situation by assuming the observed time series is a realization of $\mathbf{X}$ at times $t = -m, -m+1, \ldots, n-2, n-1$. If the distributions of $(Y_0, Y_{-1}, \ldots)$ and $(Y_1, Y_2, \ldots)$ are different, then $t = 0$ is a change-point in $\mathbf{X}$ and we obtain m observations before and n observations afterwards.

A generalization of the situation with only one change in the distribution is the multiple change-point problem where the time series potentially has more than one change-point. A common approach to solve this problem is to iteratively apply a method for detecting single change-points, i.e., after the detection of the first change-point, the method is applied to the time segments before and afterwards to search for further change-points. In Section 4.1, we will consider this approach in more detail.

2.1 ORDINAL PATTERN DISTRIBUTIONS

We use the concept of ordinal pattern distributions to describe the order structure of a time series. An ordinal pattern represents the order relations among successive values of a time series; if the values are pairwise different, it is natural to identify ordinal patterns with permutations. Counting the number of occurrences of the permutations in a time series (or parts of it) yields the empirical distribution of ordinal patterns.

Here we give a formal definition of ordinal patterns. Let $d \in \mathbb{N}$ be fixed. By $\mathbf{\Pi}$ we denote the set of permutations $\{0, 1, \ldots, d\}$, which we represent by $(d+1)$-tuples containing each of the numbers $0, 1, \ldots, d$ exactly one time. Let the permutations be denoted by $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_{(d+1)!}$ so that $\mathbf{\Pi} = \{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_{(d+1)!}\}$. Now consider the partition of $\mathbb{R}^{d+1}$ obtained by identifying vectors whose components are correspondingly ordered: For $\boldsymbol{\pi} = (r_0, r_1, \ldots, r_d) \in \mathbf{\Pi}$, let $B(\boldsymbol{\pi})$ be the subset of $\mathbb{R}^{d+1}$ containing every vector $(x_0, x_1, \ldots, x_d)$ satisfying

(*i*) $x_{r_0} \geq x_{r_1} \geq \ldots \geq x_{r_d}$,

(*ii*) if $x_{r_i} = x_{r_{i+1}}$, then $r_i > r_{i+1}$.

Obviously, the union of all subsets $B(\boldsymbol{\pi}_1)$, $B(\boldsymbol{\pi}_2)$, ..., $B(\boldsymbol{\pi}_{(d+1)!})$ is equal to $\mathbb{R}^{d+1}$, and by condition (*ii*) the pair of subsets $B(\boldsymbol{\pi}_i)$, $B(\boldsymbol{\pi}_j)$ is disjoint if $i \neq j$. By saying that the *ordinal pattern* of *order* d at *time* $t \in \mathbb{Z}$ is given by $\boldsymbol{\pi} = (r_0, r_1, \ldots, r_d)$, we designate the event

$$(X_t,\, X_{t-1},\, X_{t-2}, \ldots,\, X_{t-d}) \quad \in \quad B(\boldsymbol{\pi}).$$

According to (1), this event is equivalent to

$$X_{t-r_0} \,>\, X_{t-r_1} \,>\, \ldots \,>\, X_{t-r_d}$$

$\mathbb{P}$-almost surely. Note that a more general definition of ordinal patterns includes an additional parameter $\tau \in \mathbb{N}$ (called the *delay*) which allows us to consider the events $(X_t,\, X_{t-\tau}, \ldots,\, X_{t-d\tau}) \in B(\boldsymbol{\pi})$ (see Keller et al. (2007a)).

Figure 1 shows all the possible outcomes for ordinal patterns of order $d = 3$. For example, if $\omega \in \Omega$ is such that $X_t(\omega) > X_{t-1}(\omega) > X_{t-2}(\omega) > X_{t-3}(\omega)$ (thus, the time series roughly looks like the upper left plot in Figure 1), then the ordinal pattern at time t is given by (0,1,2,3). If $X_t(\omega) > X_{t-1}(\omega) > X_{t-3}(\omega) > X_{t-2}(\omega)$, then the ordinal pattern is given by $(0, 1, 3, 2)$, and so on. The dark gray patterns $(0, 1, 2, 3)$ and $(3, 2, 1, 0)$ in Figure 1 occur if the underlying time series is monotonically increasing and decreasing, respectively. The light gray patterns occur if the past two increments

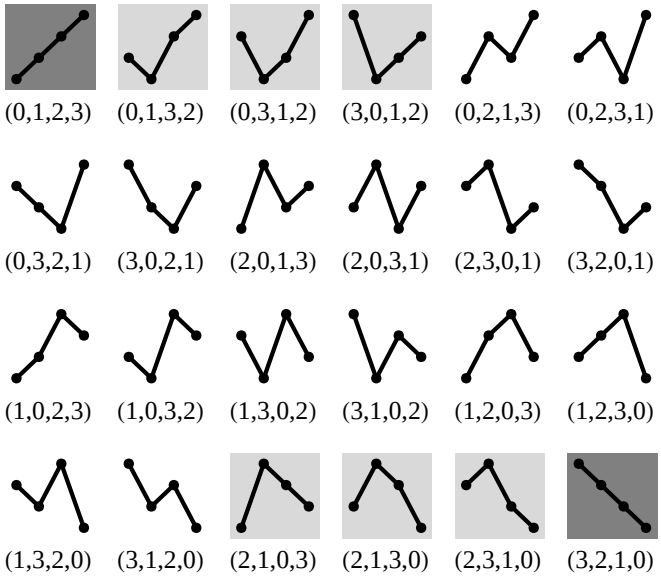


Figure 1: Possible outcomes for ordinal patterns of order $d = 3$.

have the same sign, and the white patterns occur after changes between "upwards" and "downwards".

Next, we consider the distribution of ordinal patterns in the process $\mathbf{X}$. For $t \in \mathbb{Z}$ and $k = 1, 2, \ldots, (d+1)!$ define

$$p_k(t) \;:=\; \mathbb{P}(\,(X_t,\, X_{t-1},\, \ldots,\, X_{t-d}) \,\in\, B(\boldsymbol{\pi}_k)\,),$$

that is, $p_k(t)$ denotes the probability that the ordinal pattern at time t is given by $\boldsymbol{\pi}_k$. By the *distribution* of ordinal patterns at time t we mean the stochastic vector

$$\mathbf{p}(t) \;:=\; (p_1(t),\, p_2(t),\, \ldots,\, p_{(d+1)!}(t))\,.$$

Note that the ordinal pattern at time t actually only depends on the increments $Y_t, Y_{t-1}, \ldots, Y_{t-d+1}$ (see Sinn and Keller (2011)). Hence, a sufficient condition for the ordinal pattern distributions $\mathbf{p}(t_1)$ and $\mathbf{p}(t_2)$ to be identical is that the random vectors $(Y_{t_1}, Y_{t_1-1}, \ldots, Y_{t_1-d+1})$ and $(Y_{t_2}, Y_{t_2-1}, \ldots, Y_{t_2-d+1})$ have the same distribution. According to assumption (A2), we obtain that

$$\ldots \,=\, \mathbf{p}(-1) \,=\, \mathbf{p}(0) \quad \text{and} \quad \mathbf{p}(d) \,=\, \mathbf{p}(d+1) \,=\, \ldots,$$

i.e., the ordinal pattern distribution at times $t \leq 0$ and $t \geq d$, respectively, are identical. In general, the distributions $\mathbf{p}(t)$ with $t = 1, 2, \ldots, d-1$ are equal neither to $\mathbf{p}(0)$ nor to $\mathbf{p}(d)$ because they depend on increments both before and after $t = 0$.

In the following, we write $\mathbf{p}_-$ and $\mathbf{p}_+$ to denote the generic distributions $\mathbf{p}(0)$ and $\mathbf{p}(d)$, respectively. Obviously, a necessary condition for $\mathbf{p}_- \neq \mathbf{p}_+$ is that $\mathbf{X}$ has a change-point at $t = 0$. The key idea of our method for change-point detection is to measure the difference between the empirical ordinal pattern distributions in different parts of a time series.

2.2 EMPIRICAL ORDINAL PATTERN DISTRIBUTIONS

In the following, let $\mathbf{1}\{\cdot\}$ denote the function which evaluates to 1 if the statement in the brackets is true, and to 0 otherwise. For $w \in \mathbb{N}$ and $t \in \mathbb{Z}$, define

$$\hat{p}_k(w,t) \;:=\; \frac{1}{w} \sum_{s=wt+1}^{wt+w} \mathbf{1}\big\{\,(X_s,\, X_{s-1},\, \ldots,\, X_{s-d}) \,\in\, B(\boldsymbol{\pi}_k)\,\big\},$$

that is, $\hat{p}_k(w,t)$ is the relative frequency of time points in the time window $\{wt+1, wt+2, \ldots, wt+w\}$ at which the ordinal pattern is $\boldsymbol{\pi}_k$. The parameter w determines the size of the time window; as we will see in Theorem 1 below, the larger w, the closer we can expect $\hat{p}_k(w,t)$ be to $p_k(t)$.

Now, taking into account all the estimates $\hat{p}_k(w,t)$ for $k = 1, 2, \ldots, (d+1)!$ gives us the empirical ordinal pattern distribution

$$\hat{\mathbf{p}}(w,t) \;:=\; \big(\hat{p}_1(w,t),\, \hat{p}_2(w,t),\, \ldots,\, \hat{p}_{(d+1)!}(w,t)\big)\,.$$

As the following theorem shows, $\hat{\mathbf{p}}(w,t)$ converges to $\mathbf{p}_-$ if $t < 0$, and to $\mathbf{p}_+$ if $t \geq 0$ ($\mathbb{P}$-almost surely). This result also holds if the "calibration" of $\mathbf{X}$ changes at a number of points that grows sublinearly in time.

Theorem 1 *Suppose that conditions* (A1)*and* (A2) *hold. Then*

$$\lim_{w\to\infty} \hat{\mathbf{p}}(w,t) \quad = \quad \begin{cases} \mathbf{p}_- & \text{if } t<0, \\ \mathbf{p}_+ & \text{if } t\geq 0, \end{cases}$$

$\mathbb{P}$*-almost surely. This result also holds if we observe the "distorted" process* $(h(X_t,t))_{t\in\mathbb{Z}}$ *instead of* $(X_t)_{t\in\mathbb{Z}}$ *with*

$$h(x,t) \quad := \quad \sum_{j\in\mathbb{Z}} \mathbf{1}\{t\in(t_{j-1},t_j]\}\, h_j(x),$$

where the mappings $h_j:\mathbb{R}\to\mathbb{R}$ *are strictly increasing for all* $j\in\mathbb{Z}$*, and* $(t_j)_{j\in\mathbb{Z}}$ *is an increasing sequence in* $\mathbb{Z}$ *which satisfies*

$$\lim_{w\to\infty} \frac{\sharp\{j\in\mathbb{Z}:\ t_j\in[-w,w]\}}{w} \quad = \quad 0.$$

Proof. First, we consider the case $t<0$. Since $(Y_0,Y_{-1},\ldots)$ is ergodic and

$$\mathbb{E}\big[\,\mathbf{1}\{(X_s,\,X_{s-1},\ldots,\,X_{s-d})\,\in\,B(\boldsymbol{\pi}_k)\}\,\big] \quad = \quad p_k(0)$$

for all $s\leq w(t+1)$ with $w\in\mathbb{N}$, the Ergodic Theorem shows that $\hat{p}_k(w,t)$ converges to $p_k(0)$ $\mathbb{P}$-almost surely (Cornfeld et al., 1982). If $t>0$, then $wt+1\geq d$ for sufficiently large w, and hence

$$\mathbb{E}\big[\,\mathbf{1}\{(X_s,\,X_{s-1},\ldots,\,X_{s-d})\,\in\,B(\boldsymbol{\pi}_k)\}\,\big] \quad = \quad p_k(d)$$

for all $s\geq wt+1$. Since $(Y_1,Y_2,\ldots)$ is ergodic, the Ergodic Theorem shows that $\hat{p}_k(w,t)$ converges to $p_k(d)$ $\mathbb{P}$-almost surely. In the case $t=0$, we have

$$\begin{aligned}
&\hat{p}_k(w,t) \\
&\quad = \quad \frac{1}{w}\sum_{s=1}^{d-1} \mathbf{1}\{(X_s,\,X_{s-1},\,\ldots,\,X_{s-d})\,\in\,B(\boldsymbol{\pi}_k)\} \\
&\quad + \quad \frac{1}{w}\sum_{s=d}^{w} \mathbf{1}\,\{(X_s,\,X_{s-1},\,\ldots,\,X_{s-d})\,\in\,B(\boldsymbol{\pi}_k)\}.
\end{aligned}$$

Since the first term on the right hand side tends to 0 and the second one converges to $p_k(d)$ $\mathbb{P}$-almost surely, the result follows.

Now suppose that we observe $(h(X_t,t))_{t\in\mathbb{Z}}$ instead of $\mathbf{X}$. Note that if $t,j\in\mathbb{Z}$ satisfy $t-d\geq t_{j-1}$ and $t\leq t_j$, then $(h(X_t),h(X_{t-1}),\ldots,h(X_{t-d}))$ lies in $B(\boldsymbol{\pi}_k)$ if and only if $(X_t,\,X_{t-1},\,\ldots,\,X_{t-d})$ lies in $B(\boldsymbol{\pi}_k)$. Therefore, for any $t\in\mathbb{Z}$, the number of time points $s\in\{wt+1,\,wt+2,\,\ldots,\,wt+w\}$ at which the indicator functions of these two events are unequal is bounded by $(d+1)$-times the number of $j\in\mathbb{Z}$ for which $t_j\in[wt+1,wt+w]$. Since the latter number divided by w tends to 0 as $w\to\infty$, we obtain the result. □

2.3 DETECTING CHANGE-POINTS

Based on the definition of ordinal pattern distributions and the consistency result of Theorem 1, we now present our method for change-point detection. Suppose that $\mathbf{X}$ has a change-point at time $t=0$ which results in a difference between the ordinal pattern distributions $\mathbf{p}_-$ and $\mathbf{p}_+$. Furthermore, suppose that we observe $\mathbf{X}$ at times $t=-wm+1,-wm+2,\ldots,wn-1,wn$ where $w\in\mathbb{N}$ is "large" and $m,n\in\mathbb{N}$ are unknown (only the sum $m+n$ is observed). In this setting, the problem of estimating the location of the change-point is equivalent to estimating the unknown values m and n. Our proposed method proceeds as follows:

1. We partition the observations of $\mathbf{X}$ into $(m+n)$ non-overlapping blocks of size w each.

2. For each block, we compute the empirical ordinal pattern distribution, which gives us the $(m+n)$-dimensional vector

$$\begin{aligned}
&\mathbf{z}(w) := \qquad\qquad (2)\\
&\quad (\hat{\mathbf{p}}(w,-m),\,\hat{\mathbf{p}}(w,-m+1),\,\ldots,\,\hat{\mathbf{p}}(w,n-1)).
\end{aligned}$$

According to Theorem 1, we have

$$\lim_{w\to\infty} \mathbf{z}(w) \quad = \quad (\underbrace{\mathbf{p}_-,\,\ldots,\,\mathbf{p}_-}_{m\text{ times}},\underbrace{\mathbf{p}_+,\,\ldots,\,\mathbf{p}_+}_{n\text{ times}}) \quad (3)$$

$\mathbb{P}$-almost surely. Thus, if the $m+n$ ordinal pattern distributions can be divided into two clearly distinct groups, the sizes of these groups can be taken as estimates for m and n. In the following section, we apply Maximum Mean Discrepancy to determine these groups.

Before discussing related work, let us make some remarks:

Remark 1. Our method is robust with respect to changes in the "calibration" of the process $\mathbf{X}$. In particular, if h_- and h_+ are both strictly increasing functions, then the ordinal pattern distributions $\mathbf{p}_-$ and $\mathbf{p}_+$ in the transformed process $(\ldots,\,h_-(X_{-1}),\,h_-(X_0),\,h_+(X_1),\,h_+(X_2),\,\ldots)$ are identical. As mentioned in the introduction, this robustness is often desirable for the analysis of biophysical time series the exact calibration of which is unknown or varies with time.

Remark 2. If d is large enough, then our method does detect changes of the complexity of $\mathbf{X}$ measured by the permutation entropy (Bandt and Pompe, 2002; Keller and Sinn, 2009). Furthermore, it is capable to detect changes in the autocorrelation structure of Gaussian processes (Bandt and Shiha, 2007).

Remark 3. There is some trade-off between the choice of the order of the ordinal patterns d and the time window size w. In general, a larger order d permits to detect changes of the higher-order autocorrelations of $\mathbf{Y}$ or, equivalently, in the lower frequency domain. On the other hand, the time window size w needs to be much larger than $(d+1)!$ in order to obtain reliable estimates of all the ordinal pattern probabilities, however, the larger w, the less accurate the localization of the change-points. For our experiments in Section 4, we have chosen $d = 3$ and $w = 500$, that is, we consider 24 different ordinal patterns and are capable to localize change-points with a precision of ± 250 time points which, for the EEG and ECG recordings, corresponds to approximately ± 1 second. See also Section 3.3 for a further discussion on this issue.

3 MAXIMUM MEAN DISCREPANCY

In order to group the components of the vector $\mathbf{z}(w)$ defined in (2), we use a recently proposed criterion called Maximum Mean Discrepancy (MMD). Let $\mathcal{Z}$ be an arbitrary set and $\mathbf{z} = (z_1, z_2, \ldots, z_{m+n}) \in \mathcal{Z}^{m+n}$ with $m, n \in \mathbb{N}$. MMD is a new criterion for testing whether $z_1, z_2, \ldots, z_m$ and $z_{m+1}, z_{m+2}, \ldots, z_{m+n}$ come from different distributions (Gretton et al., 2007a, 2007b). The basic approach is to measure the similarity of points $z, z' \in \mathcal{Z}$ using a kernel $k(z, z')$. In our case, $\mathcal{Z}$ is the set of $(d+1)!$-dimensional stochastic vectors, and for $k(z, z')$ we choose the Radial Basis Function (RBF) kernel

$$k(z, z') \;=\; \exp\left(-\frac{\|z - z'\|^2}{2\sigma^2}\right)$$

where $\|\cdot\|$ denotes the Euclidean norm and $\sigma^2 > 0$ is a smoothing parameter. For $m', n' \in \mathbb{N}$ with $m' + n' = m + n$, the *Maximum Mean Discrepancy* is given by

$$\mathrm{MMD}(\mathbf{z}, m', n') \;:=\; \left(\frac{\mathrm{K}_1(\mathbf{z}, m', n')}{m'm'} - \frac{2\,\mathrm{K}_2(\mathbf{z}, m', n')}{m'n'} + \frac{\mathrm{K}_3(\mathbf{z}, m', n')}{n'n'}\right)^{\frac{1}{2}}$$

where

$$\begin{aligned}
\mathrm{K}_1(\mathbf{z}, m', n') &:= \sum_{i=1}^{m'}\sum_{j=1}^{m'} k(z_i, z_j)\,,\\
\mathrm{K}_2(\mathbf{z}, m', n') &:= \sum_{i=1}^{m'}\sum_{j=1}^{n'} k(z_i, z_{m'+j})\,,\\
\mathrm{K}_3(\mathbf{z}, m', n') &:= \sum_{i=1}^{n'}\sum_{j=1}^{n'} k(z_{m'+i}, z_{m'+j})\,.
\end{aligned}$$

The terms $\frac{1}{m'm'}\mathrm{K}_1(\mathbf{z}, m', n')$ and $\frac{1}{n'n'}\mathrm{K}_3(\mathbf{z}, m', n')$ can be regarded as the average similarity of points within the groups $z_1,\ z_2,\ \ldots,\ z_{m'}$ and $z_{m'+1},\ z_{m'+2},\ \ldots,\ z_{m'+n'}$, respectively, while $\frac{1}{m'n'}\mathrm{K}_2(\mathbf{z}, m', n')$ measures the average similarity between the two groups. The maximal MMD value is obtained for that partition of $\mathbf{z}$ which yields maximum similarity within the groups and maximum dissimilarity between them. The next theorem shows how the MMD criterion can be used to locate change-points in $\mathbf{X}$.

Theorem 2 *Suppose that conditions* (A1) *and* (A2) *hold and let* $\mathbf{z}(w)$ *be as defined in* (2). *Then for all* $m', n' \in \mathbb{N}$ *with* $m' + n' = m + n$, *we have*

$$\lim_{w\to\infty} \mathrm{MMD}(\mathbf{z}(w), m', n') \;=\; \min\left\{\frac{n}{n'}\,,\,\frac{m}{m'}\right\}\sqrt{2\left(1 - k(\mathbf{p}_-, \mathbf{p}_+)\right)}$$

$\mathbb{P}$-*almost surely. In particular, if* $\mathbf{p}_- \neq \mathbf{p}_+$, *then*

$$\lim_{w\to\infty}\left(\arg\max_{\substack{(m',n')\in\mathbb{N}^2,\\ m'+n'=m+n}} \mathrm{MMD}(\mathbf{z}(w), m', n')\right) \;=\; (m, n)$$

$\mathbb{P}$-*almost surely.*

Proof. Suppose that $m' \leq m$. By equation (3) and the fact that $k(z, z) = 1$ for all $z \in \mathcal{X}$, the limits of $\mathrm{K}_1(\mathbf{z}(w), m', n')$, $\mathrm{K}_2(\mathbf{z}(w), m', n')$ and $\mathrm{K}_3(\mathbf{z}(w), m', n')$ are given by $m'm'$, $m'(m - m') + m'n\,k(\mathbf{p}_-, \mathbf{p}_+)$ and $n'(n'-2n)+2n(n'-n)\,k(\mathbf{p}_-, \mathbf{p}_+)$, respectively. Putting all terms together, we obtain

$$\lim_{w\to\infty} \mathrm{MMD}(\mathbf{z}(w), m', n') \;=\; \frac{n}{n'}\sqrt{2\left(1 - k(\mathbf{p}_-, \mathbf{p}_+)\right)}.$$

In the case $m' > m$, which is equivalent to $n' \leq n$, the result follows by symmetry. Now the second statement is obvious. □

3.1 BIAS CORRECTION

Theorem 2 shows that if the ordinal pattern distributions $\mathbf{p}_-$ and $\mathbf{p}_+$ are different, then the argument (m', n') which maximizes $\mathrm{MMD}(\mathbf{z}(w), m', n')$ is a strongly consistent estimator of (m, n). For finite w, however, practical experiments suggest that this estimator is biased towards $(1, m+n-1)$ and $(m+n-1, 1)$, particularly when the difference between $\mathbf{p}_-$ and $\mathbf{p}_+$ is small (see Section 4.1). Here we give a heuristic explanation: suppose that the similarity between any two components of $\mathbf{z} = (z_1, z_2, \ldots, z_{m+n})$, as measured by the RBF kernel, is approximately equal to some constant $\delta > 0$. For the random vector $\mathbf{z}(w)$, this is likely to be the case if the difference between $\mathbf{p}_-$ and $\mathbf{p}_+$ is so small that all estimates are scattered in the same region. A simple calculation shows that

$$\mathrm{MMD}(\mathbf{z}, m', n') \;\approx\; \left(1 - \delta\right)\frac{m+n}{m'n'} \qquad (4)$$

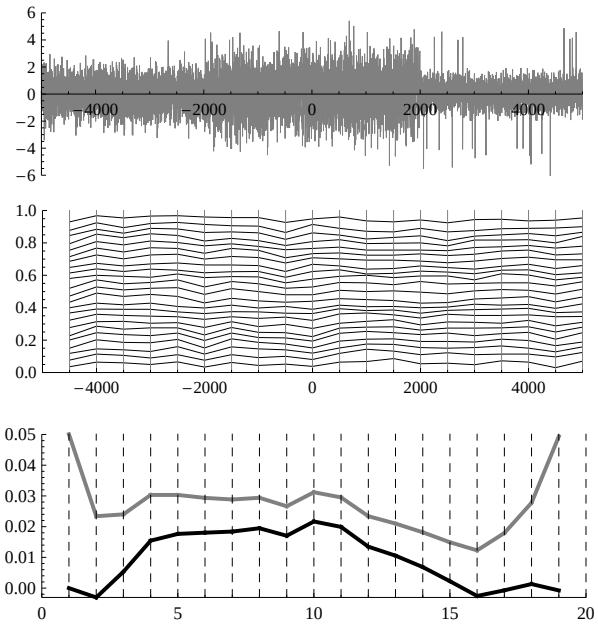


Figure 2: Top: Realization of an AR(1) process with a change-point at $t = 0$. Middle: Ordinal pattern distributions in time windows of size $w = 500$. Bottom: Resulting values for MMD (gray) and CMMD (black).

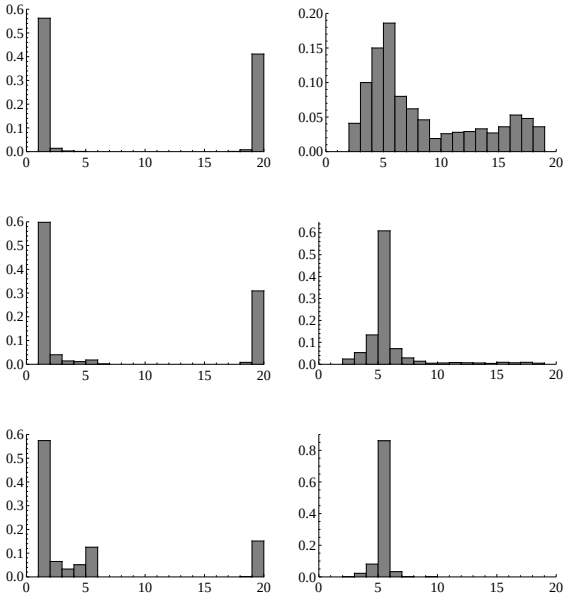


Figure 3: Distribution of the change-point estimators based on MMD (left) and CMMD (right).

for $m', n' \in \mathbb{N}$ with $m' + n' = m + n$, and hence the argument maximizing $\mathrm{MMD}(\mathbf{z}, m', n')$ is likely to be $(m', n') = (1, m + n - 1)$ or $(m', n') = (m + n - 1, 1)$. Next we propose a correction for the bias: define the *Corrected Maximum Mean Discrepancy*

$$\begin{aligned}\mathrm{CMMD}(\mathbf{z}, m', n') &:= \mathrm{MMD}(\mathbf{z}, m', n') \\ &\quad - \frac{m+n-1}{m'n'} \max_{\substack{(\tilde{m},\tilde{n})\in\mathbb{N}^2,\\ \tilde{m}+\tilde{n}=m+n}} \mathrm{MMD}(\mathbf{z}, \tilde{m}, \tilde{n}).\end{aligned}$$

The correction term on the right hand side can be regarded as an estimate of the bias of the MMD statistic (compare to (4)). Experimental results show the superiority of using CMMD instead of MMD (see Section 4.1). Note that the estimator of (m, n) based on CMMD is no longer consistent; in order to preserve consistency, the correction term would have to be scaled by a factor that tends to 0 as $w \to \infty$.

3.2 EFFICIENT COMPUTATION

Let us discuss the computational complexity of our method. The computation of the ordinal pattern distributions is linear in the length of the input sequence and can be realized by very efficient algorithms (Keller et al., 2007b). Computing the estimate of (m, n) by directly evaluating $\mathrm{MMD}(\mathbf{z}(w), m', n')$ for all $m', n' \in \mathbb{N}$ with $m' + n' = m + n$ requires $O((m+n)^3)$ arithmetic operations (Gretton et al., 2007a). For a more efficient computation, note that

$$\begin{aligned}\mathrm{K}_1(\mathbf{z}, m'+1, n'-1) &= \mathrm{K}_1(\mathbf{z}, m', n') \\ &\quad + 2\sum_{i=1}^{m'} k(z_i, z_{m'+1}) + k(z_{m'+1}, z_{m'+1}), \\ \mathrm{K}_2(\mathbf{z}, m'+1, n'-1) &= \mathrm{K}_2(\mathbf{z}, m', n') \\ &\quad - \sum_{i=1}^{m'} k(z_i, z_{m'+1}) + \sum_{i=1}^{n'-1} k(z_{m'+1}, z_{m'+1+i}), \\ \mathrm{K}_3(\mathbf{z}, m'+1, n'-1) &= \mathrm{K}_3(\mathbf{z}, m', n') \\ &\quad - 2\sum_{i=1}^{n'} k(z_{m'+1}, z_{m'+i}) + k(z_{m'+1}, z_{m'+1}).\end{aligned}$$

Using these formulas, $\mathrm{MMD}(\mathbf{z}(w), m', n')$ can be evaluated iteratively for all $(m', n') = (1, m + n - 1), \ldots, (m + n - 1, 1)$. Thus, computing the argument which maximizes $\mathrm{MMD}(\mathbf{z}(w), m', n')$ requires $O((m+n)^2)$ arithmetic operations.

3.3 RELATED WORK

There is an extensive literature on change-point detection, including both parameteric and non-parametric approaches (Chen and Gupta, 2000; Brodsky and Darkhovsky, 1993, 2000). Recently, a kernel-based method been proposed by Harchaoui et al. (2009). The main difference to our approach is that all these methods aim to find the *exact* location of change-points and give some guarantees on the significance level.

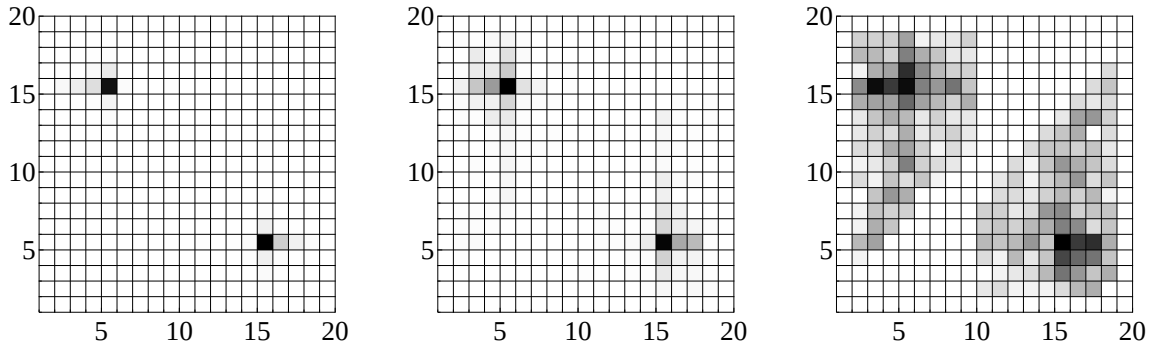


Figure 4: Detection of multiple change-points. The black coloring corresponds to the maximum frequencies 34.0% (left), 15.1% (middle) and 2.2% (right), the white coloring corresponds to 0%.

This precision comes at the price of a high computational burden; for example, Harchaoui's method has a running time which is cubic in the input size. Our method is designed to find the *approximate* location of change-points, where the precision is controlled by the time window size w. Because of its computational efficiency, our method is particularly attractive for the fast exploration of high-resolution time series such as EEG recordings, which often consist of hundreds of thousands of data points. Note that our approach can be combined with existing methods in order to first obtain the approximate locations of a set of candidate change-points before determining their exact locations and testing for significance.

4 EXPERIMENTS

4.1 SIMULATED DATA

Change of the autoregressive coefficient. Let us first illustrate our method for simulated data. The top plot of Figure 2 shows a realization of a first order autoregressive (AR(1)) process with standard normal innovations where, at time $t = 0$, the autoregressive coefficient changes from 0.1 to 0.3. There are two changes of the process "calibration": at time $t = -2000$, the variance increases from 1 to 2, and at time $t = 2000$, the variance decreases to $\frac{1}{2}$ and values greater than 2 and smaller than -2, respectively, are scaled by the factor 2. According to Remark 1 at the end of Section 2, we are not interested in detecting these changes of the calibration, but only in the change-point at $t = 0$.

The middle plot of Figure 2 shows the distribution of ordinal patterns in time windows of size $w = 500$. The order is $d = 3$, so there are 24 different patterns (compare to Figure 1). Their relative frequencies are represented by the space between horizontal lines: the space between the bottom line and the first line from below represents the relative frequency of the ordinal pattern (0,1,2,3) (the first pattern in Figure 1), the space between the first and the second line from below represents the relative frequency of (0,1,3,2) (the second pattern in Figure 1), and so on. In the framework of Section 2, we have $m = n = 10$, i.e., there are 10 time windows before and after the change-point. As can be seen, the distributions before and after the change-point are slightly different. The bottom plot of Figure 2 shows the resulting MMD and CMMD values for $(m', n') = (1, 19)$, $(2, 18)$, $\ldots$, $(19, 1)$. In all our experiments, we used $\sigma^2 = 1$ in the RBF kernel, however, we found that the results were not very sensitive with respect to the choice of σ^2. The maximal MMD value is obtained for $(m', n') = (19, 1)$, while maximizing the argument of CMMD yields the true value $(10, 10)$.

Comparing MMD and CMMD. In Figure 3, we compare the distribution of the change-point estimators based on MMD and CMMD. Again, the underlying model is an AR(1) process of length $10,000$ and the time window size is $w = 500$. In this experiment, we chose $(m, n) = (5, 15)$, and the autoregressive coefficient changes from 0.1 to 0.2 (top), from 0.1 to 0.3 (middle) and from 0.1 to 0.4 (bottom). The distributions of the estimators are obtained by $1,000$ Monte Carlo replications. As can be seen, the estimator based on CMMD performs better in all three cases; the one based on MMD is biased towards (1,19) and (19,1), particularly if the change of the autoregressive coefficient is small.

Multiple change-point detection. Figure 4 shows the results obtained for the detection of multiple change-points. The underlying model is an AR(1) process of length $10,000$ with a change of the autoregressive coefficient at $2,500$ and $7,500$

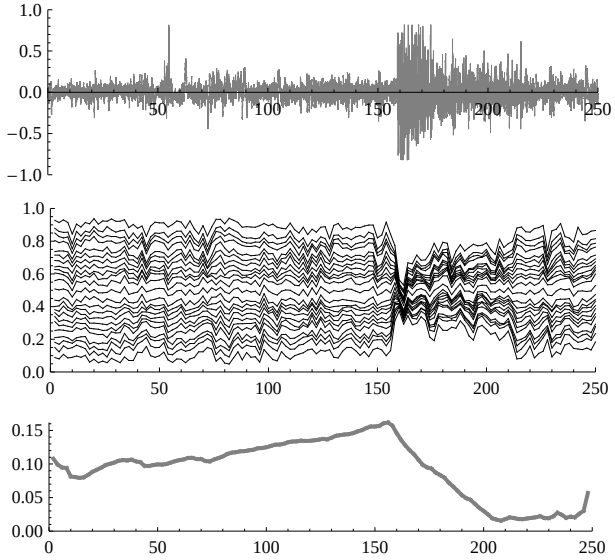


Figure 5: Top: 250 seconds long part of an EEG recording. Middle: Ordinal pattern distributions. Bottom: Resulting MMD values.

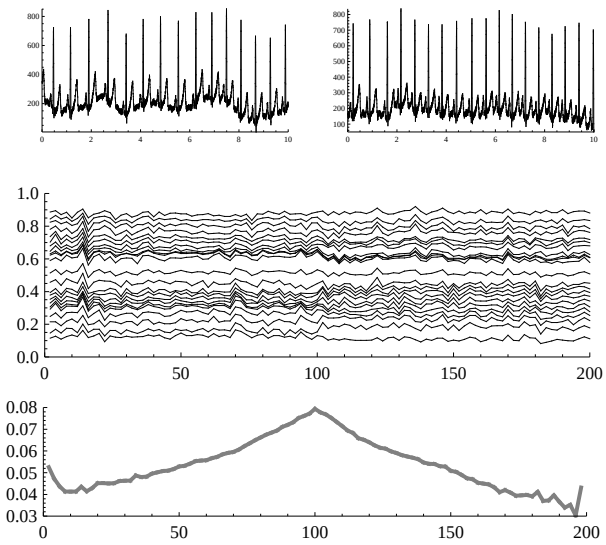


Figure 6: Top: 10 seconds of an ECG before and after an increase of the heart rate. Middle: Ordinal pattern distributions. Bottom: Resulting MMD values.

time points, namely, from 0.1 to 0.4 and 0.4 to 0.1 (left), from 0.1 to 0.3 and 0.3 to 0.1 (middle), and from 0.1 to 0.2 and 0.2 to 0.1 (right). To localize these change-points, we applied our method twice: first to the entire sequence, then to the longer one of the resulting two time segments. Figure 4 displays the distribution of the estimates obtained by $1,000$ Monte Carlo replications. The coloring of the cell (x, y) represents the frequency of replications with the first change-point detected at $500x$ time points and the second one at $500y$ time points. As can be seen, the localization in the first two experiments is fairly precise. In the third experiment, the estimates are more scattered, however, this is true even in the single change-point setting (compare to Figure 3).

4.2 REAL-LIFE TIME SERIES

Epileptic activity in EEG time series. The top plot of Figure 5 shows a 250 seconds long part of an EEG recording which was digitized with a sampling rate of 256 Hertz, so the time series consists of $250 \cdot 256 = 64,000$ data points. The increase in amplitude after 160 seconds is related to the onset of an epileptic seizure. The middle plot shows the distribution of ordinal patterns in non-overlapping time windows of 2 seconds, that is, $w = 512$. Same as in Figure 2, the order of the patterns is $d = 3$. The bottom plot displays the resulting MMD values. The maximum is obtained at the onset of the seizure, thus indicating the presence of a change-point after 160 seconds.

Heart rate changes in ECG data. Figure 6 shows the application of our method to the detection of heart rate changes in electrocardiography (ECG) recordings. The middle plot displays the distribution of ordinal patterns in 100 non-overlapping time windows of 2 seconds, again corresponding to the time window size $w = 512$. After 100 seconds, there is a slight change in the heart rate from approximately 90 to 105 beats per minute (compare to the plots in the top row). As the bottom plot shows, this change-point is easily detected even by the non-corrected MMD values.

5 CONCLUSIONS

We have presented a new method for detecting change-point in time series based on measuring the maximum mean discrepancy of ordinal pattern distributions. The main advantages of our approach are its computational efficiency and robustness towards (non-linear) signal distortions. These properties make it particularly attractive for the fast exploration of biophysical time series the exact calibration of which is unknown or varies with time. We believe that our approach can be efficiently combined with existing change-point detection methods to first approximately localize of a set of candidates before determining the exact location and testing for significance. We have established asymptotical properties of our method and evaluated its performance both for simulated and real-life data, including the application to multiple change-point problems.

References

C. Bandt and B. Pompe (2002). Permutation entropy: A natural complexity measure for time series. *Phys. Rev. Lett. 88*, 174102.

C. Bandt and F. Shiha (2007). Order patterns in time series. *J. Time Ser. Anal.* 28, 646-665.

B.E. Brodsky and B.S. Darkhovsky (1993). *Nonparametric Methods in Change-Point Problems.* Kluwer Academic Publishers, Dordrecht.

B.E. Brodsky and B.S. Darkhovsky (2000). *Nonparametric Statistical Diagnosis.* Kluwer Academic Publishers, Dordrecht.

A.A. Bruzzo, B. Gesierich, M. Santi, C.A. Tassinari, N. Birbaumer and G. Rubboli (2008). Permutation entropy to detect vigilance changes and preictal states from scalp EEG in epileptic patients. A preliminary study. *Neurol. Sci.* 29, 3-9.

J. Chen and A.K. Gupta (2000). *Parametric Statistical Change Point Analysis.* Birkhäuser, Boston.

I.P. Cornfeld, S.V. Fomin and Y.G. Sinai (1982). *Ergodic Theory.* Springer-Verlag, Berlin.

A. Gretton, K.M. Borgwardt, M. Rasch, B. Schölkopf and A.J. Smola (2007a). A Kernel Method for the Two-Sample-Problem. *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, 513-520.

A. Gretton, K.M. Borgwardt, M. Rasch, B. Schölkopf and A.J. Smola, A (2007b). A Kernel Approach to Comparing Distributions. *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 1637-1641.

Z. Harchaoui, F. Bach and E. Moulines (2009). Kernel change-point analysis. *Advances in Neural Information Processing Systems* 21, 609-616.

K. Keller, H. Lauffer and M. Sinn (2007a). Ordinal analysis of EEG time series. *Chaos and Complexity Letters 2*, 247-258.

K. Keller and M. Sinn (2009). A standardized approach to the Kolmogorov-Sinai entropy. *Nonlinearity* 22, 2417-2422.

K. Keller, M. Sinn and J. Emonds (2007b). Time series from the ordinal viewpoint. *Stochastics and Dynamics 2*, 247-272.

X. Li, G. Ouyang and D.A. Richards (2007). Predictability analysis of absence seizures with permutation entropy. *Epilepsy Research* 77, 70-74.

X. Li, S.M.E. Cui and L.J. Voss (2008). Using permutation entropy to measure the electroencephalographic effect of Sevoflurane. *Anesthesiology* 109, 3, 448-456.

E. Olofsen, J.W. Sleigh and A. Dahan (2008). Permutation entropy of the electroencephalogram: a measure of anaesthetic drug effect. *British Journal of Anaesthesia* 110, 6, 810-821.

M. Sinn and K. Keller (2011). Estimation of ordinal pattern probabilities in Gaussian processes with stationary increments. *Computational Statistics and Data Analysis* 55, 4, 1781-1790.

Efficiently Searching for Frustrated Cycles in MAP Inference

David Sontag, Do Kook Choe, Yitao Li
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University

Abstract

Dual decomposition provides a tractable framework for designing algorithms for finding the most probable (MAP) configuration in graphical models. However, for many real-world inference problems, the typical decomposition has a large integrality gap, due to frustrated cycles. One way to tighten the relaxation is to introduce additional constraints that explicitly enforce cycle consistency. Earlier work showed that cluster-pursuit algorithms, which iteratively introduce cycle and other higher-order consistency constraints, allows one to exactly solve many hard inference problems. However, these algorithms explicitly enumerate a candidate set of clusters, limiting them to triplets or other short cycles. We solve the search problem for cycle constraints, giving a nearly linear time algorithm for finding the most frustrated cycle of arbitrary length. We show how to use this search algorithm together with the dual decomposition framework and cluster-pursuit. The new algorithm exactly solves MAP inference problems arising from relational classification and stereo vision.

1 Introduction

Graphical models such as Markov random fields have been successfully applied to many fields, from computer vision and natural language processing, to computational biology. Exact probabilistic inference is generally intractable in complex models having many dependencies between the variables. Here we consider the problem of finding the most probable (MAP) assignment in graphical models with discrete states.

Dual decomposition provides a powerful framework for designing tractable MAP inference algorithms [19]. This approach attempts to minimize an upper bound on the MAP assignment by reparameterization of the potentials. Many optimization algorithms have been proposed to minimize the dual, such as those based on subgradients [10, 13], message-passing approaches based on block coordinate-descent [6, 11, 14, 23], and provably convergent algorithms [7, 9, 16]. Every decomposition corresponds to a particular linear programming (LP) relaxation. The dual LP that is most frequently solved by these algorithms corresponds to the pairwise LP relaxation, which enforces local consistency constraints between factors that share variables.

However, for many real-world inference problems, the pairwise LP relaxation has a large integrality gap. These situations arise because there are frustrated cycles where a fractional solution can obtain a higher objective value by letting the edge marginals along the cycle be globally inconsistent. We can try to avoid these fractional solutions by instead optimizing over a tighter LP relaxation. There are well-known methods for tightening relaxations, such as the Sherali-Adams hierarchy of relaxations which uses cluster consistency constraints to enforce the consistency of all edge marginals in a cluster, for all clusters of some fixed size [15]. However, these linear programs are typically computationally infeasible to solve because they tighten the relaxation uniformly across all of the problem. For graphical models of moderate size, even the first lifting of the Sherali-Adams hierarchy (all triplet clusters, also called the *cycle relaxation*) is too slow to optimize over.

Several authors have proposed cluster-pursuit algorithms that iteratively tighten the dual using cycle

consistency constraints [3, 9, 12, 21, 24]. These approaches are based on dictionary enumeration. They consider a small set of candidate cycles (e.g., all 3-cycles in a graph, or 4-cycles for grids), and evaluate a score to decide which cycle clusters to add to the decomposition. This explicit enumeration limits the applicability of these algorithms to graphical models where it is obvious where to look for the candidate cycles. However, many difficult inference problems are on sparse graphical models with few short cycles, such as the factor graphs used for low-density parity check codes, or Markov random fields whose structure follows that of a social network or the web graph (frequently used for relational classification).

In this paper, we address the most significant shortcoming of these cluster-pursuit algorithms: the difficulty of finding where to tighten the relaxation in sparse graphs. We show in Section 4 that, for non-binary graphical models, it is NP-hard to find the best cycle according to the bound criterion score used in earlier work [21]. Thus, we need an alternative approach to address the search problem of finding where to tighten the relaxation.

We describe a new approach where, as before, we solve the dual of the pairwise LP relaxation, but where we now search for k-ary cycle inequalities [20] to tighten the relaxation rather than cluster consistency constraints. We consider the same greedy bound minimization criterion used earlier for cluster consistency constraints, corresponding to the amount that the dual objective would decrease with just one coordinate descent step on the new dual variable(s). We show that one can, in near-linear time (in the size of the projection graph), find a k-ary cycle inequality whose guaranteed bound decrease is a constant fraction of the best possible bound decrease according to this criterion. The resulting algorithm is similar to the separation algorithm given in [20] for finding the most violated cycle inequality in the primal, but is not based on shortest paths. We use the cycle inequalities only as a means of obtaining an efficient search algorithm, and we add the full cycle cluster to the dual decomposition as in previous work.

Somewhat surprisingly, in Section 4 we show that, for *binary* graphical models and when the dual has been solved to optimality, the two bound criterions (the one given in [21] and this one) coincide. Thus, at least for binary graphical models, the algorithm that we present completely solves the open problem from [21] of how to efficiently search over cycle clusters.

2 Background

We consider MAP inference problems on factor graphs with n variables $X_1, \ldots, X_n$, where each variable takes discrete states $x_i \in \text{Vals}(X_i)$. The MAP inference problem is then

$$\text{MAP}(\theta) = \max_{\mathbf{x}} \sum_{\mathbf{c} \in C} \theta_{\mathbf{c}}(\mathbf{x_c}), \tag{1}$$

where C denotes the set of factors, and $\theta_{\mathbf{c}}(\mathbf{x_c})$ is the log of the potential function for factor $\mathbf{c}$. Let $G = (V, E)$ be the Markov network corresponding to this factor graph, with one edge $ij \in E$ for every two variables i and j that appear together in some factor. The notation $N(i)$ refers to the set of variables that neighbor i in the Markov network.

2.1 Dual Decomposition

The dual decomposition approach [19] attempts to address the intractability of the joint maximization in Eq. 1 by introducing dual variables δ and minimizing an upper bound on the MAP assignment:

$$\min_{\delta} L(\delta), \quad L(\delta) = \sum_{\mathbf{c} \in C} \max_{\mathbf{x_c}} \theta^{\delta}_{\mathbf{c}}(\mathbf{x_c}), \tag{2}$$

where the reparameterizations $\theta^{\delta}_{\mathbf{c}}(\mathbf{x_c})$ are given by

$$\begin{aligned}
\theta^{\delta}_i(x_i) &= \theta_i(x_i) + \sum_{j \in N(i)} \delta_{ij \to i}(x_i) \qquad \forall i \in V, \\
\theta^{\delta}_{ij}(x_i, x_j) &= \theta_{ij}(x_i, x_j) - \delta_{ij \to i}(x_i) - \delta_{ij \to j}(x_j) \\
&\quad + \sum_{\mathbf{c}: |\mathbf{c}| \geq 3,\ i,j \in \mathbf{c}} \delta_{\mathbf{c} \to ij}(x_i, x_j) \quad \forall ij \in E, \\
\theta^{\delta}_{\mathbf{c}}(\mathbf{x_c}) &= \theta_{\mathbf{c}}(\mathbf{x_c}) - \sum_{i,j \in \mathbf{c}} \delta_{\mathbf{c} \to ij}(x_i, x_j) \quad \forall |\mathbf{c}| \geq 3.
\end{aligned}$$

The key property of the function $L(\delta)$ is that it only involves maximization over local assignments $\mathbf{x_c}$, a task which we assume to be tractable. The dual thus decouples the original problem, resulting in a problem that can be optimized using local operations.

If we ever find a global assignment $\mathbf{x}$ which locally maximizes all of the subproblems, then $\mathbf{x}$ is guaranteed to be the MAP assignment. Thus, the dual solution δ has the ability to provide a *certificate of optimality*. Even in cases when the relaxation is loose, the dual provides an upper bound, $\text{MAP}(\theta) \leq L(\delta)$, that can be useful within branch-and-bound.

The algorithms described in this paper make use of the reparameterized edge potentials $\theta^{\delta}_{ij}(x_i, x_j)$ when searching for frustrated cycles. For notational clarity and to be consistent with earlier work, we use

$b_{ij}(x_i, x_j)$ to denote $\theta^{\delta}_{ij}(x_i, x_j)$ for the current dual variables δ, also calling these the edge "beliefs".

2.2 Cluster Pursuit

If there still remains an integrality gap after solving the current dual, i.e. $\text{MAP}(\theta) < L(\delta^*)$, one can *tighten* the relaxation by introducing new zero-valued potentials $\theta_{\mathbf{c}}(\mathbf{x_c})$ for clusters $\mathbf{c}$ not originally in the factor graph [21, 24]. The advantage of doing this together with dual decomposition is that one can *warm start*, initializing the new dual's variables using the previous dual solution. The resulting algorithm alternates between minimizing the current dual for some number of iterations (not necessarily to optimality) and searching for new clusters to use in tightening the relaxation. If the dual is solved using coordinate-descent, then every step of the algorithm monotonically improves the upper bound on the MAP value.

The key problem addressed by earlier work was how to choose *which* clusters to use to tighten the relaxation. In particular, Sontag et al. [21] proposed to evaluate the utility of adding a cycle C to the relaxation by the greedy bound minimization criterion

$$d(C) = \sum_{e \in C} \max_{\mathbf{x}_e} b_e(\mathbf{x}_e) - \max_{\mathbf{x}_C} \left[\sum_{e \in C} b_e(\mathbf{x}_e) \right]. \quad (3)$$

Note that only the edges in the cycle C are used in the maximization over $\mathbf{x}_C$. As a result, $d(C)$ can be computed in $O(k^3|C|)$ time, where k is the number of states for each variable. The criterion corresponds to the amount that the dual would decrease if we add this cycle cluster and perform one block coordinate descent step on all dual variables corresponding to the new cluster. Once a cycle is selected for addition, [21] triangulates the cycle and adds each of the triplet clusters; we do this too.

2.3 Linear Programming Relaxation

The dual decomposition for MAP inference given in Eq. 2 can be shown to be equivalent, by LP duality, to the following linear programming relaxation:

$$\max_{\boldsymbol{\mu} \in M_L} \sum_{\mathbf{c}} \sum_{\mathbf{x_c}} \theta_{\mathbf{c}}(\mathbf{x_c}) \mu_{\mathbf{c}}(\mathbf{x_c}) \quad (4)$$

where the *local marginal polytope* M_L enforces that $\{\mu_i(x_i), \forall x_i\}$ and $\{\mu_{\mathbf{c}}(\mathbf{x_c}), \forall \mathbf{x_c}\}$ correspond to valid (local) probability distributions and that, for each factor $\mathbf{c}$, $\mu_{\mathbf{c}}(\mathbf{x_c})$ is consistent with $\mu_i(x_i)$ for all $i \in \mathbf{c}, x_i$:

$$M_L = \left\{ \boldsymbol{\mu} \geq 0 : \begin{array}{ll} \sum_{x_i} \mu_i(x_i) = 1, & \forall i \\ \sum_{\mathbf{x}_{\mathbf{c}\setminus i}} \mu_{\mathbf{c}}(\mathbf{x_c}) = \mu_i(x_i) & \\ \sum_{\mathbf{x}_{\mathbf{c}\setminus\{i,j\}}} \mu_{\mathbf{c}}(\mathbf{x_c}) = \mu_{ij}(x_i, x_j) & \end{array} \right\},$$

where the second set of constraints is for all $\mathbf{c}, i \in \mathbf{c}, x_i$, and the third set of constraints is for all factors $\mathbf{c}$ such that $|\mathbf{c}| \geq 3$, $i, j \in \mathbf{c}, x_i, x_j$.

The exact MAP inference problem would be obtained if we instead had maximized Eq. 4 over the *marginal polytope* [22], which enforces that all feasible points $\boldsymbol{\mu}$ correspond to marginals arising from some exponential family distribution with the same sufficient statistics. For most graphical models the marginal polytope is intractable to optimize over, which is why we use the relaxation M_L.

2.4 k-ary Cycle Inequalities

The main contribution of this paper is to show how to solve the search problem for cycle consistency constraints (within the framework of dual decomposition) by introducing a new bound criterion based on the k-ary cycle inequalities [20].

The cycle inequalities [1, 2, 4] are a set of constraints for the marginal polytope of graphical models with binary variables, which arises from the observation that a cycle must have an even (possibly zero) number of cut edges. Suppose we start at node i, where $x_i = 0$. As we traverse the cycle, the assignment changes each time we cross a cut edge. Since we must return to $x_i = 0$, the assignment can only change an even number of times. This concept can be generalized to obtain the k-ary cycle inequalities, which are valid constraints for graphical models with non-binary variables [20].

For each variable i, let π_i^q be a *partition* of all of its states into two disjoint non-empty sets. Let π_i denote the set of all partitions of variable i. The k-ary cycle inequalities are defined using the *projection graph* $G_\pi = (V_\pi, E_\pi)$, which has one node for each partition in each set π_i and all such nodes are fully connected across adjacent variables. That is, we have $V_\pi = \bigcup_{i \in V} \pi_i$, and

$$E_\pi = \{(\pi_i^q, \pi_j^r) \mid (i,j) \in E,\, q \leq |\pi_i|,\, r \leq |\pi_j|\}. \quad (5)$$

We obtain a different projection graph depending on the quantity and type of partitions that we choose for each variable. The algorithms in this paper are applicable for *any* projection graph. In our experimental results, we primarily use the k-projection

graph, which simply has a partition $\pi_i^{x_i} = \{x_i\}$ (versus all other states) for every variable i and state x_i. Thus, if every variable takes k states, the projection graph will have $k|V|$ nodes and $k^2|E|$ edges. In the supplementary material we describe an algorithm which finds the optimal partition for each variable with respect to each edge. If using the k-projection graph does not succeed in finding a frustrated cycle, we re-run the cycle search method using this expanded set of partitions.

There is one k-ary cycle inequality for every cycle C in the projection graph G_π and for every set of edges $F \subseteq C$ such that $|F|$ is odd:

$$\sum_{mn \in C \setminus F} \sum_{\substack{x_i, x_j : \\ \pi_i^q(x_i) \neq \\ \pi_j^r(x_j)}} \mu_{ij}(x_i, x_j) + \sum_{mn \in F} \sum_{\substack{x_i, x_j : \\ \pi_i^q(x_i) = \\ \pi_j^r(x_j)}} \mu_{ij}(x_i, x_j) \geq 1$$

where $mn = (\pi_i^q, \pi_j^r) \in E_\pi$. Although there are exponentially many k-ary cycle inequalities, [20] showed how the most violated inequality can be found in polynomial time, using a shortest-path algorithm. However, unlike [20] we use these inequalities in the *dual*. In the next section, we will give a new algorithm to efficiently find violated k-ary cycle inequalities while working within the framework of dual decomposition.

3 Cycle Inequalities in the Dual

In this section we give a column-generation approach for adding cycle inequalities within the dual decomposition framework. Consider the terms of the dual objective (Eq. 2) that would be affected by adding a single dual variable $\lambda_{C,F,\pi}$ corresponding to one k-ary cycle inequality specified by $C \subseteq E$, $F \subseteq C$ (recall that $|F|$ must be odd), and π.[1] After adding $\lambda_{C,F,\pi}$, the new dual has the following terms (see supplementary material for derivation):

$$\sum_{ij \in F} \max_{x_i, x_j} \Big(b_{ij}(x_i, x_j) + \lambda_{C,F,\pi} \mathbf{1}[\pi_i(x_i) = \pi_j(x_j)] \Big)$$
$$+ \sum_{ij \in C \setminus F} \max_{x_i, x_j} \Big(b_{ij}(x_i, x_j) + \lambda_{C,F,\pi} \mathbf{1}[\pi_i(x_i) \neq \pi_j(x_j)] \Big)$$
$$- \lambda_{C,F,\pi}.$$

For each edge $ij \in C$, define the weight

$$s_{ij}^\pi = \max_{x_i, x_j \,:\, \pi_i(x_i) = \pi_j(x_j)} b_{ij}(x_i, x_j) - \max_{x_i, x_j \,:\, \pi_i(x_i) \neq \pi_j(x_j)} b_{ij}(x_i, x_j).$$

[1] When used in the context of a cycle in the graphical model, $C \subseteq E$, as opposed to a cycle in the projection graph, the notation π specifies a particular partition for each variable along the cycle.

We show in the supplementary material that the coordinate descent step for $\lambda_{C,F,\pi}$ is given by $\lambda_{C,F,\pi} = \min_{ij \in C} w_{ij}^{\pi,F}$, where $w_{ij}^{\pi,F} = s_{ij}^\pi$ for $ij \notin F$ and $w_{ij}^{\pi,F} = -s_{ij}^\pi$ for $ij \in F$.

The amount that the dual objective decreases with one coordinate descent step on $\lambda_{C,F,\pi}$, assuming that $\lambda_{C,F,\pi}$ was previously zero, is

$$d(C, F, \pi) = \max(0, \min_{ij \in C} w_{ij}^{\pi,F}). \tag{6}$$

Example 1. Consider a triangle on three edges ($C = \{12, 23, 31\}$), with $x_i \in \{0, 1\}$, where $\theta_i(x_i) = 0$ $\forall i, x_i$ and $\theta_{ij}(x_i, x_j) = 1$ if $x_i \neq x_j$, and 0 otherwise. Since this example is binary, we simply use $\pi_i(x_i) = x_i$. Let all of the dual variables δ be 0, and assume that initially there are no cycle inequalities. The best integer solution has value 2, while the pairwise LP relaxation gives only a loose upper bound of 3 (note: δ as defined can be shown to be optimal for the dual, i.e. Eq. 2).

Consider the problem of finding the best cycle inequality according to $\arg\max_{C,F} d(C, F)$. First, note that $b_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j)$, so $w_{ij}^F = 1$ for $ij \in F$ and $w_{ij}^F = -1$ for $ij \notin F$. If $F = \emptyset$, then $w_{ij}^F = -1$ for all edges, and so $d(C, F) = 0$. On the other hand, if $F = C$, then $w_{ij}^F = 1$ for all edges, which gives a bound decrease of $d(C, F) = 1$, corresponding to $\lambda_{C,F} = 1$.

3.1 Separation Algorithm

The bound criterion $d(C, F, \pi)$ given in Eq. 6 is analogous to the bound criterion $d(C)$ used by [21] (see Eq. 3) to evaluate the utility of adding a cycle C to the relaxation. We now consider the algorithmic problem of finding the *best* dual variable $\lambda_{C,F,\pi}$ to add, according to $d(C, F, \pi)$:

$$\max_{C, F \subseteq C \text{ s.t. } |F| \text{ odd}, \pi} d(C, F, \pi). \tag{7}$$

We show that this can be computed efficiently using a variation on the shortest-path based separation algorithm for k-ary cycle inequalities [20]. We first note that Eq. 7 is equal to

$$\max \Big(0, \max_{C, F \subseteq C \text{ s.t. } |F| \text{ odd}, \pi} \min_{ij \in C} w_{ij}^{\pi,F}\Big). \tag{8}$$

We can simplify this further by noticing that for $d(C, F, \pi)$ to be positive, we need $ij \in F$ when $s_{ij}^\pi < 0$, and $ij \notin F$ when $s_{ij}^\pi > 0$. Thus, ignoring the maximization over the partitioning π, Eq. 8

Algorithm FindOddCycle (Graph G_π, edge weights $\{s_{mn} : (m,n) \in E_\pi\}$)

```
Construct a spanning tree T of G_π. Set r to be the root of T.
sign[r] ← +1
for each vertex t ∈ T (in order of increasing distance from r in T):
    sign[t] ← sign[pa(t)] · s_{t,pa(t)}
for each (m,n) ∈ E_π \ T:
    if sign[m] ≠ sign[n] · s_mn
        return cycle given by the edge (m,n) and the path m ⇝ r ⇝ n in T

return no odd signed cycle found
```

Figure 1: Assuming that the edge weights s_{mn} are in $\{-1, 1\}$, this algorithm will find a cycle with an odd number of -1 edges, if one exists. $\text{pa}(t)$ refers to the parent of node t in the spanning tree T.

is equivalent to

$$\max\Big(0, \max_{\substack{C \subseteq E \text{ s.t.} \\ \prod_{ij \in C} \text{sign}(s^\pi_{ij}) = -1}} \min_{ij \in C} |s^\pi_{ij}|\Big). \quad (9)$$

We conclude that the maximum bound decrease is achieved by the cycle in G with an odd number of negative weight edges that maximizes the minimum of the absolute value of the weights along the cycle.

For Markov networks with non-binary variables, we also wish to maximize over the partition for each variable. Let G_π denote the projection graph, where the variables and edges are as defined in Eq. 5. All subsequent quantities will use the partitions for the variables specified by the edges $mn = (\pi^q_i, \pi^r_j) \in E_\pi$ in the projection graph. Assign weights to the edges[2]

$$s_{mn} = \max_{x_i, x_j : \pi^q_i(x_i) = \pi^r_j(x_j)} b_{ij}(x_i, x_j) - \max_{x_i, x_j : \pi^q_i(x_i) \neq \pi^r_j(x_j)} b_{ij}(x_i, x_j), \quad (10)$$

and remove all edges with $s_{mn} = 0$. The algorithm that we describe next will find the maximum of

$$\max\Big(0, \max_{\substack{C \subseteq E_\pi \text{ s.t.} \\ \prod_{mn \in C} \text{sign}(s_{mn}) = -1}} \min_{mn \in C} |s_{mn}|\Big). \quad (11)$$

Suppose that $s_{mn} \in \{+1, -1\}$. Then, $\min_{mn \in C} |s_{mn}| = 1$ and the optimization problem simplifies to finding a cycle with an odd number of -1 edges. This can be solved in linear time by the algorithm given in Figure 1 (if the graph is not connected, do this for each component).[3]

[2]For the k-projection graph, this step can be implemented efficiently, taking time $O(k^2|E|)$ to compute all edge weights instead of $O(k^2|E_\pi|)$.

[3]In the case when there is more than one cycle with an odd number of -1 edges, the particular cycle that we find depends on the choice of spanning tree. However, the algorithm is always guaranteed to find *some* cycle with an odd number of -1 edges, when one exists, regardless of the choice of spanning tree.

Now consider the case of general s_{mn}. We can solve the optimization in Eq. 11 by doing a binary search on $|s_{mn}|$. There are only $|E_\pi|$ possible edge weights, so to do this search we first sort the values $\{|s_{mn}| : mn \in E_\pi\}$. At each step, we consider the subgraph G' consisting of all $mn \in E_\pi$ such that $|s_{mn}| > R$, where R is the threshold used in the binary search. We then let $s_{m'n'} = \text{sign}(s_{mn})$ for $m'n' \in G'$, and search for a cycle with an odd number of negative edges using the algorithm described in Figure 1. The binary search will find the largest R such that there is a negative-signed cycle $C \in G'$, if one exists. The best choice of $\lambda_{C,F,\pi}$ is then given by this cycle C, the edges F corresponding to the negative-weight edges in C, and π given by the partitions used in C. The total running time is only $O(|E_\pi| \log |E_\pi|)$.

If the cycle that is returned uses each variable only once (i.e., does not consider more than one partition for a single variable), the corresponding cycle and choice of partitions will be optimal for Eq. 8. Otherwise, one can show that the guaranteed bound decrease after one coordinate descent step on the new cycle inequality's dual variable is a constant fraction of that guaranteed by Eq. 8.

When we use this algorithm in the experiments of Section 6, we ignore F and π, instead fully enforcing cycle consistency for the cycle C that is best according to this bound criterion.

4 Theoretical Results

Consider the restricted set of clusters $\mathcal{C}_{\text{cycles}}(G)$ corresponding to cycles of arbitrary length,

$$\mathcal{C}_{\text{cycles}}(G) = \Big\{C \subseteq E \mid C \text{ forms a cycle in } G\Big\}. \quad (12)$$

A natural question is whether it is possible to find the best cycle cluster to add to the relaxation according to the greedy bound minimization criteria $d(C)$

[21]. It is easily shown that $\max_{F,\pi} d(C,F,\pi) \leq d(C)$ for all cycles C. Thus, if it were not for computational concerns, searching using $d(C)$ would be optimal. Unfortunately, we show a number of negative results proving that searching for the best cycle according to $d(C)$ is computationally intractable.

We show the following, where k refers to the number of states per node.

1. For $k = 2$, when the beliefs $b_e(\mathbf{x}_e)$ are dual optimal, maximizing Eq. 3 is *equivalent* to finding the best cycle inequality in the dual.[4]
2. For $k = 2$, maximizing Eq. 3 is NP-hard when the beliefs $b_e(\mathbf{x}_e)$ are not dual optimal.
3. For $k > 2$, maximizing Eq. 3 is always NP-hard.

By dual optimal, we mean that the beliefs correspond to a dual optimal solution of the current LP relaxation. Note that, before solving the dual LP to optimality, $b_e(\mathbf{x}_e)$ can be almost anything. For example, we can set $\theta_e(\mathbf{x}_e)$ arbitrarily and consider the separation problem at the first iteration.

Theorem 4.1. *When $k = 2$ and the beliefs $b_{ij}(x_i, x_j)$ correspond to a dual optimal solution, $\max_{C \in \mathcal{C}_{cycles}(G)} d(C) = \max_{C,F:|F| \text{ odd}} d(C,F)$.*

The proof of Theorem 4.1 makes use of the assumption that $b_{ij}(x_i, x_j)$ corresponds to a dual optimal solution in only one step, when applying the complementary slackness conditions for the edge variables. Thus, Theorem 4.1 holds for any dual decomposition for MAP which is at least as tight as the pairwise LP relaxation. In particular, adding cycle constraints does not change the premise of the theorem.

One conclusion that is immediate given Theorem 4.1 is that (for binary MRFs) the cycle inequalities give at least as tight of a relaxation as the cycle relaxation. In fact, for a single cycle, just one cycle inequality suffices to make the LP relaxation tight for a given instance. This is precisely what we observed in Example 1.

4.1 NP-Hardness Results

It is often possible to obtain much better results by tightening the relaxation even before the dual is solved to optimality [21]. Unfortunately, although we showed in Section 3 that for k-ary cycle inequalities $\max_{C,F,\pi} d(C,F,\pi)$ can be computed efficiently, the corresponding problem for cycle consistency constraints is significantly more difficult:

[4]This result is for binary variables only, so we let the projection be $\pi_i(x_i) = x_i$ and omit the π notation.

Theorem 4.2. *The optimization problem $\max_{C \in \mathcal{C}_{cycles}(G)} d(C)$ is NP-hard for $k = 2$ and beliefs $b_{ij}(x_i, x_j)$ arising from a non-optimal dual feasible point.*

We next show that, for $k > 2$, not even dual optimality helps:

Theorem 4.3. *The optimization problem $\max_{C \in \mathcal{C}_{cycles}(G)} d(C)$ is NP-hard for $k \geq 3$ even for beliefs $b_{ij}(x_i, x_j)$ corresponding to a dual optimal solution of the pairwise relaxation.*

Both proofs use a reduction from the Hamiltonian cycle problem. Full proofs can be found in the supplementary material.

5 Related Work

Our new algorithm is closely related to two earlier approaches for tightening the dual using cycle constraints. First, Komodakis et al. proposed to tighten the pairwise LP relaxation by a sequence of cycle repairing operations [12]. Their algorithm is applicable to graphical models with non-binary variables. For binary graphical models, when the dual is at optimality, two cycle repairs – corresponding to the two anchors of any variable (using their terminology) – can be seen to be equivalent to one coordinate descent step on a new cycle inequality for this cycle. We solve the open problem of how to *find* the cycles where cycle repairs are necessary. In their experiments, [12] explicitly enumerated over short cycles.

Second, Johnson proposed an algorithm to find *inconsistent* cycles in the dual [9]. His approach applies only to binary-valued graphical models, and only when the dual is close to optimality. In these cases, his algorithm can be shown to find a cycle C such that $d(C,F) > 0$ for some $F \subseteq C$, $|F|$ odd. His algorithm, which inspired our approach, constructs $s_{ij} \in \{+1, -1\}$ and looks for inconsistent cycles using the linear time method described in Section 3.1. Because of numerical difficulties, Johnson needed to use an edge-wise correlation measure, computed using the primal solution obtained from the smoothed dual [9, p.134]. By drawing the connection to cycle inequalities, we obtain a *weighted* approach whereas his was *unweighted*. As a result, there are no numerical difficulties, and our algorithm can be applied long before solving the dual to optimality.

It may seem surprising that the dual separation algorithm is so much faster than the primal separation algorithm for k-ary cycle inequalities [20]. However, this is because the dual searches over a smaller class

of cycle inequalities. Consider the case where we have solved the dual to optimality for the current relaxation. Then, using the complementary slackness conditions one can show that, for any cycle inequality C, F, π such that $d(C, F, \pi) > 0$, and for any primal solution μ,

$$\sum_{mn \in C \setminus F} \left(\mu^{\pi}_{mn}(0,1) + \mu^{\pi}_{mn}(1,0) \right) + \sum_{mn \in F} \left(\mu^{\pi}_{mn}(0,0) + \mu^{\pi}_{mn}(1,1) \right) = 0. \quad (13)$$

The right hand side could be anything less than 1 for us to obtain a violated cycle inequality, and it is always non-negative because $\mu_{ij}(x_i, x_j) \geq 0$. But, since the right hand side of Eq. 13 is 0, we conclude that the dual separation algorithm is only able to find cycle inequalities to add to the relaxation that are *very violated*. By Theorem 4.1, the same conclusion holds for binary graphical models for the cluster-pursuit algorithm given in [21], when applied to a dual optimal solution – if a triplet cluster C is found such that $d(C) > 0$, then there exists a cycle inequality C, F that is very violated by all primal solutions. It is also possible to give a $O(|E_\pi|)$ time separation algorithm in the primal that would separate this smaller class of k-ary cycle inequalities.

6 Experiments

In this section we report experimental results for the new cycle search algorithm (we call this the "cycle" method) applied to MAP inference problems arising from predicting protein-protein interactions [5, 8], protein side-chain placement [25], and stereo vision [25]. We compare the cycle method to cluster-pursuit using dictionary enumeration with triplet clusters [21], which we call the "triplet" method. Both algorithms were implemented using C++, and differ only in the mechanism used for cycle search. All experiments were performed on a 2.4 GHz AMD Opteron(tm) machine with 128 GB of memory.

We use the Max-Product Linear Programming (MPLP) algorithm to minimize the dual [6]. On all problems, we start by running MPLP for 1000 iterations. If the integrality gap is less than 10^{-4}, the algorithm terminates. Otherwise, we alternate between further tightening the relaxation (using either the cycle or triplet method) and running another 20 MPLP iterations, until the problem is solved optimally or a time limit is reached. For the triplet algorithm, we add between 5 and 20 triplets per outer iteration, and for the cycle algorithm, we add from 1 to 5 cycles per outer iteration (we required at least 5 new triplet clusters).

In our experiments we also consider a combination of the two approaches, which we call the "triplet+cycle" method, where we add the clusters found by *both* cycle search algorithms.

Biasing toward short cycles. Whereas previous cluster-pursuit approaches considered clusters of only three or four variables, our algorithm could potentially choose to add a cycle involving a large number of variables. Consequently, the *length* of the cycle added can significantly affect the per-iteration running time. Our algorithm must balance between (a) choosing cycles which most improve the dual objective and (b) keeping the per-iteration running time reasonable.

The length of the cycles found depends both on the initial *choice of the spanning tree* (line 1 of algorithm FindOddCycle, given in Figure 1) and on *which edge is chosen* to create the cycle (lines 5–7). Notice that the length of any cycle returned by FindOddCycle is at most twice the depth of the spanning tree T. We found that using breadth-first search to create the spanning tree results in much more shallow trees, and as a result significantly shorter cycles compared to depth-first search.

After creating the spanning tree and propagating the signs, any edge that has opposite signs on its endpoints could be chosen to create the cycle (lines 5–7). The cycle corresponds to the union of the edge and the paths from each endpoint to their least common ancestor (LCA) in T. We find the lengths of each of these cycles by running a LCA algorithm (we note that this can be done in amortized constant time using the algorithm of [17]). Then, we sort the edges in increasing order according to the length of the corresponding cycles, and return the shortest few.

We also experimented with an algorithm which is guaranteed to find the shortest cycles, but found that its running time was too long to be practical.

6.1 Protein-Protein Interaction

We consider 8 inference problems arising from the relational classification task of protein-protein interaction prediction [5, 8]. This inference problem aims at predicting protein-protein interactions (PPIs) given high-throughput protein-protein interaction and other cellular experimental data. The PPI problems are Markov networks with over 14,000 binary variables denoting the cellular localization of the proteins and whether any particular pair of pro-

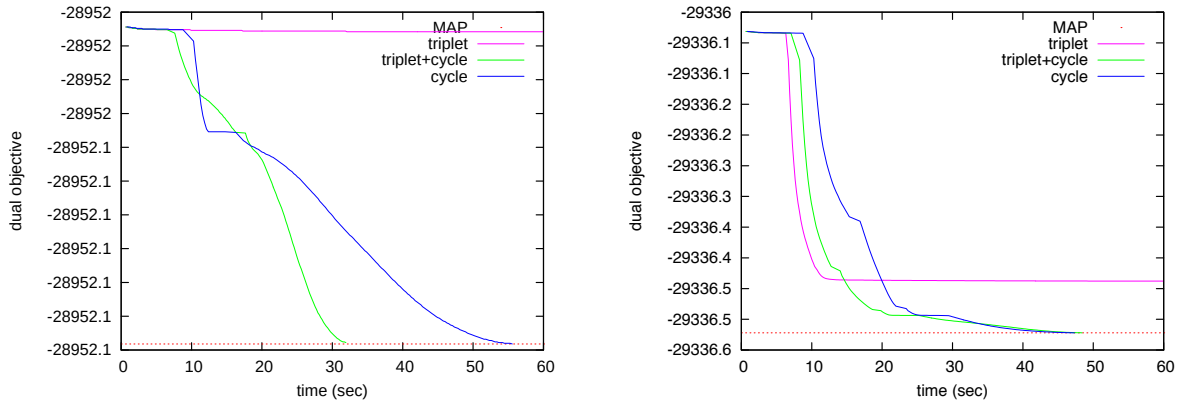


Figure 2: Comparison of the cycle search algorithms on a MAP inference problem from protein-protein interaction prediction. The time shown is for the tightening stage, *after* the initial 1000 iterations of ADLP.

teins interact. The Markov network has over 30,000 node and triad potentials. These inference problems are difficult to solve because the triad potentials induce many frustrated cycles. Each of the 8 inference problems corresponds to the parameters found at some iteration during learning, and the difficulty of inference varies substantially among them.

For 2 of the PPI problems, the cycle method finds the MAP assignment in less than a minute after solving the initial dual to optimality (see Figure 2).[5] For the other 6 problems, the cycle method obtains a slightly better dual objective value than the triplet method when we terminated at half an hour.

In contrast, the triplet method is unable to exactly solve any of the PPI problems. We noticed that the bound criterion [21] for most triplets found in these problems is close to 0, which explains the considerable difficulty for the triplet algorithm to quickly choose the best clusters to add to the relaxation.

These results indicate that the cycle algorithm has a significant advantage over the triplet algorithm on inference problems on sparse graphs.

6.2 Side-Chain Prediction

We next considered 30 protein side-chain placement inference problems, corresponding to the 30 proteins from [25] for which the pairwise LP relaxation is not tight. Previous work has shown that the triplet method exactly solves these [21]. The triplet algorithm's running time ranges from 1.12 to 182.18 seconds with a median of 20.42 seconds.

[5]For the PPI problems MPLP converges very slowly, so we instead use the ADLP algorithm [16] to solve the initial dual, before tightening.

The cycle method also finds the MAP assignment for all 30 proteins. The cycle algorithm's running time is between 2.92 and 3788 seconds, with a median of 24.83 seconds. The outlier which took 3788 seconds to solve was '1kmo'. The cycle method performs significantly worse on this example than the triplet method because the former needs 94 tightening iterations to find the single cluster that solves MAP exactly, whereas the latter finds it immediately.

The cycle algorithm using *only* the k-projection graph does not exactly solve '1qb7' and '1rl6'.[6] We show in Figure 3 the dual objective for '1qb7'. At around 50 seconds the cycle method, not finding any useful cycle with the k-projection graph, runs the partition finding algorithm explained in the supplementary section, finds the necessary cluster, and solves the protein to optimality.

The "triplet+cycle" method solves all of the side-chain placement problems quickly. As seen in Figure 3, it is slightly slower than the triplet method because of the additional time for running the cycle search method. From these results, we conclude that it is best to use both search methods, when feasible.

6.3 Importance of bound criterion

As we mentioned in the related work, for graphical models with binary variables, our cycle search method is similar to the algorithm earlier proposed by Johnson [9]. The key difference is that our approach has a bound criterion which is used to select *which* frustrated cycle to include, whereas Johnson's approach was unweighted (finding *any* odd-signed cycle). In this section we illustrate the importance of having the weighted bound criterion. In addition

[6][20] also needed the full projection graph for '1rl6'.

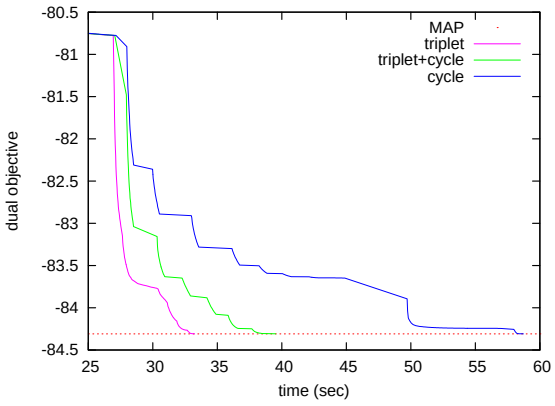


Figure 3: Comparison of all 3 methods on the protein-side chain placement problem '1qb7'.

to the 30 protein side-chain placement problems that we solved exactly, we also consider 4 inference problems arising from stereo vision (all variants of the "Tsukuba" image sequence), which were previously studied by [21, 25].

After finding the optimal value of the bound criterion, R (the threshold used in the binary search, described in Section 3.1), we prune the projection graph to consider only those edges with $|s_{mn}| \geq R/c$ for $c = 1, 4$, and 128. $c = 1$ corresponds to the usual cycle method, whereas $c = 128$ is more similar to an unweighted bound criterion (analogous to [9]).

We considered a protein failed if we could not obtain a certificate within 15 minutes. On the protein side-chain problems, we found that $c = 4$ worked best, solving all proteins but '1kmo' within 3.2 minutes. With $c = 1$, the cycle method additionally failed to solve '1ug6'. With $c = 128$, the cycle method additionally failed to solve '1a8i', '1et9' and '1gsk' (5 failures in total). One reason why $c = 4$ may be preferable over $c = 1$ for non-binary problems is because – since more edges are considered – the cycle method is more likely to find a shorter length cycle which uses each variable only once, rather than multiple times with different partitions.

When testing on stereo vision data, we found that the cycle algorithm was able to solve all 4 problems exactly within 1 hour with $c = 1$, but could not solve any of them within 3 hours for $c = 128$. We conclude that the bound criterion is essential for the cycle method to quickly solve MAP inference problems.

After observing that varying the number of MPLP iterations in each of the outer loops (i.e., after adding cycles) had a very minor effect on the overall running times, we also conclude that the cycle method is robust to the dual not being solved to optimality.

6.4 Summary of results

Using the "triplet+cycle" method, we exactly and quickly solved all 30 protein side-chain problems and 2 protein-protein interaction problems.

The cycle algorithm solved all 4 stereo problems exactly within 24–48 minutes using only 9–33 tightening iterations. The number of clusters added by the algorithm were between 227 and 1217.

7 Discussion

We believe that our algorithm for finding frustrated cycles may also be useful in graphical models that are not sparse. In these settings, dictionary enumeration methods which tighten the relaxation one triplet cluster at a time can eventually succeed at making all cycles consistent. However, the cycle method has several theoretical advantages. First, the dual bound criterion can be zero for some triplet clusters, but non-zero for a larger cycle involving the same variables; we give a concrete example of this in the supplementary material. In these cases, the cycle method would succeed at improving the dual objective, whereas the triplet method would obtain no guidance from the bound criterion and as a result would randomly choose a triplet to use in tightening the relaxation. Second, if a frustrated cycle is long, the triplet method – which adds clusters one at a time – would take several iterations before eventually adding all triplet clusters which triangulate the cycle, thus enforcing cycle consistency. In contrast, the cycle method would take a single iteration.

In this paper, we used the dual variables corresponding to the cycle inequalities only as a means for finding a frustrated cycle, after which we fully enforce cycle consistency. When the variables have a large number of states, enforcing even one cycle consistency constraint can significantly increase the computation time [18]. In these situations, it may be preferable to instead directly do coordinate descent with respect to the cycle inequality dual variables.

Finally, although in this paper we used the dual algorithm as a stand-alone MAP solver, for some problems it may be more effective to use within branch-and-bound, giving a branch-and-cut approach. The efficient cycle search algorithm can also be used together with any method for solving the dual decomposition, such as ADLP [16].

References

[1] F. Barahona. On cuts and matchings in planar graphs. *Mathematical Programming*, 60:53–68, 1993.

[2] F. Barahona and A. R. Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.

[3] D. Batra, S. Nowozin, and P. Kohli. Tighter relaxations for MAP-MRF inference: A local primal-dual gap based separation algorithm. *Journal of Machine Learning Research - Proceedings Track*, 15:146–154, 2011.

[4] M. M. Deza and M. Laurent. *Geometry of Cuts and Metrics*, volume 15 of *Algorithms and Combinatorics*. Springer, 1997.

[5] G. Elidan, I. Mcgraw, and D. Koller. Residual belief propagation: informed scheduling for asynchronous message passing. In *UAI*, 2006.

[6] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NIPS*. MIT Press, Cambridge, MA, 2007.

[7] T. Hazan and A. Shashua. Norm-product belief propagation: primal-dual message-passing for approximate inference. *IEEE Trans. Inf. Theor.*, 56(12):6294–6316, Dec. 2010.

[8] A. Jaimovich, G. Elidan, H. Margalit, and N. Friedman. Towards an integrated protein-protein interaction network: A relational Markov network approach. *Journal of Computational Biology*, 13(2):145–164, 2006.

[9] J. Johnson. *Convex Relaxation Methods for Graphical Models: Lagrangian and Maximum Entropy Approaches*. PhD thesis, EECS, MIT, 2008.

[10] V. Jojic, S. Gould, and D. Koller. Fast and smooth: Accelerated dual decomposition for MAP inference. In *Proceedings of International Conference on Machine Learning (ICML)*, 2010.

[11] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, 2006.

[12] N. Komodakis and N. Paragios. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *ECCV*, pages 806–820, 2008.

[13] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):531 –552, March 2011.

[14] V. A. Kovalevsky and V. K. Koval. A diffusion algorithm for decreasing energy of max-sum labeling problem. Glushkov Institute of Cybernetics, Kiev, USSR. Unpublished, approx. 1975.

[15] M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0–1 programming. *Math. Oper. Res.*, 28(3):470–496, 2003.

[16] O. Meshi and A. Globerson. An alternating direction method for dual MAP LP relaxation. In *Proceedings of ECML PKDD*, pages 470–483, Berlin, Heidelberg, 2011. Springer-Verlag.

[17] B. Schieber and U. Vishkin. On finding lowest common ancestors: simplification and parallelization. *SIAM J. Comput.*, 17(6):1253–1262, Dec. 1988.

[18] D. Sontag, A. Globerson, and T. Jaakkola. Clusters and coarse partitions in LP relaxations. In *NIPS 21*, pages 1537–1544. MIT Press, 2009.

[19] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2011.

[20] D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *NIPS 20*, pages 1393–1400, Cambridge, MA, 2008. MIT Press.

[21] D. Sontag, T. Meltzer, A. Globerson, Y. Weiss, and T. Jaakkola. Tightening LP relaxations for MAP using message-passing. In *UAI*, pages 503–510. AUAI Press, 2008.

[22] M. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA, 2008.

[23] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(7):1165–1179, 2007.

[24] T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In *CVPR*, 2008.

[25] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – an empirical study. *JMLR*, 7:1887–1907, 2006.

An Approximate Solution Method for Large Risk-Averse Markov Decision Processes

Marek Petrik
IBM Research
Yorktown Heights, NY 10598
mpetrik@us.ibm.com

Dharmashankar Subramanian
IBM Research
Yorktown Heights, NY 10598
dharmash@us.ibm.com

Abstract

Stochastic domains often involve risk-averse decision makers. While recent work has focused on how to model risk in Markov decision processes using risk measures, it has not addressed the problem of solving large risk-averse formulations. In this paper, we propose and analyze a new method for solving large risk-averse MDPs with hybrid continuous-discrete state spaces and continuous action spaces. The proposed method iteratively improves a bound on the value function using a linearity structure of the MDP. We demonstrate the utility and properties of the method on a portfolio optimization problem.

1 Introduction

It is common for decision makers to be risk-averse when uncertainty is involved. For example, risk-averse financial portfolio managers prefer an investment portfolio that mitigates the worst-case plausible losses even if the corresponding expected return is suboptimal. Since the worst-case losses corresponding to solutions that optimize the expected returns can be unacceptable to such decision-makers, there is a need to explicitly model risk-averse objectives. While it is common to use utility functions to capture the decision maker's tolerance of risk, this approach cannot capture many empirically observed risk-averse preferences (e.g. [13]). In addition, decision makers are often unable to provide appropriate utility functions, and non-linear utility functions can complicate the application of dynamic programming [12]. In this paper, we focus on *coherent risk measures*—an alternative model of risk-aversion.

Coherent measures of risk were developed in the field of finance and represent an alternative model of risk-avoidance to utility functions [2, 5]. A coherent risk measure ϱ is a generalization of the risk-neutral expectation operator that maps a random variable X to a real number. Informally, while the risk-neutral expectation computes the average corresponding to a given reference probability distribution, a coherent risk measure takes the worst average assessed over a suitably defined neighborhood around the reference distribution. As such, a coherent risk measure is also interpretable as a conservative way to account for ambiguity in the precise knowledge of the reference distribution. To be a coherent risk measure, the function ϱ must satisfy properties that include convexity and monotonicity, which we describe in detail later.

The properties of coherent risk measures in static settings have been studied extensively and are very well understood [5]. Their application to dynamic settings, in which decisions are taken sequentially as additional stochastic information arrives, is more complicated. Dynamic risk measures have been usually studied in stochastic programming settings which we discuss in Section 6. Recently, dynamic risk measures have been used to formulate risk-averse Markov decision processes [21, 13]. Unfortunately, the methods proposed in these papers only solve small MDPs in which both the states and actions can enumerated. In this paper, we extend the modeling methodology of [21] and [13] to solve structured MDPs with continuous state and action spaces.

To illustrate an MDP with continuous state and action spaces, consider the problem of managing a stock portfolio [11]. In this problem, the decision maker must allocate the investment among a portfolio of stocks in order to maximize the financial gain. The one-step gain of each stock is stochastic, but depends on the current market state; i.e. the distribution of returns is different for volatile and calm markets. The objective is to maximize the expected returns while minimizing the risk of significant losses.

The main components of a Markov decision process are the state and action spaces. For example, the state in

the portfolio management problem is represented by the current position in the stocks and cash, and the state of the market. While the current position may be modeled as a continuous vector, it is common to model the market in terms of a finite set of possible market states [11]. The resulting state space is naturally a hybrid discrete-continuous state space. The actions represent the change in the investments in every period and are also continuous. Because of the continuous aspects of the state and action sets, this problem cannot be solved using the methods proposed in [13]. Large MDPs such as this one are typically solved using approximate dynamic programming, a reinforcement learning method [18]. The method that we propose is related to approximate dynamic programming, but it also exploits the linearity of the domain and the coherent risk measure.

The remainder of the paper is organized as follows. First, Section 2 introduces coherent risk measures and describes their properties in multi-stage optimization. Section 3 then defines risk-averse hybrid linearly controlled MDPs. These problems have both continuous and discrete states, continuous actions, linear transitions, and their objective is formulated in terms of a coherent risk measure. In Section 4, we propose Risk-averse Dual Dynamic Programming (RDDP)—a method to solve hybrid risk-averse MDPs. Our method is related to stochastic dual dynamic programming and point-based methods for solving POMDPs [16]. In Section 5, we evaluate the algorithm on the portfolio management problem and analyze the solution properties. Finally, Section 6 discusses the connections to relevant work on risk-averse objectives in multi-stage stochastic programs.

2 Risk Measures in Markov Decision Processes

In this section, we briefly overview the state of the art in applying risk measures in dynamic settings. We formally define risk-averse Markov decision processes and other necessary notation. We omit some minor technical assumptions needed for continuous state and action sets; refer for example to Sections 4.3 and 4.4 of [19] for the technical details.

A *Markov decision process* is a tuple $(\mathcal{S}, \mathcal{A}, W, P, c)$ defined for time steps $\mathcal{T} = \{0 \dots T\}$ as follows. Let $\mathcal{S}$ be a *compact* state space and $\mathcal{A} \subseteq \mathbb{R}^m$ be a *compact* continuous action space. Let $W : \mathcal{S} \rightrightarrows \mathcal{A}$ denote a measurable set-valued function (or a multimap) that maps each state to the *compact* set of permissible actions and let $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ denote the transition probabilities where $\mathcal{P}(\mathcal{S})$ is the set of all probability measures on $\mathcal{S}$; $P(s, a)$ represents the next-state probability measure for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Let $c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ represent the transition cost; $c(s_t, a_t, s_{t+1})$ represents the cost incurred when transiting from s_t to s_{t+1} upon taking action a_t. Finally, let s_0 denote a deterministic initial state. In the final time-step $T + 1$, we assume that there is no action taken and the problem terminates.

The focus of this paper is on *finite*-horizon total return models. The solution of a finite-horizon MDP is a policy which is composed of decision rules. A *decision rule* $d_t : \mathcal{S} \to \mathcal{A}$ at time t determines the action to be taken in each state. A collection of decision rules, one per each time step, is a deterministic Markov *policy* $\pi : \mathcal{T} \times \mathcal{S} \mapsto \mathcal{A}$. The set of all policies is denoted as Π.

The objective in solving risk-neutral MDPs is to minimize the total cost, or *return*, which is defined for $\pi = (d_1, \dots, d_t)$ as:

$$\varsigma(\pi) = \mathbf{E}_{P_\pi}\left[\sum_{t \in \mathcal{T}} c(S_t, d_t(S_t), S_{t+1})\right], \qquad (1)$$

where S_t is a random variable that represents the state at time t, distributed according to the probability distribution P_π induced by the policy π. The optimal return is $\varsigma^\star = \min_{\pi \in \Pi} \varsigma(\pi)$. It is well-known that there exists an optimal Markov deterministic policy, so it suffices to restrict our attention to this class of policies.

Risk averse MDPs have been formalized recently in [21] and [13]. Informally, given a reference probability distribution for each transition, these formulations modify the objective Eq. (1) by specifically penalizing a subset of the most adverse realizations of the induced cost distribution. In particular, the expectation is replaced by a *dynamically consistent* risk measure ϱ. Risk measures that are dynamically consistent can be applied to multistage optimization problems without violating the dynamic programming principle [22].

Assume that the one-step random MDP costs corresponding to action a_t taken at time step t are represented by $\{Z_{t+1}\}_{t \in \mathcal{T}}$ where $Z_{t+1} \in \mathcal{Z}_{t+1}$, and $\mathcal{Z}_{t+1}$ is the space of random variables adapted to the information filtration at time $t+1$, i.e. the history of the underlying stochastic process at time $t+1$. An ordinary risk measure would assign the objective based on the total sum of the costs $\varrho(\{Z_{t+1}\}_{t \in \mathcal{T}}) = \rho(\sum_{t \in \mathcal{T}} Z_{t+1})$. A dynamically consistent risk measure, on the other hand, is defined as a composition of one-step risk mappings:

$$\varrho(Z_1, \dots, Z_{T+1}) = \rho_0\left(Z_1 + \rho_1\left(Z_2 + \dots + \rho_T\left(Z_{T+1}\right)\right)\right).$$

The one-step conditional risk mappings $\rho_t : \mathcal{Z}_{t+1} \to \mathcal{Z}_t$ must satisfy the following properties:

(A1) *Convexity:* $\rho_{t+1}(\alpha \cdot Z + (1-\alpha) \cdot Z') \leq \alpha \cdot \rho_{t+1}(Z) + (1-\alpha) \cdot \rho_{t+1}(Z')$ for all $Z, Z' \in \mathcal{Z}_{t+1}$ and $\alpha \in [0, 1]$.

(A2) *Monotonicity:* If $Z \preceq Z'$ then $\rho_{t+1}(Z) \preceq \rho_{t+1}(Z')$.

(A3) *Translation equivariance:* If $a \in \mathcal{Z}_t$, then $\rho_{t+1}(Z + a) = \rho_{t+1}(Z) + a$.

(A4) *Positive homogeneity:* If $t > 0$, then $\rho_{t+1}(t \cdot Z) = t \cdot \rho_{t+1}(Z)$.

Note that the value $\rho_t(Z)$ is a random variable in $\mathcal{Z}_t$ and the inequalities between random variables are assumed to hold almost surely.

A general dynamically consistent risk measure may be cumbersome to use due to two reasons. The parameters that specify the particular choice of the mapping ρ_t could in general depend on the entire history until time t, and further, the argument on which the mapping applies, namely, Z_{t+1} could in general depend on the entire history until time $t+1$, therefore requiring non-Markov optimal policies. We, therefore, further restrict our treatment to *Markov risk measures* [21].

Markov risk measures require that the parameters that specify the particular choice of the conditional risk mappings ρ_t are independent of the history, given the current state, and further that the mapping operation is also independent of the history, given the current state. This is a natural choice for MDPs because the source of uncertainty is the stochastic transition corresponding to any given state-action pair (s_t, a_t), and the reference probability distribution for this stochastic transition is independent of the history, given the current state. Intuitively, an input to a conditional risk mapping of a Markov risk measure ρ_t is the random variable of the costs incurred during a transition from a state $s_t \in \mathcal{S}$ and the output of the risk mapping is a single number for each s_t.

The simplest example of a conditional risk mapping is the expectation in which case the objective becomes risk-neutral. The risk neutral objective would be:

$$\varrho(Z_1, \ldots, Z_{T+1}) = \mathbf{E}\left[Z_1 + \mathbf{E}\left[\ldots + \mathbf{E}\left[Z_{T+1}\right]\right]\right].$$

A more general risk measure that allows a convenient specification of the appropriate level of risk tolerance is the Average Value at Risk (AV@R$_\alpha$) parameterized at a chosen level $\alpha > 0$ which is defined as [5]:

$$\text{AV@R}_\alpha(Z) = \max_{q \in \mathcal{Q}_\alpha} \mathbf{E}_q[Z] \ , \tag{2}$$

$$\mathcal{Q}_\alpha = \left\{ q : q(\omega) \leq \frac{p(\omega)}{\alpha}, 0 \leq q(\omega) \leq 1, \omega \in \Omega \right\}.$$

Here, Ω is the *finite* sample space and p is a reference distribution which, in our case, is the distribution induced by the transition probabilities. It is easy to see that AV@R$_1(Z) = \mathbf{E}[Z]$ and AV@R$_0(Z)$ is the worst-case (or robust) realization. In plain words, average value at risk is the conditional expectation beyond the α quantile. Average value at risk, therefore, will assign a higher overall cost to a scenario with heavier tails even if the expected value stays the same. Fig. 1 illustrates how a AV@R is computed in comparison with a plain expectation.

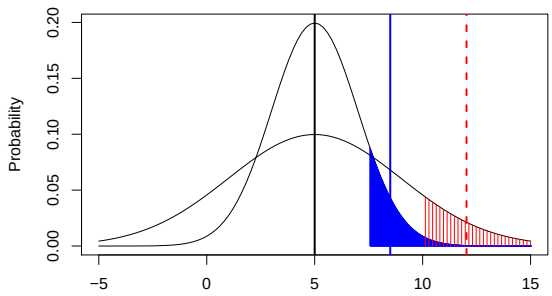


Figure 1: Comparison of two distributions with identical expectations and different AV@R$_{0.1}$ values. The filled-in regions illustrate the quantiles, while the vertical lines indicate the expectations and values at risk.

In the interest of clarity, we concentrate in the remainder of the paper on conditional risk mappings that are a convex combination of average value at risk and expectation and defined as follows:

$$\rho_t(X) = (1-\lambda)\mathbf{E}_p[X] + \lambda\,\text{AV@R}_\alpha(X) \tag{3}$$

We will use the fact that the set $\mathcal{Q}_\alpha$ is polyhedral to derive the algorithms. However, it is well known that any conditional risk mapping that satisfies the assumptions (A1)-(A4) can be represented as:

$$\rho_t(Z) = \sup_{Q \in \mathcal{Q}(s_t, a_t)} \mathbf{E}_Q[Z]$$

where $\mathcal{Q}(s_t, a_t) \subseteq \mathcal{P}(\mathcal{S})$ is a closed bounded convex set of probability measures on $\mathcal{S}$. In addition, when the conditional risk mapping satisfies the *comonotonicity* assumption [5], the set $\mathcal{Q}$ is polyhedral. The proposed algorithm can be, therefore, applied to any dynamically consistent risk measure in which the conditional risk mappings satisfy the comonotonicity assumption.

To derive RDDP, we need yet another representation of AV@R as a minimization linear program. Because the set $\mathcal{Q}_\alpha$ is a polytope, the optimization in Eq. (3) can be written as a linear program. The dual of this linear program yields the following representation:

$$\rho_t(Z) = (1-\lambda)\mathbf{E}_p[Z] + \lambda \begin{bmatrix} \min\limits_{\mu,\xi} & \mu + \dfrac{1}{\alpha} p^\intercal \xi \\ \text{s.t.} & \xi + \mathbf{1}\mu \geq Z \\ & \xi \geq \mathbf{0} \end{bmatrix} \tag{4}$$

The duality we used above for finite spaces can be generalized to more general sample spaces under reasonable technical assumptions [20]. The optimal value of μ corresponds to the quantile of Z at level α.

One attractive property of dynamically consistent risk measures is that they not only model risk aversion, but also preserve most of the structure of MDPs that makes them easy to solve. In particular, there exists an optimal deterministic Markov policy in risk-averse MDPs with Markov risk measures [21], under some mild additional technical assumptions. The additional technical conditions that are needed are the continuity of $P(s_t, \cdot)$ and lower semicontinuity of W and $c(s_t, a_t, s_{t+1})$ with respect to a_t.

It is also possible to define the *value function* in risk-averse MDPs with Markov risk measures. The finite-horizon value function $v_t : \mathcal{S} \to \mathbb{R}$ is defined identically as for the risk neutral case, except the expectations are replaced by a risk measure; that is for some policy π the value function represents the return obtained when starting in a given state. The action-value function $q_\pi : \mathcal{T} \times \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ for a policy π, also known as Q-function [1, 18], represents the value function in a state s *after* taking an action a. The optimal value function is a value function that corresponds to an optimal policy.

The optimal value function in risk-averse MDPs must satisfy the Bellman optimality conditions, which are stated in the following theorem.

Theorem 1 ([21])**.** *A finite-horizon value function $v^\star(t)$ is optimal if and only if it satisfies for all $s \in \mathcal{S}$ and $t \in \mathcal{T}$ the following equality:*

$$v_t^\star(s_t) = \min_{a \in \mathcal{A}} q_t^\star(s_t, a)$$
$$q_t^\star(s_t, a) = \rho_t \left(c_t(s_t, a, S_{t+1}) + v_{t+1}^\star(S_{t+1}) \right).$$

Here, S_{t+1} is a random variable representing the state at $t+1$ distributed according to the transition probability $P(s_t, a)$ and $v_{T+1}(s) = 0$.

We will use RDDP to compute an approximately optimal value function for a risk averse MDP. The actual solution is the greedy policy with respect to this value function. A policy π is *greedy* with respect to a value function q when it is defined for any $s_t \in \mathcal{S}$ at time t as:

$$\pi(s_t) \in \arg\min_{a \in \mathcal{A}} q_t(s_t, a) \ .$$

Using the greedy policy is the most common method of computing a policy from a value function.

This section introduced a very general model for risk-averse Markov decision processes. In the remainder of the paper, Section 3 defines a linear structure of the state and action spaces that is common in some domains, and Section 4 derives RDDP that exploits this structure to efficiently approximate the optimal value function.

3 Hybrid Linearly Controlled Problems

Standard MDP solution methods, such as value or policy iteration, do not scale easily to large or continuous Markov decision processes. Unfortunately, many practical applications involve MDPs with continuously many states and actions. In this section, we describe a specific class of risk averse MDPs with linear transitions and costs that can be solved more efficiently.

Informally, a hybrid linearly controlled problem is an MDP with a state set that has both a discrete and a continuous component. The actions are continuous. The transition probabilities are independent of continuous states and the transitions for continuous states can be described linearly in terms of the continuous states and actions. The following definition describes the problem formally.

Definition 1. *A* hybrid linearly controlled problem *is an MDP $(\mathcal{S}, \mathcal{A}, W, P, c)$ such that:*

- *States $\mathcal{S} = \mathcal{D} \times \mathcal{X}$ where $\mathcal{X} \subseteq \mathbb{R}^n$ is a closed polyhedral set of continuous states and $\mathcal{D}$ is a (small) finite set of discrete states.*
- *Actions $\mathcal{A} \subseteq \mathbb{R}^m$ is a closed convex polyhedral set for some m.*
- *Admissible actions for $(d, x) \in \mathcal{S}$ are restricted by the set-valued map $W : \mathcal{S} \rightrightarrows \mathcal{A}$ defined as:*

  $$W(d, x) = \{a \in \mathcal{A} \ : \ A_d a = b_d - X_d x, l_d \leq a \leq u_d\}.$$

- *Transitions $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ are defined as follows. Assume a* finite *sample space Ω_d for each $d \in \mathcal{D}$ with a given reference probability distribution P_D. Ω_d can be seen as a $d-$specific, one-step finite sample space corresponding to the evolution of an underlying exogenous, stationary, stochastic process. Further, assume the following random variables are specified: $D : \Omega_d \to \mathcal{D}, T_x : \Omega_d \to \mathbb{R}^{n \times n}, T_a : \Omega_d \to \mathbb{R}^{n \times m}, U : \Omega_d \to \mathbb{R}^n$. The mapping D captures the action-independent transition from one discrete state to another in the set $\mathcal{D}$. Now define a random variable:*

  $$X = T_x x + T_a a + U \ .$$

  The mapping X captures the transition from one continuous state to another in the set $\mathcal{X}$ upon taking action a. Both the discrete and continuous state transitions are induced by P_D. In other words, $P((d, x), a)$ is the probability distribution induced by P_D over all (random) pairs (D, X).
- *Cost function $c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a linear function in terms of states and actions:*

  $$c(s, a, s') = c((d, x), a, (d', x')) = c_a^\mathsf{T} a + c_x^\mathsf{T} x + c_n^\mathsf{T} x'.$$

For sake of simplicity, we assume that there exists an admissible action for every state (W is non-empty). This is equivalent to the complete recourse assumption, which is common in stochastic programming.

Dynamic Portfolio Management

We now use a portfolio optimization problem—described in [11]—to illustrate the above model of a hybrid linearly controlled system; later we use this model to experimentally evaluate RDDP. The model assumes three risky assets, cash, and a market state. The one-step returns of the risky assets are denoted by a vector $r_t = (r_t^1, \ldots, r_t^3)$ for any time t. In addition, there is assumed to be a risk-free asset—cash—with a constant and known return r^f. All returns are inflation adjusted. The monetary holdings of the assets at time t are denoted by a vector $x_t = (x_t^1, \ldots, x_t^3)$ and the risk-free cash position is c_t. The decision maker decides on trades that are bought (positive) and sold (negative) on a daily basis, which are denoted by $a_t = (a_t^1, \ldots, a_t^3)$. The transaction costs $\kappa(a_t)$ for a trade are proportional and defined as:

$$\kappa(a_t) = \sum_{i=1}^{3} \delta_i^+ \left[a_t^i\right]_+ - \delta_i^- \left[a_t^i\right]_-$$

for some non-negative δ_i^+ and δ_i^-. The asset holdings x_t and cash position c_t evolve according to:

$$x_{t+1}^i = r_{t+1}^i \cdot (x_t^i + a_t^i) \quad \forall i$$
$$c_{t+1} = r^f \cdot (c_t - \mathbf{1}^\mathsf{T} a_t - \kappa(a_t)) \ .$$

The return rates evolve with the state of the market, as we describe below. The goal of the investor is to maximize a function of the total terminal wealth; it is necessary to not only maximize the expected wealth but also account for the associated risk.

The returns for risky assets evolve according to an exogenous stochastic process that is independent of the investor's position. In particular, the market state is a one-dimensional continuous real variable z_t which partially predicts the returns. The market state and the returns evolve linearly with a jointly normally distributed noise as:

$$\begin{pmatrix} \ln r_{t+1} \\ z_{t+1} \end{pmatrix} = \begin{pmatrix} a_r + b_r z_t \\ a_z + b_z z_t \end{pmatrix} + \begin{pmatrix} e_{t+1} \\ v_{t+1} \end{pmatrix} \ , \qquad (5)$$

where the stochastic variables (e_{t+1}, v_{t+1}) are distributed according to a multivariate normal with zero mean and variance Σ for all times. The values $a_r, b_r, a_z, b_z, \sigma$ were estimated from NYSE data for 1927-1996 [11]. Note that r_{t+1} depends only on z_t and is independent of r_t. We describe the values of these variables in more detail in Appendix A.

The one-dimensional market state variable and the market rates transitions are discretized [11, 3]. The market state variable is discretized uniformly to 19 points. The distributions for market rates are then estimated for the discrete grid using Gaussian quadrature methods with 3 points per each dimension (27 total) to precisely match the mean and the variance [10].

The dynamic portfolio optimization problem can then be naturally modeled as a hybrid linearly controlled problem. The state space $\mathcal{S} = (\mathcal{D}, \mathcal{X})$ is factored into discrete states $\mathcal{D} = \{1 \ldots 19\}$ that represent the market state z and continuous states $\mathcal{X} \subset \mathbb{R}^4$ that represent the asset investment and the monetary position. The first three elements of $x \in \mathcal{X}$ represent the asset positions and the last element represents the cash.

The set of actions $\mathcal{A}$ represents the feasible trades $a_t \in \mathbb{R}^3$ for each of the three assets. The cash position is adjusted accordingly, depending on the balance of the trades and transaction costs. The actions are constrained by $W(s)$ in order for the asset and cash positions to remain non-negative as follows:

$$W(d,x) = \left\{ a \quad : \quad \begin{array}{c} x(4) - \mathbf{1}^\mathsf{T} a - \kappa(a) \geq 0 \\ x(i) + a(i) \geq 0 \quad i = 1 \ldots 3 \end{array} \right\} \ .$$

The costs represent the total change in wealth, which is a function of the capital gains and the transaction costs. Formally, for any $(d,x) \in \mathcal{S}$, $a \in \mathcal{A}$, $(d',x') \in \mathcal{S}$ the cost is the reduction in total wealth:

$$c((d,x), a, (d',x')) = \mathbf{1}^\mathsf{T}(x - x') \ .$$

The evolution of the dynamic portfolio optimization is depicted in Fig. 2. The round nodes indicate states, while the square nodes indicate intermediate state-action positions. The solid arrows represent the optimization decisions to rebalance the portfolio and the dashed arrows represent stochastic transitions that determine the next market state and the returns. The figure also indicates at which point the risk mappings are applied.

4 Risk Averse Dual Dynamic Programming

In this section, we describe a new approximate algorithm to solve hybrid linearly controlled problems. This algorithm is loosely based on value iteration but uses the polyhedral structure of the MDP to simultaneously update value functions over a range of states. The method is approximate because it uses simulation to identify the relevant states.

The Bellman optimality equations, as described in Theorem 1, can be easily adapted for the hybrid linearly controlled problem as follows.

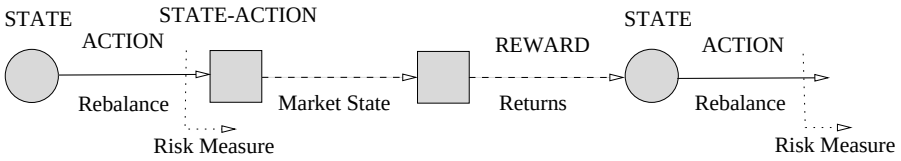


Figure 2: Transitions in the portfolio optimization problem.

Corollary 1 (Bellman optimality condition)**.** *A finite-horizon value function $v^\star(t)$ is optimal if and only if it satisfies for all $s \in \mathcal{S}$ and $t \in \mathcal{T}$ the following equality:*

$$v_t^\star(s_t) = \min_{a \in W(s_t)} q_t^\star(s_t, a) \tag{6}$$

$$q_t^\star(s_t, a) = \rho_t \left(c(s_t, a, S_{t+1}) + v_{t+1}^\star(S_{t+1}) \right) \tag{7}$$

Here, S_{t+1} is a random variable representing the state at time $t+1$ distributed according to the transition probability $P(s_t, a)$ and $v_{T+1}(s_{T+1}) = 0$.

Assume a state $s = (d, x)$ and consider D and X to be random variables representing the next state as in Definition 1. Then Eq. (7), which is in particular a composition of a risk measure with an affine map, can be written as:

$$\begin{aligned} q_t^\star(s,a) &= q_t^\star((d,x),a) \\ &= \rho_t \left(c_a^\mathsf{T} a + c_x^\mathsf{T} x + c_n^\mathsf{T} X + v_{t+1}(D, X) \right) \\ &= \rho_t \big(c_a^\mathsf{T} a + c_x^\mathsf{T} x + c_n^\mathsf{T} T_x x + c_n^\mathsf{T} T_a a + c_n^\mathsf{T} U + \\ &\quad + v_{t+1}(D, T_x x + T_a a + U) \big) \\ &= \rho_t \left(Z_{t+1}(d, x, a) \right) \ , \end{aligned}$$

for the finite vector-valued (random) variable $Z_{t+1}(d, x, a) = c_a^\mathsf{T} a + c_x^\mathsf{T} x + c_n^\mathsf{T} T_x x + c_n^\mathsf{T} T_a a + c_n^\mathsf{T} U + v_{t+1}(D, T_x x + T_a a + U)$, which has dimension $|\Omega_d|$.

To derive RDDP, we now show that the Bellman optimality conditions for linearly controlled hybrid problems can be formulated as a linear program. The optimization variables in the problem correspond to actions and the risk measure. The constraints are a function of the state. Using Eq. (4), it can be readily shown that Eq. (7) a linear program. This representation can then be coupled with the optimization over the action a in Eq. (6) to get the following linear program representation of the Bellman optimality conditions:

$$v_t^\star(d, x) = \left[\begin{array}{ll} \min\limits_{a,\mu,\xi} & (1-\lambda) p_d^\mathsf{T} Z_{t+1}(d, x, a) + \lambda \big(\mu + \frac{p_d^\mathsf{T} \xi}{\alpha} \big) \\ \text{s.t.} & A_d a = b_d - B_d x \\ & l_d \le a \le u_d \\ & \xi + \mathbf{1}\mu \ge Z_{t+1}(d, x, a) \\ & \xi \ge \mathbf{0} \end{array} \right], \tag{8}$$

where p_d is the measure P_D for $d \in \mathcal{D}$. The notation above assumes that random variables are represented as vectors as the following equality illustrates:

$$p_d^\mathsf{T} Z_{t+1}(d, x, a) = \sum_{\omega \in \Omega_d} p_d(\omega) Z_{t+1}(d, x, a, \omega) \ ,$$

where: Ω_d is the *finite* sample space and $Z_{t+1}(d, x, a, \omega) = c_a^\mathsf{T} a + c_x^\mathsf{T} x + c_n^\mathsf{T} T_x(\omega) x + c_n^\mathsf{T} T_a(\omega) a + c_n^\mathsf{T} U(\omega) + v_{t+1}^\star(D(\omega), T_x(\omega) x + T_a(\omega) a + U(\omega))$.

Now, using Eq. (8), it is easy to show the following proposition that describes the structure of value functions.

Proposition 1. *Assuming that $W(s)$ is non-empty for all $s \in \mathcal{S}$, the optimal value function is piecewise affine convex function of x for each $d \in \mathcal{D}$:*

$$v_t^\star(d, x) = \max_{i \in \mathcal{I}_t(d)} \ (q_{x,i}^\mathsf{T} x + q_{c,i}) \ , \tag{9}$$

where $\mathcal{I}_t(d)$ is a finite non-empty set. In addition, the state-action value function $q_t^\star(s, a)$ is piecewise affine convex in a for any $s \in \mathcal{S}$.

Proof. We prove the proposition by a backward induction on t. The proposition holds trivially for v_{T+1} (since $v_{T+1} = 0$). Consider $t = T$. The linear program in Eq. (8) may be simplified as,

$$v_T^\star(d, x) = \left[\begin{array}{ll} \min\limits_{a,\mu,\xi,z} & \lambda \cdot \mu + \sum\limits_{\omega \in \Omega_d} p_d(\omega) \Big((1-\lambda) \cdot z(\omega) + \\ & \quad + \frac{\lambda}{\alpha} \cdot \xi(\omega) \Big) \\ \text{s.t.} & \delta|_1 : \ A_d a = b_d - B_d x \\ & \delta|_2 : \ z(\omega) - (c_a^\mathsf{T} + c_n^\mathsf{T} T_a(\omega)) a = \\ & \quad = (c_x^\mathsf{T} + c_n^\mathsf{T} T_x(\omega)) x + c_n^\mathsf{T} U(\omega) \\ & \delta|_3 : \ l_d \le a, \quad \delta|_4 : \ a \le u_d \\ & \xi(\omega) + \mu - z(\omega) \ge 0, \quad \xi \ge \mathbf{0} \ . \end{array} \right] \tag{10}$$

Due to the assumption of a non-empty and bounded $W(s)$ for all $s \in \mathcal{S}$, the set of extreme points corresponding to the dual polytope of the above linear program is a non-empty, finite set that is independent of x, and depends only on d. Let $\mathcal{I}_t(d)$ denote this finite set. By duality, we have

$$\begin{aligned} v_T^\star(d, x) = \max_{i \in \mathcal{I}_t(d)} \ & \Big(\delta_i|_1^\mathsf{T} (b_d - B_d x) \\ & + \sum_{\omega \in \Omega_d} \delta_i|_2(\omega) \big((c_x^\mathsf{T} + c_n^\mathsf{T} T_x(\omega)) x + c_n^\mathsf{T} U(\omega) \big) . \\ & + \delta_i|_3^\mathsf{T} l_d + \delta_i|_4^\mathsf{T} u_d \Big) \end{aligned}$$

The above expression is indeed a piecewise affine convex function of x, where we may readily identify $q_{x,i}$ and $q_{c,i}$ by algebraically grouping terms that are linear in x and the additive constant, for each i. Thus the proposition is true for $t = T$. Assuming that it is true for any $t+1 \leq T$, we can show next that it is true for t as well.

Now, for any $t < T$, the mathematical optimization in Eq. (8) with the representation of $v^\star_{t+1}$ as in Eq. (9) is the following linear program.

$$
\begin{aligned}
& v^\star_t(d,x) = \\
& \left[\begin{array}{ll}
\min\limits_{\substack{a,\mu,\\ \xi,z,y}} & \lambda\cdot\mu + \sum\limits_{\omega\in\Omega_d} p_d(\omega)\Big((1-\lambda)\cdot z(\omega) + \\
& \qquad + \dfrac{\lambda}{\alpha}\cdot\xi(\omega)\Big) \\
\text{s.t.} & \delta|_1:\ A_d a = b_d - B_d x \\
& \delta|_2:\ z(\omega) - (c_a^\mathsf{T} + c_n^\mathsf{T} T_a(\omega))a = \\
& \quad = (c_x^\mathsf{T} + c_n^\mathsf{T} T_x(\omega))x + c_n^\mathsf{T} U(\omega) + y(\omega) \\
& \delta|_3:\ l_d \leq a, \quad \delta|_4:\ a \leq u_d \\
& \delta|_5:\ y(\omega) - q_{x,j}^\mathsf{T} T_a(\omega) a \geq q_{x,j}^\mathsf{T}\cdot \\
& \quad \cdot\big(T_x(\omega)x + U(\omega)\big) + q_{c,j} \\
& \qquad \forall j \in \mathcal{I}_{t+1}(D(\omega)) \\
& \xi(\omega) + \mu - z(\omega) \geq 0, \quad \xi \geq \mathbf{0}\ .
\end{array}\right]
\end{aligned}
\tag{11}
$$

The constraint family indexed by $\delta|_5$ is a consequence of the induction hypothesis, where $y(\omega)$ is an auxiliary variable that captures $v^\star_{t+1}(D(\omega), X(\omega))$. A similar argument using duality for Eq. (11) gives:

$$
\begin{aligned}
v^\star_t(d,x) = \max_{i\in\mathcal{I}_t(d)} \Big(& \delta_i|_1^\mathsf{T}(b_d - B_d x) + \\
& + \sum_{\omega\in\Omega_d} \delta_i|_2(\omega)\big((c_x^\mathsf{T} + c_n^\mathsf{T} T_x(\omega))x + c_n^\mathsf{T} U(\omega)\big) + \\
& + \delta_i|_3^\mathsf{T} l_d + \delta_i|_4^\mathsf{T} u_d + \\
& + \sum_{\substack{\omega\in\Omega_d,\\ j\in\mathcal{I}_{t+1}(D(\omega))}} \delta_i|_5(\omega,j)\cdot\big(q_{x,j}^\mathsf{T}(T_x(\omega)x + U(\omega)) + \\
& + q_{c,j}\big)\Big),
\end{aligned}
\tag{12}
$$

thereby giving a piecewise affine convex function of x for $v^\star_t(d,x)$. □

Note that the convex representation in Eq. (9) has an exponentially large number of pieces in general, and is not usable in its exact form. The algorithm we propose, RDDP, takes advantage of the piecewise linear representation of the optimal value function to approximate it efficiently. This method is related to value iteration; it iteratively grows an approximation of the large set $\mathcal{I}_t(d)$ by adding one element in each step. Fortunately, it is not necessary to compute the optimal value function to get a good policy—a good value function will suffice. We propose an algorithm that approximates the value function from below by using only a subset of $\mathcal{I}_t(d)$. The algorithm is summarized in Algorithm 1.

In particular, we identify the relevant linear pieces of each value function by successively refining *lower bounds* on the value function. Because the representation as shown in Eq. (9) is convex, a subgradient inequality of the function may be readily derived at any (d,x) which will serve as a lower bound. The following proposition summarizes this property.

Proposition 2. *Consider $v^\star_t(d,x)$ described as in Proposition 1, and let $\hat{x} \in \mathcal{X}$ be any fixed continuous component of the hybrid state. For any chosen value d, let $\delta_{\hat{i}}$ be the corresponding optimal dual solution of Eq. (11), where $\hat{i} \in \mathcal{I}_t(d)$, and $f^\star(\hat{x})$ be the corresponding optimal objective value. Then: $e_x^\mathsf{T} x + e_c \leq v^\star_t(d,x)$ where:*

$$
\begin{aligned}
e_x^\mathsf{T} &= -\delta_{\hat{i}}^\star|_1^\mathsf{T} X_d + \sum_{\omega\in\Omega_d}\big(\delta_{\hat{i}}^\star|_2(\omega) c_n^\mathsf{T} T_x(\omega)\big) \\
&\quad + \sum_{\substack{\omega\in\Omega_d,\\ j\in\mathcal{I}_{t+1}(D(\omega))}} \big(\delta_{\hat{i}}^\star|_5(\omega,j) q_{x,j}^\mathsf{T} T_x(\omega)\big) \\
e_c &= f^\star(\hat{x})\ .
\end{aligned}
$$

In addition, with such lower bounds added in Algorithm 1, we are guaranteed to converge to the optimal value function.

Proof. The lower bounding inequality for $v^\star_t(d,x)$ follows directly from the subgradient of the convex value function as established in Proposition 1. It is a direct consequence of Eq. (12), since $\hat{i} \in \mathcal{I}_t(d)$. In particular, starting with an empty set $\mathcal{J}_t(d)$ as our approximation for $\mathcal{I}_t(d)$, at each time step we iteratively improve $\mathcal{J}_t(d)$ by adding an additional element $\hat{i}$. The convergence to the optimal solution follows because the scenario tree that represents all T-step realizations of the uncertainty is finite. Then, using backward induction, finite termination of the algorithm can be readily shown. □

5 Empirical Results

In this section, we present numerical results of RDDP on the portfolio optimization domain described in Section 3. The results not only demonstrate the utility of RDDP in modeling and solving risk-averse MDPs but also illustrate the properties of risk-averse behavior in portfolio optimization. Please note that it is impractical to solve the portfolio optimization problem using discretization since the state-actions space is 8-dimensional. Even with a moderate discretization with 8 points per each dimension, computing the value of the risk measure Eq. (2) for all states and actions

Algorithm 1: Risk-sensitive Dual Dynamic Programming (RDDP)

Data: MDP Model
Result: Lower bound on value function: $\underline{v}_t^\star$
$\mathcal{J}_t(d) \leftarrow \{\}$ $\forall d \in \mathcal{D}$ $\forall t$;
while *iterations remain* **do**
  `// Forward pass: sample states`
  $s_0 = (d_0, x_0) \leftarrow$ initial state ;
  for $t \in 0 \dots T$ **do**
    $a_t \leftarrow$ Solve LP Eq. (11) with $\mathcal{J}_{t+1}(d)$;
    $\omega_{t+1} \leftarrow$ sample from Ω_{d_t}.
    $x_{t+1} \leftarrow T_x(\omega_{t+1})x_t + T_a(\omega_{t+1})a_t + U(\omega_{t+1})$;
    $d_{t+1} \leftarrow D(\omega_{t+1})$;
  `// Backward pass: compute lower bounds`
  for $t \in T \dots 1$ **do**
    for $d' \in \mathcal{D}$ **do**
      Solve LP Eq. (11) for (d', x_t) ;
      Compute $\hat{i}$, $q_{x,\hat{i}} = e_x$, $q_{c,\hat{i}} = e_c$ from Proposition 2 ;
      $\mathcal{J}_t(d') \leftarrow \mathcal{J}_t(d') \cup \{\hat{i}\}$;
  Update lower bound for initial state. Solve LP Eq. (11) for (d_0, x_0) ;

involves a linear program with $8^8 * 19 * 27 \approx 9 \cdot 10^9$ variables; here 19 and 27 represent the number of transitions.

We start by evaluating RDDP convergence in a risk-neutral setting; it is easy to evaluate the quality of such a policy by simulation. The policy computed by RDDP for this objective is simple: it always invests the full wealth in the small-cap stock which has the highest expected value. As Proposition 2 shows, RDDP updates a guaranteed lower bound on the optimal value function in each iteration. Because this is a minimization problem, a lower bound on the value function corresponds to an upper bound on the total wealth. Fig. 3 shows the convergence of the lower bound for the consecutive iterations of RDDP. The figure compares the upper bound with simulated returns for 5 time steps and 3000 runs for each policy. The results show that the bound converges close to the optimal solution in 10 iterations. Each iteration involves solving 100 linear programs in Eq. (11).

The opposite of the risk-neutral formulation is the worst-case—or robust—formulation with $\lambda = 1$ and $\alpha = 0$. The robust formulation essentially corresponds to replacing the expectation over the outcomes by the worst case value. The optimal solution in this setting is to invest the entire wealth in cash, which has a constant return and RDDP converges to it in a single iteration. While the worst-case solution is easy to compute, it is hard to evaluate its value by simulation. This is because one needs to sample all realizations to find the worst one.

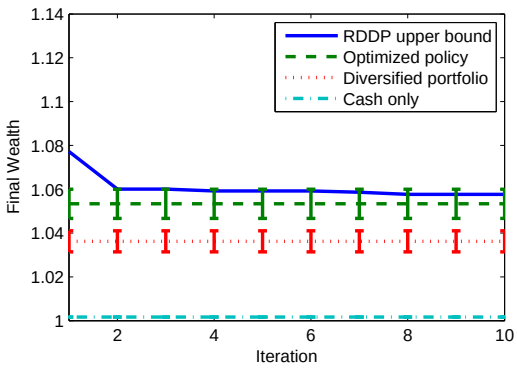


Figure 3: Simulated returns and an upper bound computed by RDDP. The confidence bounds correspond to 2 standard deviations.

Finally, we evaluate a mildly risk-averse solution with $\lambda = 0.2$ and $\alpha = 0.7$. Given this objective, the decision maker optimizes the expected return with a weight of 80% and the 70% tail expectation with a weight of 20%. The solution computed for this risk-averse objective invests in a mix of cash and the small-cap stock that depends on the market state. There is no investment in large or medium-cap stock.

Fig. 4 compares the investment value for risk neutral and averse solutions. The investment for the risk-averse solution is 90% in cash and 10% in small-cap stop; the fractions are reversed for the risk-neutral solution. The value of the risk-neutral solution increases with market state because the expected return of stocks increases with higher volatility. The value of the risk averse solution increases for calm market states (1-5) but decreases with higher market volatility as the portfolio mix becomes more biased towards cash. The value function the decreases sharply when the volatility is high because of trading fees charged when rebalancing towards cash.

The risk-averse solution achieves an average return of about 0.6%, which is much lower than 5.4% for the risk neutral solution. However, the variance of the risk-averse solution is close to 0 and the return is about 5 times greater than the return of pure cash investment 0.1%. RDDP also efficiently optimizes the dynamically consistent risk measure reducing the lower bound by a factor of 26.9 in the first three iterations.

It is interesting to compare the RDDP solution to existing methods for solving portfolio optimization problems. The risk aversion in previous work was modeled by a concave utility function and the solutions were heuristics based on frictionless models [11, 3].

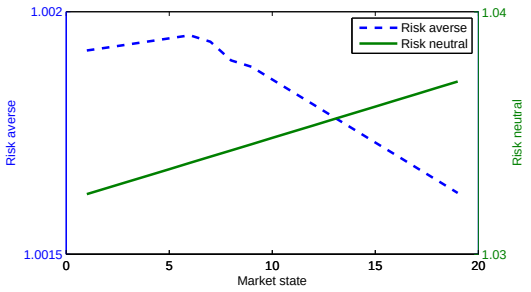


Figure 4: Value of investment as a function of the market state (negated value function). Market volatility increases with increasing state.

The utility-based risk aversion, coupled with the assumption of no transaction fees, led to diversification of the portfolio between middle and large capitalization stocks. On the other hand, the RDDP solution with the AV@R risk measure prefers diversification into cash to other stocks. Future work will need to address whether this difference is due to different risk objectives or different approximation techniques used to compute the solutions.

6 Related Work and Discussion

In this section, we discuss connections between RDDP and related work in both stochastic programming and artificial intelligence communities. Large or continuous risk-neutral MDPs, such as the hybrid linearly controlled problems, are often solved by approximate dynamic programming or reinforcement learning. Approximate dynamic programming overcomes the large size of these problems by restricting value functions to a small linear space [18]. We are not aware, however, of any approximate dynamic programming methods that optimize risk-averse objectives. RDDP is similar to approximate dynamic programming, except the features are constructed automatically using the special structure of the problem. As a result, RDDP is easier to apply to compatible problems, but is less general because it cannot be applied to non-linear problems.

RDDP is also related to stochastic dual dynamic programming (SDDP). SDDP is an approximate method for solving large multistage linear programs with certain independence assumptions [14, 15, 22].Some of these methods have been extended to risk averse objectives [9, 6, 17, 7]; although they typically assume a different independence structure from the proposed linearly controlled problems.

In a parallel stream of work on risk-averse optimization, models with polyhedral risk measures have been studied [4, 8]. Unlike the Markov risk measures, polyhedral risk measures are not dynamically consistent, but must satisfy other properties that make the resulting problems easy to solve.

	Large-cap	Mid-cap	Small-cap	Cash
Mean (a_r, a_z)	0.0053	0.0067	0.0072	0.0000
Coefficient (b_r,b_z)	0.0028	0.0049	0.0062	0.9700

Table 1: Regression coefficients of rate means

	Large-cap	Mid-cap	Small-cap	Cash
Large-cap	0.002894	0.003532	0.003910	-0.000115
Mid-cap		0.004886	0.005712	-0.000144
Small-cap			0.007259	-0.000163
Dividend Yield				0.052900

Table 2: Noise covariance Σ_{ev}

7 Conclusion

The paper describes RDDP, a new algorithm for solving Markov decision processes with risk averse objectives and continuous states and actions. We model the risk-aversion using dynamically consistent convex risk measures, which is an established approach in stochastic finance. Our experimental results on a portfolio optimization problem indicate that RDDP can be converge quickly to a close-to-optimal solution for both risk-averse and risk-neutral objectives. Our risk averse solutions also provide new insights into the properties of risk-aversion in the portfolio optimization setting.

While the focus of this paper is on a financial application, there are numerous other domains that exhibit similar linear properties, such as stochastic inventory management, or hydrothermal energy management. The portfolio optimization problem represents a good benchmark, because it is relatively simple to describe and fit to real data, and a similar structure can be found in many domains that involve optimal utilization of finite resources.

One significant weakness of RDDP is that the convergence criterion is not well defined and risk-averse objectives may be hard to evaluate by simulation. It is not hard, however, to remedy both these issues. Using the convexity of the optimal value function and the lower bound computed by RDDP, it is also possible to compute an upper bound on the value function based on Jensen's inequality. The difference between the upper and lower bounds then provides a reliable stopping criterion and an empirical value of a policy.

A Numerical Values

The values $a_r, b_r, a_z, b_z, \sigma$ were estimated from NYSE data for 1927-1996 [11]. Table 1 summarizes the regression coefficients a_r, b_r, a_z, b_z and Table 2 summarizes the covariance of the noise σ. The risk-free return on cash is $r_f = 1.00042$. The initial market state is $z_0 = 0$.

References

[1] Andrew Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[2] Aharon Ben-Tal and Marc Teboulle. An old-new concept of convex risk measures: The optimized certainty equivalent. *Mathematical Finance*, 17:449–476, 2007.

[3] David B. Brown and J. Smith. Dynamic portfolio optimization with transaction costs: Heuristics and dual bounds. *Management Science*, 57(10):1752–1770, 2011.

[4] A. Eichorn and W. Romisch. Polyhedral measures of risk. *SIAM Journal on Optimization*, 16:69–95, 2005.

[5] Hans Follmer and Alexander Schied. *Stochastic Finance: An introduction in discrete time*. Walter de Gruyter, 3rd edition edition, 2011.

[6] Vincent Guigues. Value of rolling horizon policies for risk-averse hydro-thermal planning. Technical report, Optimization Online, 2010.

[7] Vincent Guigues. SDDP for some interstage dependent risk averse problems and application to hydro-thermal planning. Technical report, Optimization Online, 2011.

[8] Vincent Guigues and Werner Romisch. Sampling-based decomposition methods for risk-averse multistage stochastic programs. Technical report, Optimization Online, 2011.

[9] Vincent Guigues and Claudia Sagastizabal. Risk averse feasible policies for stochastic linear programming. Technical report, Optimization Online, 2009.

[10] K. L. Judd. *Numerical methods in Economics*. MIT Press, 1998.

[11] A. W. Lynch. Portfolio choice and equity characteristics: Characterizing the hedging demands induced by return predictability. *Journal of Financial Economics*, 62:67–130, 2001.

[12] Janusz Marecki and Pradeep Varakantham. Risk-sensitive planning in partially observable environments. In *Conference on Autonomous Agents and Multiagent Systems*, 2010.

[13] Takayuki Osogami. Iterated risk measures for risk-sensitive Markov decision processes with discounted. In *Uncertainty in Artificial Intelligence*, 2011.

[14] M. V. F. Pereira. Stochastic optimization of a multi-reservoir hydroelectric system: A decomposition approach. *Water Resource Research*, 21:779–792, 1985.

[15] M. V. F. Pereira. Stochastic operation scheduling of large hydroelectric systems. *Electric Power and Energy Systems*, 11(3):161–169, 1989.

[16] Marek Petrik and Shlomo Zilberstein. Linear dynamic programs for resource management. In *Conference on Artificial Intelligence (AAAI)*, 2011.

[17] A. Philpott and V. de Matos. Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. Technical report, Optimization Online, 2010.

[18] Warren B. Powell. *Approximate Dynamic Programming*. Wiley-Interscience, 2007.

[19] Martin L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 2005.

[20] R.T. Rockafellar and S. Uryasev. Conditional Value-At-Risk for general loss distributions. *Journal of Banking and Finance*, 26(7):1443–1471, 2002.

[21] Andrzej Ruszczynski. Risk-averse dynamic programming for Markov decision processes. *Mathematical Programming B*, 125(2):235–261, 2010.

[22] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lecture Notes on Stochastic Programming: Modeling and Theory*. SIAM, 2009.

Probability and Asset Updating using Bayesian Networks for Combinatorial Prediction Markets

Wei Sun
Center of Excellence in C4I
George Mason University
Fairfax, VA 22030

Robin Hanson
Department of Economics
George Mason University
Fairfax, VA 22030

Kathryn B. Laskey
Department of Systems Engineering and Operations Research
George Mason University
Fairfax, VA 22030

Charles Twardy
Center of Excellence in C4I
George Mason University
Fairfax, VA 22030

Abstract

A market-maker-based prediction market lets forecasters aggregate information by editing a consensus probability distribution either directly or by trading securities that pay off contingent on an event of interest. Combinatorial prediction markets allow trading on any event that can be specified as a combination of a base set of events. However, explicitly representing the full joint distribution is infeasible for markets with more than a few base events. A factored representation such as a Bayesian network (BN) can achieve tractable computation for problems with many related variables. Standard BN inference algorithms, such as the junction tree algorithm, can be used to update a representation of the entire joint distribution given a change to any local conditional probability. However, in order to let traders reuse assets from prior trades while never allowing assets to become negative, a BN based prediction market also needs to update a representation of each user's assets and find the conditional state in which a user has minimum assets. Users also find it useful to see their expected assets given an edit outcome. We show how to generalize the junction tree algorithm to perform all these computations.

1 INTRODUCTION

Prediction is a fundamental task for AI systems. There is strong theoretical and empirical support for the superiority of ensemble forecasts over individual forecasts (Solomonoff, 1978). While weighted forecasts are theoretically optimal, it has been surprisingly difficult to beat a simple unweighted average. Prediction markets have emerged as a simple and robust way to give high-performing forecasters more influence on the aggregated forecast. Not only do prediction markets typically out-perform both individual human forecasters and naïve unweighted averages (c.f., Chen and Pennock 2010), they show promise as an information combination method for machine aggregation as well. Barbu and Lay (2011) show that combining machine learners with prediction markets often outperforms top ensemble predictors like Random Forest on UCI datasets. A key advantage is that "the market mechanism allows the aggregation of specialized classifiers that participate only on specific instances." That is, learners can opt to bid only on cases they understand. So while most prediction markets use only human traders, they need not. Indeed, smart traders use algorithms, and Nagar and Malone (2011) provide strong evidence that human-machine combinations outperform either: machines do well when the rules hold, while humans recognize "broken leg" situations where they don't.

We focus on market-scoring-rule systems which create prediction markets from the sequential application of proper scoring rules, sidestepping impossibility theorems that apply to simultaneous aggregation of forecasts (Hanson, 2003; Chen and Pennock, 2010). Independent of whether traders are human and/or machine, we address the question of how to make the market itself more expressive by allowing combinatorial trades. While traditional prediction markets ignore dependencies among events, combinatorial prediction markets explicitly consider and exploit dependencies among base events. As Chen and Pennock (2010) argue:

> Why do we need or want combinatorial-outcome markets? Simply put, they allow for the collection of more information. Combinatorial outcomes allow traders to assess the correlations among base objects, not just their independent likelihoods, for example the correlation between Democrats winning

in Ohio and Pennsylvania.

We prove some new results regarding combinatorial prediction markets and report on an implementation using Bayesian networks.

1.1 PREDICTION MARKETS

Prediction markets make probabilistic forecasts by allowing participants to trade contingent assets. Prices in such a market can be interpreted as probabilities: if an asset paying \$1 contingent on event E is currently selling for \$0.75, then the current market probability of E is 75%. Prediction markets are an increasingly popular way to aggregate information and judgments from groups (Tziralis and Tatsiopoulos, 2007). Traders self-select to speak on the topics they think they know best. Those with more knowledge achieve greater influence by acquiring more assets, and market prices inform everyone of trader information.

In a market-maker-based prediction market, an automated trader stands ready to buy or sell assets on any relevant event. The prices it offers can be seen as a current trader consensus on the probabilities of those events, and trades can be seen as edits of consensus probabilities. In a logarithmic market scoring rule based (LMSR-based) prediction market, the market maker varies its price exponentially with the quantity of assets it sells. Tiny trades are fair bets at the consensus probabilities (Hanson, 2003). Larger trades change the consensus probabilities; we call such trades "edits." Users make edits in an attempt to maximize assets, thereby forming a consensus distribution that aggregates information from all market participants.

1.2 COMBINATORIAL PREDICTION MARKETS

In a combinatorial prediction market, one can trade on any event that can be specified as a combination (e.g. 'and' or 'or') of a base set of events. A market-maker-based combinatorial prediction market, therefore, declares a complete consistent probability distribution over a combinatorial space of events, and lets participants edit any part of that distribution. In a combinatorial LMSR-based market, users can make conditional bets that satisfy intuitive independence properties. For example, letting "$\neg$" denote "not", a trader who increases the value of $p(A|B)$ gains if B and A occur and loses if B and $\neg A$ occur. Such an edit changes neither $p(B)$ nor $p(A|\neg B)$ and the trader neither gains nor loses if $\neg B$ occurs (Hanson, 2007).

With a large set of base events, the number of event combinations becomes astronomical, making it intractable in general to compute market prices and trades. In particular, it is in general NP-hard to maintain correct LMSR prices across an exponentially large outcome space (Chen et al., 2008a).

One way to achieve tractability is to limit the complexity of the consensus probability distribution by using a factored representation of the joint distribution. Examples include Bayesian networks (BNs) and Markov networks, which admit standard algorithms to efficiently compute conditional marginals and perform evidential updating (e.g. Pearl 1988; Jensen 1996; Shachter et al., 1990). Graphical models are widely used in many applications, often achieving tractable inference in networks with thousands of variables.

Bayesian networks have been used to represent joint distributions in prediction markets. Chen et al. (2008b) used a BN to represent prices in a tournament, and Pennock and Xia (2011) used a BN to represent probabilities in a LMSR-based combinatorial prediction market.

Pennock and Xia (2011) proved that probabilities can be updated in polynomial time for edits which do not violate the conditional independence assumptions of a decomposable BN of fixed treewidth. However, they do not show how to accomplish two other tasks that are important in practical prediction markets. They do not show how to let traders reuse assets purchased in previous trades to pay for new trades. They also do not show how to calculate a trader's expected assets to see if the trader is already 'long' or 'short' on an issue before making an edit.

1.3 REUSING ASSETS

In the LMSR framework of Pennock and Xia (2011), each edit takes the form of a participant paying cash to a market maker to obtain an event contingent asset, which pays cash if a certain event happens. A trader who has run out of cash is not permitted to make any more trades. Yet the assets one has obtained from prior trades are often sufficient to guarantee many more trades. For example, suppose a user buys an asset "Pays \$10 if A," and then later buys an asset "Pays \$10 if $\neg A$." Because one of these assets is guaranteed to pay off, the two assets are together worth \$10 in cash, and could be used to support future purchases. However, in a naïve implementation, this \$10 is unnecessarily tied up until the truth-value of A is resolved. While we might imagine that a system could easily notice the guaranteed payoff and trade those assets in for cash, it would be difficult to notice more complex combinations of trades worth a guaranteed amount of cash.

In a market that allows conditional trades, a consensus probability $p(A|B) = x$ corresponds to a market

price of $\$x$ for a trade that pays \$1 if events B and A both occur, pays nothing if B and $\neg A$ both occur, and is called off (returning the purchase price to the user) if $\neg B$ occurs. Although Pennock and Xia (2011) do not use conditional securities, conditional probabilities are established by trading securities that depend on joint states. A consensus probability $p(A|B) = x$ corresponds to a market price of $\$x$ to purchase two separate securities, one paying \$1 if B and A both occur and the other paying $\$x$ if $\neg B$ occurs.

Now, suppose a user wants to trade on event A given N mutually exclusive conditions B_i, where the current market probabilities are $p(A|B_i) = x_i, i = 1 \ldots N$. Without asset reuse, such a user would have to purchase N separate pairs of assets, where the i^{th} pair costs $\$x_i$ and pays \$1 if A and B_i occur, 0 if $\neg A$ and B_i occur, and x_i if $\neg B_i$ occurs. The total purchase price of $\sum x_i$ would be tied up until one of the B_i occurred, although the collection of assets is guaranteed to pay off at least $\sum \$x_i - \max\{\$x_i\}$. If N is large and the probabilities are non-negligible, a considerable sum could be unnecessarily tied up.

1.4 OUR CONTRIBUTIONS

In order to be able to reuse assets, we first need to represent the user's assets in a form that allows efficient computation to find the minimum asset state after the user's edits. Further, a factored representation of assets will provide significant savings in space and improve efficiency of asset management. In this paper, we show how to exploit the junction tree to efficiently maintain a representation of a trader's state-dependent assets, i.e., for each state the final cash this trader would hold if this state were revealed in the end to be the actual state. We also show how to use these data structures to efficiently find the largest amount by which the user could raise or lower the probability of an event of interest, before the change might result in the trader holding negative assets in some state. Keeping edits within these limits ensures that traders can reuse assets while never 'going broke.' Asset reuse enables more efficient information aggregation (i.e., users can make more trades before running out of assets) for a given amount of assets.

A trader about to make an edit also usually finds it useful to know whether she is 'long' or 'short' on the issue she is about to trade. For example, a user editing the value of $p(A|B)$ might like to know whether she should currently expect to gain more if B and A happens, or if B and $\neg A$ happens. Learning that she already stands to gain more if A and B happens should make a risk-averse trader more reluctant to raise the value of $p(A|B)$, thereby acquiring more such assets. In this paper we show how to efficiently maintain a representation of each trader's expected assets, where the expectation is with respect to the market consensus probabilities. We also show how to efficiently calculate the conditional expectations relevant for being 'long' or 'short' on a given edit.

The key word here is "efficiently". Existing combinatorial implementations follow the naïve joint-state enumeration described by Hanson (2007), and are therefore limited to at most about 20 related binary variables. Pennock and Xia (2011) proved that one can use Bayesian networks for combinatorial prediction markets. However, they treat an inverse system where assets are easy to calculate and prices are the bottleneck. Further, they did not have an implementation of their method. We use a single Bayesian (or Markov) network for representing the market probability distributions, stored as clique potentials in the associated junction tree. Further, we prove that assets can be represented using the same factorization as the joint distribution. Thus, our method maintains a parallel junction tree data structure for each user's assets. We describe algorithms for updating these asset junction trees when users make edits. Asset factorization is an important advance in space and computational efficiency. We have developed a complete MATLAB implementation and a partial Java implementation of our algorithms. To our knowledge, ours is the first published implementation of BN-based combinatorial prediction markets.

Although arbitrary graphical probability models are of course intractable, in this paper we show how to adapt the junction tree algorithm to perform the desired computations in models whose treewidth is not too large. It follows trivially that for a fixed treewidth, complexity is polynomial in the number of variables and linear in the number of simultaneous edits. This is a major improvement over existing naïve implementations, which are equivalent to a fully-connected graph and therefore exponential in the number of variables. We describe numerical experiments that demonstrate the anticipated exponential savings in space and time for given bounds on treewidth.

Organization of this paper: The next section states our definitions and notation. Section 3 briefly describes relevant work on Bayesian networks and the junction tree algorithm. Section 4.3 presents our probability and asset updating algorithm. In Section 5, we first walk through the algorithm using a simple 3-node BN model. We then examine the scalability of our algorithm in a simulation study. Sections 6 and 7 provide discussion and conclusion.

2 DEFINITIONS AND NOTATION

Capital letters such as A, B_i, X denote random variables. Bold capital letters (e.g., $\mathbf{A}, \mathbf{B}_i, \mathbf{X}$) denote vectors of random variables. Corresponding lowercase letters (e.g., $a, b_i, x, \mathbf{a}, \mathbf{b}_i, \mathbf{x}$) denote particular instantiations of the random variables. Unless stated otherwise, symbols p and ϕ denote probability distributions and likelihoods respectively. $\mathcal{B}$ represents a BN, $\mathcal{T}$ the corresponding clique tree, and $\mathbb{C}, \mathbb{S}$ the set of cliques and set of separators, respectively. The conditional probability distribution of X given Y is denoted by $p(X|Y)$. For a Bayesian network on random variables $\mathbf{X} = \langle X_1, X_2, \ldots, X_n \rangle$, we use $Pa(X_i)$ to denote the parents of X_i in the directed acyclic graph corresponding to the BN. Thus, the conditional probability distribution (CPD) of the random variable X_i is denoted by $p(X_i|Pa(X_i))$. We denote negation with "$\neg$", so $(X = \neg x) \equiv (X \neq x)$.

Each user u has an asset value $S_{\mathbf{x}}$ associated with each joint outcome $\mathbf{x}$. When necessary to index assets by user, we add a superscript, $S^u_{\mathbf{x}}$. The expected assets $\bar{S}$ for user u is the user's net worth:

$$\bar{S}^u = \sum_{\mathbf{x} \in \Omega} p(\mathbf{x}) S^u_{\mathbf{x}} \tag{1}$$

where Ω is the Cartesian product of the state spaces of all random variables. We denote the cardinality of the joint state space by $L = |\Omega|$.

3 PRELIMINARIES

If we represent the prediction market's probability distribution as a Bayesian network, then edits in the prediction market correspond to soft evidence on variables in the Bayesian network (see Section 3.1 below), and probability updating can be done using standard algorithms. Allowable edits are determined by the user's assets, and in particular that state which has the fewest assets for the edit being considered. We show below that the same junction tree can be used to represent a factorization of both probabilities and assets. The minimum asset state can be found by using min-propagation in the asset junction tree. That can be done is a consequence of a theorem by Dawid (1992) showing that min-propagation works for any function of the probabilities in a junction tree. We use this fact in Section 4.3.

A Bayesian network $\mathcal{B}$ factors the joint distribution for the random vector $\mathbf{X}$ into a product of local distributions:

$$p(\mathbf{x}) = \prod_{1 \leq k \leq n} p(X_k = x_k | \mathbf{X}_{Pa(X_k)} = \mathbf{x}_{Pa(X_k)})$$

This factorization in turn can be compiled into an undirected tree structure called a junction tree. The junction tree is composed of cliques and separators, such that p is the product of all clique marginal distributions divided by the product of all separator marginal distributions:

$$p(\mathbf{x}) = \frac{\prod_{c \in \mathbb{C}} p_c(\mathbf{x}_c)}{\prod_{s \in \mathbb{S}} p_s(\mathbf{x}_s)}. \tag{2}$$

Here, $\mathbf{x}_c$ and $\mathbf{x}_s$ denote the states of the variables in clique c and separator s, respectively, and p_c and p_s are the marginal distributions for the clique and separator variables, respectively. The junction tree algorithm (Lauritzen and Spiegelhalter, 1988) uses this transformation to perform exact inference on the BN. We note that although we focus on BNs, our algorithms apply equally well to any representation that can be compiled into a junction tree.

3.1 SOFT EVIDENCE & BELIEF UPDATING

General BN inference computes the posterior distribution given observations, also known as hard evidence. In prediction markets, we need to update market distributions given user edits that revise some non-extreme p to another non-extreme p' on variables. That is, evidence from user edits is in general uncertain. The literature considers two kinds of uncertain evidence. *Soft* evidence specifies a new probability distribution of the variable regardless of its previous distribution (Koski and Noble, 2009; Langevin and Valtorta, 2008; Valtorta et al., 2002), whereas *virtual* evidence, also known as *likelihood* evidence, represents the relative likelihood of the evidence given the true state. These likelihood values do not necessarily sum to 1 over all states. Virtual evidence is often implemented as observations on a hidden "dummy" node as the child node of the node on which virtual evidence is specified. In our case, we implement user edits as soft evidence. We use $\phi(X)$ to denote soft evidence on variable X. Usually, soft evidence on a single variable can be implemented as virtual evidence (Pearl, 1990), allowing standard junction tree inference algorithms to apply.

In our prediction market, a user edits the probability that the target variable T is in one of its states t, for example, changing $p(T = t) = a$ into $p(T = t) = b$. We then assign $1 - b$ to T's other states in proportion to their previous probabilities, and represent this edit

as soft evidence on T. Similarly, if the user would like to edit $p(T = t|\mathbf{A} = \mathbf{a})$ conditional on the values of other variables $\mathbf{A} = \mathbf{a}$, we represent this conditional edit as conditional soft evidence. An unconditional edit corresponds to an empty assumption set $\mathbf{A} = \emptyset$.

3.2 MIN-CALIBRATION OF JUNCTION TREE

In the standard junction tree algorithm for discrete BNs, marginalization is done by summing out variables. When we replace summation with maximization in the propagation algorithm, a max-calibrated junction tree with max potentials for all cliques will be returned. It is then straightforward to find the configurations over all states with maximum joint probability based on these max-potentials. Similarly, min-calibration of the junction tree can be performed by replacing summation with minimization, and the minimum probability configuration can be found accordingly. Further, Dawid (1992) proved that min/max-calibrations are valid for functions on the potentials. We will use this fact to find the boundary conditions for a user's assets, thus allowing asset reuse while preventing the possibility of assets going negative.

4 PROBABILITY AND ASSET UPDATING ALGORITHM

Once again, any edit in the prediction market is implemented by asserting soft evidence in the corresponding BN. We use the junction tree inference algorithm to update the consensus joint probability distribution after each edit. Then we use LMSR as the market maker to update the user's assets accordingly. We assume a market trading on purely discrete variables and so the representing BN is a purely discrete network. After updating, each clique in the junction tree maintains the correct joint distribution of variables in the clique.

A given user has assets $S_{\mathbf{x}}$ associated with every joint state $\mathbf{x}$ of the domain variables. LMSR dictates that assets change in proportion to the log of the ratio of probabilities:

$$\Delta S_{\mathbf{x}} = b \ln \frac{p'(\mathbf{x})}{p(\mathbf{x})},$$

where $p'(\mathbf{x})$ is the new probability for the joint state $\mathbf{x}$ arising from the user's edit, $p(\mathbf{x})$ is the previous probability, and b is a constant defining the unit of currency. Allowed changes must respect the rule that the minimum across all states of the user's assets must not be allowed to drop below zero, i.e., $\min_x S_x \geq 0$. After an edit to the consensus distribution, the asset data structure changes for the user making the edit; the asset data structures for all other users remain unchanged.

4.1 ASSET FACTORIZATION

It is convenient to define a transformation $q(\mathbf{x})$ of the assets $S_{\mathbf{x}}$, such that

$$S_{\mathbf{x}} = b \ln(q(\mathbf{x})). \tag{3}$$

We then have

$$\frac{q'(\mathbf{x})}{q(\mathbf{x})} = \frac{p'(\mathbf{x})}{p(\mathbf{x})}, \tag{4}$$

where $q(\mathbf{x})'$ is the updated asset for joint state $\mathbf{x}$, corresponding to the probability change $p'(\mathbf{x})$. The derivation is:

$$\begin{aligned} S'_{\mathbf{x}} &= S_{\mathbf{x}} + \Delta S_{\mathbf{x}} \\ &= S_{\mathbf{x}} + b \ln \frac{p'(\mathbf{x})}{p(\mathbf{x})} \\ &= b \ln(q(\mathbf{x})) + b \ln \frac{p'(\mathbf{x})}{p(\mathbf{x})} \\ &= b \ln(q'(\mathbf{x})) \end{aligned}$$

$$\Rightarrow b \ln(q'(\mathbf{x})) = b \ln(q(\mathbf{x})) + b \ln \frac{p'(\mathbf{x})}{p(\mathbf{x})}$$

$$\Rightarrow \ln(q'(\mathbf{x})) - \ln q(\mathbf{x}) = \ln \frac{p'(\mathbf{x})}{p(\mathbf{x})}.$$

Therefore, the identity $\ln a - \ln b = \ln a/b$ establishes Equation (4).

To interpret b, note that if the user changes the probability at state $\mathbf{x}$ from $p(\mathbf{x})$ to $p'(\mathbf{x}) = 1$ and $\mathbf{x}$ turns out to be true, the user gains $\Delta S_{\mathbf{x}} = -b \ln p(\mathbf{x})$. The maximum possible gain is therefore $-b \ln p(\mathbf{x}_*)$, where $\mathbf{x}_*$ is the minimum probability state. If all states start out equally likely, i.e., $p(\mathbf{x}) = 1/L$ for all $\mathbf{x}$, then the maximum gain to users, and the maximum loss to the market maker, is $b \ln L$, where L is the total number of states. Therefore to bound losses to be no more than M, we usually initialize all market states to be equally likely at the start of trading and set $b = M/\ln L$.

Because q starts out independent of the state and changes in proportion to changes in p, we can decompose q in a similar manner to the decomposition of p, shown in Equation (2). Specifically,

$$q(\mathbf{x}) = \frac{\prod_{c \in \mathbb{C}} q_c(\mathbf{x}_c)}{\prod_{s \in \mathbb{S}} q_s(\mathbf{x}_s)}, \tag{5}$$

where q_c and q_s are local asset components defined on the clique and separator variables, respectively. Notice the similarity to (2). This factored representation for assets is preserved long as edits are confined to variables in the same clique, i.e., trades are *structure preserving* (Pennock and Xia, 2011). This allows us to do all calculations locally in every clique and separator, because Equation (4) is valid for the joint space of

each clique and separator. Namely, we can establish the same junction tree structure for the assets q and make local updates when edits occur. When there is a probability edit, we propagate the soft evidence in the junction tree to obtain the correct probability update for every clique and separator. For structure preserving trades, assets can be updated simply by choosing a clique c containing the variables being traded and multiplying $q_c(\mathbf{x}_c)$ by the probability ratio:

$$q'_c(\mathbf{x}_c) = q_c(\mathbf{x}_c)\frac{p'_c(\mathbf{x}_c)}{p_c(\mathbf{x}_c)}. \quad (6)$$

Combining this with Equation (5) gives the same result as Equation (4). However, we usually do not need to compute global assets q for each possible joint state. For space and computational efficiency, a representation of the user's assets is stored locally in cliques of the asset junction tree. This asset junction tree is used to compute the minimum asset value and its associated state, as well as the expected value user's expected assets.

To ensure that the user's assets remain non-negative in all states, we must place limits on the edits a user is allowed to make. Equivalently, no edit may allow the transformed assets q as defined in Equation (3) to become less than 1. We must find the limits on edits beyond which the probability change will result in negative assets in some state. Let us assume that the user is editing $p(T = t|\mathbf{A} = \mathbf{a})$, denoted as p^t, to $p^\#$. Let m_t denote her current minimum q given $(\mathbf{A} = \mathbf{a}, T = t)$. Let $m_{\neg t}$ denote her current minimum q given $(\mathbf{A} = \mathbf{a}, T \neq t)$. If $(\mathbf{A} = \mathbf{a}, T = t)$ occur after the edit, we have to ensure that the updated minimum $q^\#$ remains greater than 1. That is:

$$q^\# = m_t\frac{p^\#}{p^t} \geq 1.$$

Then

$$p^\# \geq \frac{p^t}{m_t}.$$

Similarly, if $(\mathbf{A} = \mathbf{a}, T \neq t)$ occurs after the edit, the updated minimum assets q^u must remain greater than 1. That is:

$$q^u = m_{\neg t}\frac{1-p^\#}{1-p^t} \geq 1.$$

It follows that

$$p^\# \leq 1 - \frac{1-p^t}{m_{\neg t}}.$$

Summarizing the above results, the allowable edit range for $p(T = t|\mathbf{A} = \mathbf{a})$ is:

$$\left[\frac{p(T = t|\mathbf{A} = \mathbf{a})}{m_t}, 1 - \frac{1 - p(T = t|\mathbf{A} = \mathbf{a})}{m_{\neg t}}\right]. \quad (7)$$

Note that the minimum assets can be found by min-calibration over the asset junction tree, as mentioned in Section (3.2).

For every user, we maintain a separate asset junction tree in which the affected clique is updated only after this particular user makes an edit. Edits made by a given user will have no effect other users' assets. But because every edit changes the market probability, we update market distribution after each edit accordingly (where the updates are stored as clique potentials of the junction tree).

4.2 EXPECTED VALUE / SCORE

A user typically wants to know the expected value of her assets given the current market consensus prices. If she is contemplating an edit to event A, she would want to know what her expected assets will be if A happens and if $\neg A$ happens. These expectations can be calculated efficiently given the factorization represented in the junction tree. The expected score is obtained as follows (recall that c indexes cliques and s indexes separators).

$$\bar{S} = \sum_c \sum_{x_c} S_c(\mathbf{x}_c)p_c(\mathbf{x}_c) - \sum_s \sum_{x_s} S_s(\mathbf{x}_s)p_s(\mathbf{x}_s), \quad (8)$$

where

$$S_c(\mathbf{x}_c) = b\ln(q_c(\mathbf{x}_c)), \quad (9)$$

and

$$S_s(\mathbf{x}_s) = b\ln(q_s(\mathbf{x}_s)). \quad (10)$$

This result is derived as follows. First, we substitute (5) into (3) and then substitute (9) and (10) into the result to obtain the following expression for $S_\mathbf{x}$:

$$S_\mathbf{x} = \left[\sum_c S_c(\mathbf{x}_c) - \sum_s S_s(\mathbf{x}_s)\right]. \quad (11)$$

Next, we substitute (11) into (1), suppressing the superscript u, to obtain:

$$\begin{aligned}\bar{S} &= \sum_x p(\mathbf{x})\left[\sum_c S_c(\mathbf{x}_c) - \sum_s S_s(\mathbf{x}_s)\right] \\ &= \sum_c \sum_x S_c(\mathbf{x}_c)p(\mathbf{x}) - \sum_s \sum_x S_s(\mathbf{x}_s)p(\mathbf{x}).\end{aligned}$$

Letting $\mathbf{x}_{\neg c}$ and $\mathbf{x}_{\neg s}$ denote the states of the variables not in clique c and separator s, respectively, we obtain:

$$\begin{aligned}\bar{S} &= \sum_c \sum_{\mathbf{x}_c} S_c(\mathbf{x}_c) \sum_{\mathbf{x}_{\neg c}} p(\mathbf{x}) - \sum_s \sum_{\mathbf{x}_s} S_s(\mathbf{x}_s) \sum_{\mathbf{x}_{\neg s}} p(\mathbf{x}) \\ &= \sum_c \sum_{x_c} S_c(\mathbf{x}_c)p_c(\mathbf{x}_c) - \sum_s \sum_{x_s} S_s(\mathbf{x}_s)p_s(\mathbf{x}_s),\end{aligned}$$

which establishes (8). Further, let

$$\bar{S}_c = \sum_{x_c} S_c(\mathbf{x}_c) p_c(\mathbf{x}_c),$$

and

$$\bar{S}_s = \sum_{x_s} S_s(\mathbf{x}_s) p_s(\mathbf{x}_s).$$

Then for brevity, we can write

$$\bar{S} = \sum_c \bar{S}_c - \sum_s \bar{S}_s.$$

Clearly, the values $\bar{S}_c$ and $\bar{S}_s$ represent local expected scores for cliques and separators, respectively. This local score decomposition demonstrates once again the beauty of factorization.

4.3 PROBABILITY AND ASSET UPDATING ALGORITHM

To summarize our procedure for updating probabilities and assets, please see Algorithm 1. The algorithm maintains a consensus probability distribution and an asset structure for each user. The probability distribution and the asset structures factor according to the same junction tree. After each edit, the assets of the user making the edit and the consensus distribution are updated.

Algorithm 1 Update probability and user assets for a BN representing a combinatorial prediction market

Require: a BN model $\mathcal{B}$ over a set of domain variables $\mathbf{X}$ that represents the combinatorial prediction market joint distribution, the clique tree $\mathcal{T}$ corresponding to $\mathcal{B}$, consisting of cliques $\mathbb{C}$ and separators $\mathbb{S}$.

Require: The current market probability distribution p represented by a probability junction tree.

Require: The current assets q for user u represented by an assets junction tree.

for each conditional edit p' on the target variable $T = t$ with assumptions $\mathbf{A} = \mathbf{a}$ by user u: **do**

- Tell the user the expected scores $\bar{S}(T = t, \mathbf{A} = \mathbf{a})$ and $\bar{S}(T \neq t, \mathbf{A} = \mathbf{a})$ indicating the user's long/short status.
- Calculate the edit limits for $p(T = t|\mathbf{A} = \mathbf{a})$ using Equation (7).
- Allow user to trade $p'(T = t|\mathbf{A} = \mathbf{a})$ within the edit limits. And apply $p'(T = t|\mathbf{A} = \mathbf{a})$ as soft evidence to the junction tree.
- Update probability distributions of cliques and separators to be $p'_c,\ p'_s,\ c \in \mathbb{C}, s \in \mathbb{S}$ by calling the junction tree inference algorithm.
- Find the clique c containing target variable T and assumed variables $\mathbf{A}$, and update the assets clique corresponding to c using Equation (6).

end for

return User's expected assets after the edit; user's min-q value and its associated min-q states.

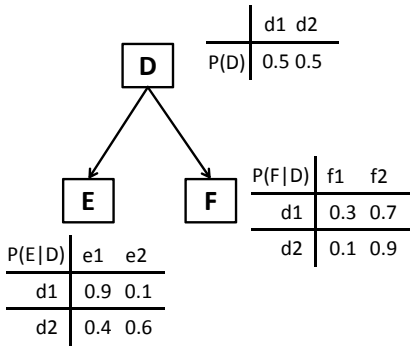


Figure 1: *BN-DEF*: An Example network With Three Binary Nodes D, E, and F.

5 NUMERICAL EVALUATION

5.1 TEST CASES FOR *BN-DEF*

We developed a complete MATLAB implementation for our algorithm using BNT (Murphy, 2001) and currently we are developing a Java implementation in UnBBayes (Matsumoto et al., 2011). Our implementation requires the target variable and all assumed (conditioning) variables to belong to the same clique. Cross-clique conditioning is possible, but requires approximations (see Section 6).

In this section, we illustrate how the algorithm works using the simple 3-node BN model *BN-DEF*, shown in Figure 1. The CPDs for nodes in *BN-DEF* are shown in the picture next to the corresponding nodes. This network has only two cliques $\{D, E\}$ and $\{D, F\}$. We work through the algorithm, showing intermediate results.

We begin by initializing q to be 100 for every cell in the asset tables. The scale parameter b is specified as $10/\ln(100)$, making the initial asset score 10 for every state for all users. We imagine two users, Joe and Amy, making successive edits.

Joe's first edit - Suppose Joe thinks the current market probability of $E = e1$ (0.65) is not reasonable, and wants to increase it. Since this will be his first trade, his initial uniform q makes his current position neutral because both min-q values given $E = e1$ and $E \neq e1$ are equal to 100. Using Equation (7), we calculate his edit limits on $E = e1$ to be $[0.0065, 0.9965]$. Based on his private information, Joe chooses 0.8 as the new $p(E = e1)$. The market distribution is then updated such that marginal probabilities of D, E, F are now $[0.58, 0.42], [0.8, 0.2]$, and $[0.22, 0.78]$ respectively. Joe's expected assets are $\bar{S} = 10.12$, and his min-q is 57.14 at two min-states - $\{d2, e2, f1\}$, and $\{d2, e2, f2\}$.

Amy's first edit - Now Amy is interested in changing $p(D = d1|F = f2)$, which due to Joe's edit is currently 0.52. Again, since this will be Amy's first trade, she has neutral positions. Using Equation (7), we find her edit limits on $D = d1$ given $F = f2$ to be $[0.0052, 0.9952]$. Suppose Amy wants to move the probability to 0.7. Now, the market distribution is updated such that the marginal probabilities of D, E, F are now $[0.72, 0.28]$, $[0.85, 0.15]$, and $[0.22, 0.78]$ respectively. Note that F is unaffected – it was assumed. Amy's expected assets are $\bar{S} = 10.11$, and her min-q is 62.54 at two min-states - $\{d2, e1, f2\}$, and $\{d2, e2, f2\}$. Note that Amy's edit does not affect any other users' asset data.

Joe's second edit - Let us now consider an extreme edit example. Joe comes back and wants to see what will happen if he makes a big move on $p(E = e1|D = d2)$ (currently equal to 0.59). First of all, Joe finds that he has a long position for this trade since his $\bar{S}(E = e1, D = d2) = 10.45$, and $\bar{S}(E \neq e1, D = d2) = 8.79$. Second, his edit limits returned by the algorithm are $[0.0048, 0.9928]$. If he decides to move $p(E = e1|D = d2)$ to 0.99, then the marginal probabilities of D, E, F can be updated to $[0.72, 0.28]$, $[0.96, 0.04]$ and $[0.22, 0.78]$, respectively. Joe's expected assets after this trade are $\bar{S} = 10.67$, and his min-q is 1.39 at two min-states - $\{d2, e2, f1\}$, and $\{d2, e2, f2\}$. Note that in this case, Joe's min-q is very close to the threshold.

5.2 SCALABILITY STUDY

To investigate scalability, we first tested our algorithms on randomly generated networks of varying sizes. Our random networks were generated using BN-Generator 0.3 (Ide et al., 2004). We varied the number of variables from 30 to 960 and the treewidth bound from 5 to 20. For each combination of number of variables and the treewidth, three random BNs were generated. All random variables were binary. For each randomly generated network, we calculated the system lock time, defined as the CPU time required to update the probability distribution after an edit is confirmed by the user. While the system is locked, new edits have to be rejected or queued; thus lock time is a key performance metric. We ran this test on our Java implementation. Although not fully complete, the Java implementation performs probability updating, the part of the calculation required to determine the system lock time.

Figure 2 shows the results of our scalability study. As is well known, the complexity of the junction tree algorithm – and therefore the maximum lock time – is polynomial in the treewidth. We can see from the figure that the system can easily handle binary networks

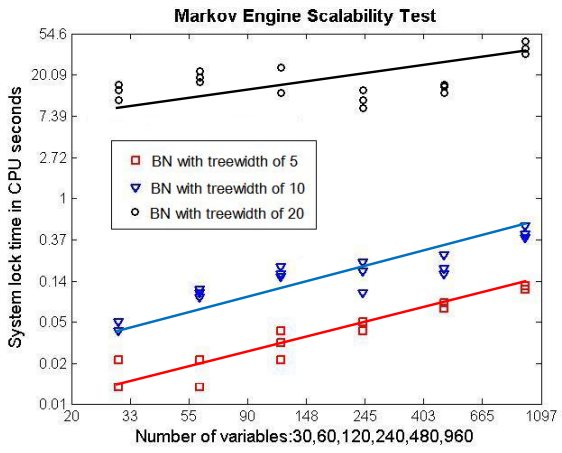


Figure 2: Edit Lock Time versus Network Size

of treewidth 10, but performance begins to degrade noticeably with treewidth 20. The algorithm is not very sensitive to increases in the number of variables when the treewidth is held constant.

We also investigated another key performance parameter, the potential rejection rate under different market environments, modeled by frequency of edits. As noted above, edits attempted while the system is locked may have to be rejected. For this study, we used ALARM (Beinlich et al., 1989), a 37-node BN with treewidth of 4 often used as a benchmark for graphical model algorithms. We chose ALARM because its size and treewidth are at the upper end of the range we expect for our live combinatorial prediction market. The objective of this test is to investigate how our approach performs with frequent edits by multiple users.

We simulated a market with 100 participants, and expected edits of 2/minute, 8/minute, and 30/minute, assuming a Poisson distribution for edit arrivals. For comparison, on Super Tuesday, InTrade had 18,629 trades from 780 unique users, for an average of 13 trades per minute. Prediction market trades are quite lumpy in both questions and time; we think our figures span a plausible range. The system lock time for ALARM is 0.3 seconds in our MATLAB implementation. We did not implement enhancements such as lazy propagation (Madsen and Jensen, 1998) that would give further edit-dependent reductions in lock time. 1000 edits were simulated by randomly chosen users from 100 market participants, given randomly chosen assumed variables and betting on randomly chosen target variable from randomly chosen cliques among the 27 cliques of ALARM's junction tree. If the randomly chosen clique had more than 3 variables, we then randomly chose two variables with their randomly chosen

states to be assumed values on which the edit was conditioned. Inter-arrival times between edits were modeled by exponentially distributed variables with means of 2, 7.5 and 30 seconds respectively, representing market intensities of 30, 8 and 2 edits per minute, correspondingly. Table 1 presents the average number of rejects and the rejection rates for our experiments.

Table 1: ALARM: Simulation Results for 1000 Edits

Market Intensity	Average rejects	Average rejection rate
2 edits/minute	11.3	1.2%
8 edits/minute	39.2	4%
30 edits/minute	142.6	14.3%

In this network, the system could sustain an arrival rate of 8 edits/minute (11,520/day) with less than a 5% rejection rate. By queuing edits and rejecting only those whose terms have become worse for the user, this can be at least halved. A compiled language would be much faster for all the non-matrix operations, and with a good choice of library, not much slower at matrix operations.

6 NON STRUCTURE-PRESERVING EDITS

When we have conditional edits such that the target variable and the assumed variables are not in the same clique, then the edits are not structure preserving. In that case, if we keep our sparser structure, then the minimum KL-distance approximation to the true posterior distribution will be the one that has the same marginal distributions for all the cliques (Darwiche, 2009).

7 CONCLUSION AND FUTURE WORK

In this paper, we describe a method to update both the probabilities and assets in a combinatorial prediction market. The approach uses a junction tree compiled from the associated Bayesian network to represent the consensus probability distribution, and a structurally identical junction tree to represent each participant's assets. Allowable edits are calculated using min-propagation in the assets junction tree, and an allowable edit is chosen. The probability distribution is updated by applying soft evidence to the (shared) probability junction tree, and propagating. Assets are updated by simple multiplication in a single clique, without requiring a separate propagation. We developed a complete implementation of our algorithm and demonstrated its scalability by performing computational experiments on randomly generated BNs with varying sizes from 30 to 960 variables, and treewidth of 5 to 20. Further, we conducted a simulation study to illustrate the robustness of our method under varying market environments. Test results show that the system performs well for networks with treewidth of up to 10, and can sustain a simulated market with expected arrival rate of 8 edits/minute with less than 5% rejection rate.

Future work will measure the computational savings (or increase in network size) compared to a more straightforward combinatorial prediction market algorithm. We also plan use this technique in an online forecasting system, and to examine the tradeoff between a minimum-KL approximation (which may confuse users) and a less accurate but simpler set of assumptions.

Acknowledgements

The authors gratefully acknowledge support from the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20062. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

We also gratefully acknowledge the hard work of graduate student Shou Matsumoto for coding the Java implementation.

References

Barbu, A. and Lay, N. (2011). An introduction to artificial prediction markets for classification. *arXiv:1102.1465*.

Beinlich, I., Suermondt, G., Chavez, R., and Cooper, G. (1989). The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceeding of 2nd European Conference on AI and Medicine*.

Chen, Y., Fortnow, L., Lambert, N., Pennock, D. M., and Wortman, J. (2008a). Complexity of combinatorial market makers. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC)*, pages 190–199.

Chen, Y., Goel, S., and Pennock, D. M. (2008b). Pricing combinatorial markets for tournaments. In *Pro-*

ceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC-2008), pages 305–314.

Chen, Y. and Pennock, D. M. (2010). Designing markets for prediction. *AI Magazine*, 31(4):42–52.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Dawid, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36.

Hanson, R. (2003). Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119.

Hanson, R. (2007). Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):3–15.

Ide, J. S., Cozman, F., and Ramos, F. (2004). Generating random Bayesian networks with constraints on induced width. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04)*, pages 323–327, Amsterdam. IOS Press.

Jensen, F. (1996). *An Introduction to Bayesian Networks*. Springer-Verlag, New York.

Koski, T. and Noble, J. (2009). *Bayesian Networks: An Introduction*. Wiley, 1st edition.

Langevin, S. and Valtorta, M. (2008). Performance evaluation of algorithms for soft evidential update in Bayesian networks: First results. In *Proceedings of the Second International Conference on Scalable Uncertainty Management (SUM-08)*, pages 294–297.

Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their applications to expert systems. In *Proceedings of the Royal Statistical Society, Series B.*, volume 50, pages 157–224.

Madsen, A. L. and Jensen, F. V. (1998). Lazy propagation in junction trees. In *In Proc. 14th Conf. on Uncertainty in Artificial Intelligence*, pages 362–369. Morgan Kaufmann Publishers.

Matsumoto, S., Carvalho, R. N., Ladeira, M., da Costa, P. C. G., Santos, L. L., Silva, D., Onishi, M., Machado, E., and Cai, K. (2011). UnBBayes: a java framework for probabilistic models in AI. In *Java in Academia and Research*. iConcept Press.

Murphy, K. P. (2001). The Bayes net toolbox for matlab. *Computing Science and Statistics*, 33:2001.

Nagar, Y. and Malone, T. (2011). Making business predictions by combining human and machine intelligence for making predictions. In *Proceedings of the Thirty Second International Conference on Information Systems (ICIS 2011)*, Shanghai. ICIS. Expanded from NIPS 2010.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffman, San Mateo.

Pearl, J. (1990). Jeffrey's rule, passage of experience, and neo-Bayesianism. In *Knowledge Representation and Defeasible Reasoning*, pages 245–265. Kluwer Academic Publishers.

Pennock, D. and Xia, L. (2011). Price updating in combinatorial prediction markets with Bayesian networks. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pages 581–588, Corvallis, Oregon. AUAI Press.

Shachter, R. D., D'Ambrosio, B., and Del Favero, B. A. (1990). Symbolic probabilistic inference in belief networks. In *Proceedings of the eighth National conference on Artificial intelligence - Volume 1*, AAAI'90, page 126131. AAAI Press.

Solomonoff, R. J. (1978). Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Transactions on Information Theory*, IT-24:422–432.

Tziralis, G. and Tatsiopoulos, I. (2007). Prediction markets: An extended literature review. *Journal of Prediction Markets*, 1(1):75–91.

Valtorta, M., Kim, Y.-G., and Vomlel, J. (2002). Soft evidential update for probabilistic multiagent systems. *International Journal of Approximate Reasoning*, 29(1):71–106.

Fast Exact Inference for Recursive Cardinality Models

Daniel Tarlow, Kevin Swersky, Richard S. Zemel,
Dept. of Computer Science
University of Toronto
{dtarlow,kswersky,zemel}@cs.toronto.edu

Ryan P. Adams,
Sch. of Eng. & Appl. Sci.
Harvard University
rpa@seas.harvard.edu

Brendan J. Frey
Prob. & Stat. Inf. Group
University of Toronto
frey@psi.toronto.edu

Abstract

Cardinality potentials are a generally useful class of high order potential that affect probabilities based on how many of D binary variables are active. Maximum a posteriori (MAP) inference for cardinality potential models is well-understood, with efficient computations taking $\mathcal{O}(D \log D)$ time. Yet efficient marginalization and sampling have not been addressed as thoroughly in the machine learning community. We show that there exists a simple algorithm for computing marginal probabilities and drawing exact joint samples that runs in $\mathcal{O}(D \log^2 D)$ time, and we show how to frame the algorithm as efficient belief propagation in a low order tree-structured model that includes additional auxiliary variables. We then develop a new, more general class of models, termed *Recursive Cardinality models*, which take advantage of this efficiency. Finally, we show how to do efficient exact inference in models composed of a tree structure *and* a cardinality potential. We explore the expressive power of Recursive Cardinality models and empirically demonstrate their utility.

1 Introduction

Probabilistic graphical models are widely used in machine learning due to their representational power and the existence of efficient algorithms for inference and learning. Typically, however, the model structure must be restricted to ensure tractability. To enable efficient *exact* inference, the most common restriction is that the model have low tree-width.

A natural question to ask is if there are other, different restrictions that we can place on models to ensure tractable exact or approximate inference. Indeed, a celebrated result is the ability of the "graph cuts" algorithm to exactly find the maximum a posteriori (MAP) assignment in any pairwise graphical model with binary variables, where the internal potential structure is restricted to be submodular. Along similar lines, polynomial-time algorithms can exactly compute the partition function in an Ising model if the underlying graph is planar (Fisher, 1961).

Extensions of these results have been a topic of much recent interest, particularly for the case of MAP inference. Gould (2011) shows how to do exact MAP inference in models with certain higher order terms via graph cut-like algorithms, and Ramalingham et al. (2008) give results for multilabel submodular models. Tarlow et al. (2010) provide efficient algorithms for a number of other high-order potentials.

Despite these successes in finding the optimal configuration, there has been relatively less progress in efficient high order marginalization and sampling. This partially stems from the difficulty of some of the computations associated with summation in these models. For example, computing the partition function for binary pairwise submodular models (where graph cuts can find the MAP) is #P-complete, so we do not expect to find an efficient exact algorithm.

One important high-order potential where such hardness results do not exist is the cardinality potential, which expresses constraints over the number of variables that take on a particular value. Such potentials come up in natural language processing, where they may express a constraint on the number of occurrences of a part-of-speech, e.g., that each sentence contains at least one verb. In computer vision, a cardinality potential might encode a prior distribution over the relationships between size of an object in an image and distance from camera. In a conference paper matching system, cardinality potentials could enforce a requirement that e.g. each paper have 3-4 reviews and each reviewer receive 8-10 papers.

A simple form of model containing a cardinality potential is a model over binary variables, where the model probability is a Gibbs distribution based on an energy function consisting of unary potentials θ_d and one cardinality potential $f(\cdot)$:

$$-E(\boldsymbol{y}) = \sum_d \theta_d y_d + f(\sum_d y_d) \quad (1)$$

$$p(\boldsymbol{y}) = \frac{\exp\{-E(\boldsymbol{y})\}}{\sum_{\boldsymbol{y}'} \exp\{-E(\boldsymbol{y}')\}}, \quad (2)$$

where no restrictions are placed on $f(\cdot)$. We call this the *standard cardinality potential model*. Perhaps the best-known algorithm in machine learning for computing marginal probabilities is due to Potetz and Lee (2008); however, the runtime is $\mathcal{O}(D^3 \log D)$, which is impractical for larger problems.

We observe in this paper that there are lesser-known algorithms from the statistics and reliability engineering literature that are applicable to this task. Though these earlier algorithms were not presented in terms of a graphical modeling framework, we will present them as such, introducing an interpretation as a two step procedure: (i) create auxiliary variables so that the high order cardinality terms can be re-expressed as unary potentials on auxiliary variables, then (ii) pass messages on a tree-structured model that includes original and auxiliary variables, using a known efficient message computation procedure to compute individual messages. The runtime for computing marginal probabilities with this procedure will be $\mathcal{O}(D \log^2 D)$. This significant efficiency improvement over the Potetz and Lee (2008) approach makes the application of cardinality potentials practical in many cases where it otherwise would not be. For example, exact maximum likelihood learning can be done efficiently in the standard cardinality potential model using this formulation.

We then go further and introduce a new high order class of potential that generalizes cardinality potentials, termed Recursive Cardinality (RC) potentials, and show that for balanced RC structures, exact marginal computations can be done in the same $\mathcal{O}(D \log^2 D)$ time. Additionally, we show how the algorithm can be slightly modified to draw an exact sample with the same runtime. We follow this up by developing several new application formulations that use cardinality and RC potentials, and we demonstrate their empirical utility. The algorithms are equally applicable within an approximate inference algorithm, like loopy BP, variational message passing, or tree-based schemes. This also allows fast approximate inference in multi-label models that contain cardinality potentials separately over each label.

Finally, we show that cardinality models can be combined with a tree-structured model, and again assuming a balanced tree, exact inference can be done in the same $\mathcal{O}(D \log^2 D)$ time (for non-balanced trees, the runtime is $\mathcal{O}(D^2)$). This leads to a model class that strictly generalizes standard tree structures, which is also able to model high order cardinality structure.

2 Related Work

2.1 Applications of Cardinality Potentials

Cardinality potentials have seen many applications, in diverse areas. For example, in worker scheduling programs in the constraint programming literature, they have been used to express regulations such "each sequence of 7 days must contain at least 2 days off" and "a worker cannot work more than 3 night shifts every 8 days" (Régin, 1996). Milch et al. (2008) develop cardinality terms in a relational modeling framework, using a motivating example of modeling how many people will attend a workshop. In error correcting codes, message passing-based decoders often use constraints on a sum of binary variables modulus 2 (Gallager, 1963). Another application is in graph problems, such as finding the maximum-weight b-matching, in which the cardinality parameter b constrains the degree of each node in the matching (Huang & Jebara, 2007), or to encode priors over sizes of partitions in graph partitioning problems (Mezuman & Weiss, 2012).

More recently, cardinality potentials have become popular in language and vision applications. In part-of-speech tagging, cardinalities can encode the constraint that each sentence contains at least one verb and noun (Ganchev et al., 2010). In image segmentation problems from computer vision, they have been utilized to encourage smoothness over large blocks of pixels (Kohli et al., 2009), and Vicente et al. (2009) show that optimizing out a histogram-based appearance model leads to an energy function that contains cardinality terms.

2.2 Maximization Algorithms

As noted previously, there is substantial work on performing MAP inference in models containing one or more cardinality potentials. In these works, there is a division between methods for restricted classes of cardinality-based potential, and those that work for arbitrary cardinality potentials. When the form of the cardinality potential is restricted, tractable exact maximization can sometimes be performed in models that contain many such potentials, e.g., Kohli et al. (2009); Ramalingham et al. (2008); Stobbe and Krause (2010); Gould (2011). A related case, where maximization can only be done approximately is the "pattern potentials" of Rother et al. (2009). For arbitrary functions of counts, the main approaches are that of Gupta et al. (2007) and Tarlow et al. (2010). The former gives

a simple $\mathcal{O}(D \log D)$ algorithm for performing MAP inference, and the latter gives an algorithm with the same complexity for computing messages necessary for max-product belief propagation.

2.3 Summation Algorithms

Relatively less work in the machine learning community has examined efficient inference of marginal probabilities in models containing cardinality potentials. The best-known approach is by Potetz and Lee (2008) (here PL). Also related is the line of work including de Salvo Braz et al. (2005) and Milch et al. (2008), but these works assume restrictions on unary potentials, and it is not clear that they are efficient in the case where there are many distinct unary potentials.

PL works with potentials $\theta(\boldsymbol{y}; \boldsymbol{w}) = f(\boldsymbol{y} \cdot \boldsymbol{w})$, where $\boldsymbol{y}$ and $\boldsymbol{w}$ are real-valued vectors. This is a general case that is more involved than cardinality potentials, requiring a clever strategy for representing messages, e.g., with adaptive histograms. However, if $\boldsymbol{y}$ is binary and $\boldsymbol{w}$ is the all-ones vector, then $f(\cdot)$ is a standard cardinality potential.

The starting point for PL is to to write down the expression for a sum-product message from f to, say, the last variable y_D:

$$m_{fD}(y_D) = \sum_{\boldsymbol{y} \setminus \{y_D\}} \left[f(\sum_d y_d) \prod_{d' \neq D} m_{d'f}(y_{d'}) \right]. \quad (3)$$

Next, PL defines a change of variables into a new set of integer-valued variables, $\boldsymbol{z}$, as follows:

$$z_1 = y_1 \qquad z_2 = z_1 + y_2 \qquad \ldots \qquad z_{D-1} = z_{D-2} + y_{D-1}.$$

Eq. (3) can then be expressed in terms of $\boldsymbol{z}$ as $\sum_{\boldsymbol{z} \setminus \{z_d\}} \Big[f(y_D + z_{D-1}) \prod_{d' \neq d} m_{d'f}(z_{d'} - z_{d'-1}) \Big]$.[1] Finally, the sums can be pushed inwards as in variable elimination, and the internal sums can be computed from inside outwards naively in time $\mathcal{O}(D^2)$. There are D summations to perform, so computing one message takes $\mathcal{O}(D^3)$ time. As observed by Felzenszwalb and Huttenlocher (2004) and noted by PL, the internal sums can be performed via FFTs in $\mathcal{O}(D \log D)$ time. Computing all D messages or marginals, then, requires $\mathcal{O}(D^3 \log D)$ time.

3 Other Summation Algorithms

Here, we review lesser-known work from the statistics and reliability engineering literature, where efficient algorithms for very similar tasks have been developed.

The idea of a recursive procedure that sums configurations in a chain-structure i.e. by using a definition of z variables similar to that of PL, dates back at least to Gail et al. (1981). In this work, the algorithmic challenge is to compute the probability that exactly k of D elements are chosen to be on, given that elements turn on independently and with non-uniform probabilities. Naively, this would require summing over $\binom{D}{k}$ configurations, but Gail et al. shows that it can be done in $\mathcal{O}(Dk)$ time using dynamic programming.

A similar task is considered by Barlow and Heidtmann (1984) and Belfore (1995) in the context of reliability engineering. The task considered computing the probability that exactly k elements are chosen to be on, or the probability that between k and l elements are chosen to be on. Belfore gives a divide-and-conquer algorithm that recursively calls an FFT routine, leading to an $O(D \log^2 D)$ algorithm. This algorithm is very similar to the approach we take in this work, and in the case of ordinary cardinality potentials (i.e. not RC potentials), it is equivalent to the upward pass of message passing that we will present in Section 5.

Finally, there is work in statistics on Poisson-Binomial (PB) distributions, which are the marginal distributions over cardinalities that arise if we have a model that contains only unary potentials. These distributions have been used in several applications in the statistics literature. We refer the reader to Chen and Liu (1997) and references therein for more details on applications.

One algorithm for computing the cumulative distribution function (CDF) of the PB distribution that uses an FFT was proposed in Fernandez and Williams (2010) and analyzed by Hong (2011). The idea is to first compute the characteristic function of the distribution, and then directly apply the inverse FFT in order to recover the CDF. The benefit of this approach is that the FFT only needs to be applied once, however computing the characteristic function still takes $\mathcal{O}(D^2)$ time.

For further discussion of other algorithms similar to the ones discussed above, we recommend Hong (2011) and references therein.

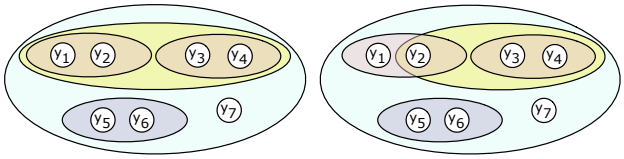
4 Recursive Cardinality Potentials

4.1 Model Structure

The first contribution in this paper is to generalize the structure of the cardinality potential to the Recursive Cardinality potential. Recursive Cardinality potentials are defined in terms of a set of subsets $\mathcal{S}$ of a set of D binary variables. The joint probability over vector $\boldsymbol{y}$ is defined as

$$p(\boldsymbol{y}) \propto \prod_{s_k \in \mathcal{S}} f_k(\sum_{y_d \in s_k} y_d), \quad (4)$$

[1] For notational convenience, assume we have $z_0 = 0$.



(a) Nested (b) Not nested

Figure 1: Examples of nested and non-nested subsets.

where the only constraint is that $\mathcal{S}$ is *nested*, as illustrated in Fig. 1. We call a set of subsets $\mathcal{S}$ nested if for every pair of subsets in $\mathcal{S}$, either they are disjoint or one is a subset of the other. Each f_k can be arbitrary, and different per k. By defining subsets over single variables, we can represent unary potentials, so we do not explicitly separate them out here.

This new construction extends the standard cardinality potential to handle multiple scales, ranging from purely local (e.g., for a pair of variables) to global, and potentially including all scales in between.

4.2 Example Cardinality Potentials

Cardinality potentials can be applied in diverse ways in order to capture a variety of interesting properties. Here we give some examples, and note that this list is far from exhaustive.

4.2.1 Ordinary Cardinality Potentials

Noisy-OR: Consider a set of D binary variables $\boldsymbol{y}$ along with a single binary variable t that depends on $\boldsymbol{y}$: $P(t=1|\boldsymbol{y}) = 1-(1-\epsilon)\prod_{d=1}^{D}(1-\lambda_d)^{y_d}$. For the special case where each λ_d is equal, we can represent this as a conditional model with a cardinality potential:

$$P(t=1|\boldsymbol{y}) = 1-(1-\epsilon)(1-\lambda)^{\sum_{d=1}^{D} y_d} \qquad (5)$$

This is simply a function of $\sum_d y_d$, and can therefore be viewed as a cardinality potential.

Smoothness and Competition: Note that even unimodal cardinality potentials have interesting properties. A convex cardinality potential is related to smoothness: it will tend to favor configurations where all of the variables are either on together or off together. In a similar manner, a concave function will cause competition amongst the binary variables.

4.2.2 Recursive Cardinality Potentials

Group and structured sparsity: By placing a cardinality potential over subsets of variables consisting of a uniform distribution over counts (or any general distribution) along with a spike at 0, one can represent the preference that variables should tend to turn off together in groups. Indeed, one can represent hierarchical sparsity using a recursive model. This gives an interesting alternative to the traditional approaches of ℓ_0 and ℓ_1 priors, as well as their group counterparts that are commonly used in structured sparsity (Zhao et al., 2006).

Hierarchical CRFs: A common approach to image segmentation in the computer vision literature is to construct a hierarchy of increasingly coarse segmentations, then to perform the segmentation jointly at the different levels of coarseness. To enforce consistency across levels of coarseness, a widely-used form of potential is the P^n (Kohli et al., 2009) potential, which encourages sets of variables to all take on the same label. If the segmentations at different levels of granularity have a nested structure, then this model can be represented using an RC potential, and thus exact marginal inference (and therefore learning) can be done efficiently.

5 Fast Sum-Product Formulation

Overview. Here, we present the fast FFT algorithm as an auxiliary variable method, where auxiliary variables are invented in such a way that the distribution over $\boldsymbol{y}$ remains unchanged, but inference in the expanded model can be done very efficiently. In other words, we are defining an augmented model $q(\boldsymbol{y}, \boldsymbol{z})$ that has the property that $\sum_z q(\boldsymbol{y}, \boldsymbol{z}) = p(\boldsymbol{y})$, but where computing all marginals in q (for both $\boldsymbol{y}$ and $\boldsymbol{z}$ variables) can be done more efficiently than directly computing marginals for each y in the original $p(\boldsymbol{y})$ (at least by using any existing method).

More specifically, the algorithm can be described as follows: auxiliary variables $\boldsymbol{z}$ will be integer-valued variables that represent the count over subsets of original variables $\boldsymbol{y}$. The auxiliary variables are structured into a binary tree, as illustrated in Fig. 2, with original $\boldsymbol{y}$ variables at leaves of the tree, so z variables at higher levels of the tree represent sums of increasingly large subsets of $\boldsymbol{y}$. There is one auxiliary variable at every internal binary tree node, so with a balanced tree structure, the joint model over $\boldsymbol{y}$ and $\boldsymbol{z}$ is a tree of depth $\log D$. The utility of this formulation is that we can now represent cardinality potentials over subsets of $\boldsymbol{y}$ variables as unary potentials over $\boldsymbol{z}$ variables.

Having constructed the auxiliary variable model, the algorithm will be simply to run an inwards and outwards pass of sum-product belief propagation. The key computational point is that due to the structure of potentials over auxiliary variables, sum-product messages can be computed efficiently even when $\boldsymbol{z}$ variables have large cardinality, by using FFTs.

5.1 Detailed Description

The algorithm takes as input a binary tree $\mathcal{T}$ with D leaf nodes, where each leaf corresponds to one y_d

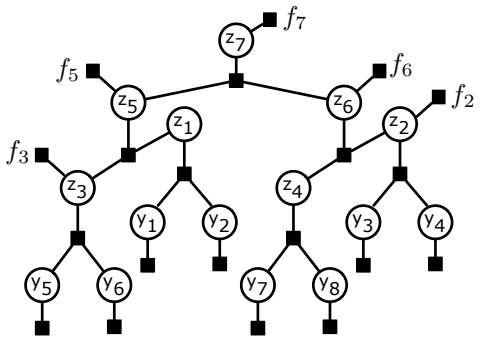


Figure 2: Each internal node z represents the count over the subset of original y variables that are descendants of the internal node. Cardinality potentials on the subset then can trivially be added as unary potentials on internal nodes. Here, in addition to a standard cardinality potential f_7, we add subset cardinality potentials f_2, f_3, f_5, and f_6.

variable. To instantiate latent variables $\boldsymbol{z}$, traverse up the tree from leaves to root, associating a z variable with each internal node. When instantiating a new variable z_p, set a deterministic relationship between it and its two children, z_l and z_r, as $z_p = z_l + z_r$. In graphical model terms, for each parent in the tree, add a deterministic potential $g_p(z_p, z_l, z_r) = \mathbf{1}_{\{z_p = z_l + z_r\}}$ to the model. With these definitions, we can define the distribution q as,

$$q(\boldsymbol{y}, \boldsymbol{z}) \propto p(\boldsymbol{y}) \prod_{p \in \mathcal{P}} \mathbf{1}_{\{z_p = z_{l(p)} + z_{r(p)}\}}, \tag{6}$$

where $\mathcal{P}$ is the set of parent nodes in $\mathcal{T}$, and $l(p)$ and $r(p)$ are indices of the left and right children of z_p.

Define the set of leaf node descendants of z_p as $s_p = \{y_d \mid z_p \text{ is an ancestor of } y_d\}$. For any setting of $\boldsymbol{z}$ to have nonzero probability, it clearly must be the case that $z_p = \sum_{y_d \in s_p} y_d$. Thus, a high order potential over subset s_p can equivalently be represented as a unary potential on z_p. Expanding the definition of p from within Eq. (6), the computational benefit becomes clear, as we can rewrite all high order potentials as unary potentials:

$$\begin{aligned} q(\boldsymbol{y}, \boldsymbol{z}) &\propto \prod_{k | s_k \in \mathcal{S}} f_k\Big(\sum_{y_d \in s_k} y_d\Big) \prod_d \theta_d(y_d) \prod_{p \in \mathcal{P}} \mathbf{1}_{\{z_p = z_{l(p)} + z_{r(p)}\}} \\ &= \prod_{k | s_k \in \mathcal{S}} f_k(z_k) \prod_d \theta_d(y_d) \prod_{p \in \mathcal{P}} \mathbf{1}_{\{z_p = z_{l(p)} + z_{r(p)}\}} \end{aligned} \tag{7}$$

The following proposition justifies correctness and makes the relationship between p and q precise:

Proposition 1. *For all $\boldsymbol{y}$, $p(\boldsymbol{y}) = \sum_{\boldsymbol{z}} q(\boldsymbol{y}, \boldsymbol{z})$.*

Proof. There is exactly one joint setting of $\boldsymbol{z}$ with nonzero probability for each joint setting of $\boldsymbol{y}$. To see this, observe that given a setting of $\boldsymbol{y}$, the one and only setting of $\boldsymbol{z}$ that satisfies all the deterministic relationships is for each z_p to set $z_p = \sum_{y_d \in s_p} y_d$. The proposition then follows directly. □

5.2 Aligning Nested Subsets with $\mathcal{T}$

We have shown how adding auxiliary variables $\boldsymbol{z}$ enables us to convert high order cardinality potentials $f_k(\sum_{y_d \in s_k} y_d)$ into a unary potential on an individual z_k variable. However, this transformation only works if there is an internal z variable for each subset s_k that we wish to put a cardinality potential over. This restriction is what leads to the nested property that we require of sets of subsets $\mathcal{S}$. For any nested set of subsets $\mathcal{S}$, however, it is possible to construct a binary tree $\mathcal{T}$ such that an internal node in the tree is created for every subset s_k. A question for future work is whether a weaker condition than nestedness can be enforced while still guaranteeing efficient exact inference.

Finally, all that remains is to show how sum-product messages can be computed efficiently.

5.2.1 Upward Messages

Let $g(z_p, z_l, z_r)$ be a factor that enforces the deterministic relationship $z_p = z_l + z_r$ between parent z_p and children z_l and z_r. The upward message vectors that we need to compute take the following form:

$$\begin{aligned} m_{g,z_p}(z_p) &= \sum_{z_l=0}^{c_l} \sum_{z_r=0}^{c_r} g(z_p, z_l, z_r) m_{z_l,g}(z_l) m_{z_r,g}(z_r) \\ &= \sum_{z_l=z_p-z_r}^{c_l} m_{z_l,g}(z_l) m_{z_r,g}(z_p - z_l). \end{aligned}$$

Expressed in this way, it becomes clear that the entire message vector (the above quantity, for all values of $z_p \in \{0, \ldots, c_p\}$) can be computed as a 1D discrete convolution. Since 1D discrete convolutions of vectors of length N can be computed in $\mathcal{O}(N \log N)$ time using FFTs, these message vectors can be computed in $\mathcal{O}(c_p \log c_p)$ time.

5.2.2 Downward Messages

Let $g(z_p, z_l, z_r)$ be a factor that enforces the deterministic relationship $z_p = z_l + z_r$ between parent z_p and children z_l and z_r. The downward message vectors that we need to compute take the following form (assume w.l.o.g. that the message is to z_l):

$$m_{g,z_l}(z_l) = \sum_{z_r=0}^{c_r} m_{z_p,g}(z_l + z_r) m_{z_r,g}(z_r).$$

This also can be computed as a 1D discrete convolution, after reversing the message vector $m_{z_r,g}$.

5.2.3 Asymptotics

Assuming balanced binary trees, the algorithm runs in $\mathcal{O}(D \log^2 D)$ time. This can be seen by using the Master theorem to solve the recurrence $T(n) = 2T(n/2) + n \log n$. If binary trees are not balanced, the worst case runtime can be $O(D^2)$. The algorithm uses $\mathcal{O}(D \log D)$ space.

5.3 Minor Extensions

We argue that a main benefit of the auxiliary variable formulation is that it allows for several variants and elaborations. We discuss some of them in this section, to illustrate the range of other similar models models where learning and inference can be done tractably.

Drawing a Joint Sample. Given the auxiliary variable formulation, drawing a joint sample from a RC model is straightforward: pass messages inward to the root as before; compute the belief at the root, which is a distribution over global counts, and draw the value for the root variable from that distribution; now proceed outwards from the root towards the leaves. The one non-triviality is that values for the two children given a parent must be drawn simultaneously. To do this, first construct the belief at the trinary factor $f(z_p, z_l, z_r)$ conditioned on the value of z_p, which has been sampled already. Given z_p, there is a diagonal along the belief matrix corresponding to the values of z_l and z_r such that $z_l + z_r = z_p$. To draw the joint sample for z_l and z_r, normalize this diagonal to sum to 1, then draw a value from that distribution. This gives the values for z_l and z_r. Recurse downwards.

Other minor extensions appear in the Appendix.

5.4 Major Extension

Up until now, we have presented RC models as if the modeler who desires efficient exact inference must choose between standard pairwise trees or RC models. In fact, this is not necessary. It is possible to do exact inference in the following model in $\mathcal{O}(D \log^2 D)$ time:

$$p(\boldsymbol{y}) \propto \prod_{(d,d') \in \mathcal{E}} \theta_{dd'}(y_d, y_{d'}) \prod_{s_k \in \mathcal{S}(\mathcal{E})} f_k(\sum_{y_d \in s_k} y_d), \quad (8)$$

where $\mathcal{E}$ is an acyclic set of edges over variables $\boldsymbol{y}$, and $\mathcal{S}(\mathcal{E})$ is a set of nested subsets with subset structure that is "compatible" with $\mathcal{E}$. In other words, the modeler can choose the structure of either $\mathcal{E}$ or $\mathcal{S}(\mathcal{E})$ arbitrarily, and there is some non-degenerate choice of the other that allows for efficient inference, but not all combinations of trees and subset structures are compatible. Note that this structure cannot be represented by a standard Recursive Cardinality potential, because having e.g., edges (d, d') and (d', d'') would violate the nested subset structure requirement. The approach is similar to the base algorithm, but it involves constructing a junction tree that has separator sets involving one y variable and one z variable. We give details in the Appendix.

6 Experiments

In this section, we empirically explore the properties of the RC model, and demonstrate its usage in several interesting scenarios. Code will be made available implementing the convolution tree algorithm for computing marginals over $\boldsymbol{y}$ and $\boldsymbol{z}$, and for joint sampling.

6.1 Chain vs Tree as D grows

The version of the RC algorithm that we presented in Section 5 is the most efficient that we have discovered. There are, however, other approaches that are more efficient than $\mathcal{O}(D^3 \log D)$, but which are less efficient than the algorithm with the best asymptotic runtime. In this section, we compare the runtime of our algorithm (the "FFT Tree") against two baselines, both of which are improvements over the PL algorithm reviewed in Section 2.3.

The first baseline introduces auxiliary variables in a chain rather than tree structure. If we then follow the approach as in Section 5, this would lead to a $\mathcal{O}(D^2 \log D)$ algorithm. It turns out that in this case, messages can be computed via a summation of two arrays, so using FFTs is unnecessary. This yields a $\mathcal{O}(D^2)$ algorithm equivalent to that of Barlow and Heidtmann (1984), which we term the "Chain" algorithm. A drawback of this algorithm is that it also requires $\mathcal{O}(D^2)$ space. However, we note that if cardinalities greater than k are disallowed by the cardinality potential, then this algorithm can be made to run in $\mathcal{O}(Dk)$ time, and it is perhaps the best choice.

The second baseline is the fast algorithm where FFTs are not used to compute messages, and instead we use an efficient but brute-force computation of the 1D convolutions required inside the message computations. We refer to this as the "Tree" algorithm. Solving the recurrence $T(n) = 2T(n/2) + n^2$ using the Master theorem, we see that the runtime is again $\mathcal{O}(D^2)$. The space usage is $\mathcal{O}(D \log D)$.

We run these algorithms on problems of size up to 2^{19} variables with a single random cardinality potential. The runtimes are reported in Fig. 3. The Chain algorithm fails after $D = 16000$ due to memory limits. The Tree algorithm is faster than the chain algorithm for larger D, but its quadratic time usage causes it to become quite slow once D nears 100k. The FFT Tree algorithm behaves nearly linearly in practice, running on problems with half a million variables in less than 100 seconds. We note that in practice, for large values

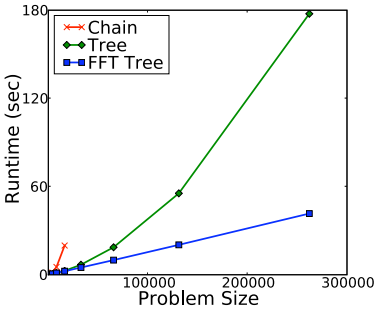
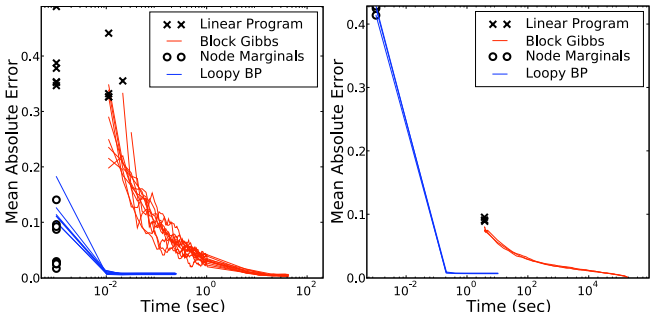


Figure 3: Runtimes of the different algorithms versus problem size. For very small D, the Chain algorithm is slightly faster, but it runs out of memory after around $D = 15000$, due to its quadratic memory usage. The FFT Tree algorithm runtime grows near linearly up through all experiments.



(a) 3x6 Grid (b) 20x40 Grid

Figure 4: Mean absolute error between inferred marginals and ground truth marginals.

of D, care must be taken to avoid numerical issues for certain settings of model parameters.

6.2 Generalized Bipartite Matching

We have variables $\boldsymbol{y} = \{y_{ij}\}_{(i,j)\in D_I \times D_J}$, where D_I is the number of rows, and D_J is the number of columns. Here, we consider the Gibbs distribution defined by the following energy function:

$$E(\boldsymbol{y}) = \sum_{ij} \theta_{ij} y_{ij} + \sum_i f_c(\sum_j y_{ij}) + \sum_j f_r(\sum_i y_{ij}),$$

where f_r and f_c are functions of row and column counts. Note that if a constraint is placed on each row, saying that exactly one binary variable in each column can be on, then this formulation can also be used to represent cardinality potentials for multilabel problems. One motivation for this model comes from the problem of paper-to-reviewer matching, where y_{ij} represents the event that paper i is matched to reviewer j. The row and column functions can then be used to enforce the constraints that each paper should be given to e.g., 3 to 4 reviewers, and each reviewer should be assigned e.g., 6 to 8 papers. Note that the energy function can clearly represent the bipartite matching problem (by constraining row and column counts to be exactly 1), so computing marginals is #P-complete. Our approach will be to perform approximate inference using loopy belief propagation (LBP), where messages are computed using the FFT tree algorithm.

We compare against several baselines on small and medium-sized problems where we constrain columns to have 1 or 2 variables labeled 1, and rows are constrained to have 2 or 3 variables labeled 1. The first, "Node Marginals" is a naive approximation that simply ignores the constraints and computes marginals under the factorized distribution consisting of just the node potentials. The second is to find the exact MAP solution to the problem using a linear program (LP) and assume that the marginals are 0 or 1. The final baseline, which we take as the ground truth for the intractable medium-sized problem, is to use a block Gibbs sampler where blocks are chosen as the four variables $y_{i_1,j_1}, y_{i_1,j_2}, y_{i_2,j_1}, y_{i_2,j_2}$ for some choice of rows i_1 and i_2 and columns j_1 and j_2. To find a valid initial configuration, we initialize the sampler with the LP solution. We attempted multiple runs using random parameter settings. The results in Fig. 4 show that LBP achieves a low bias in a relatively short amount of time. For small problems, the constraints do not always greatly influence the marginals, as exhibited by the low bias of the naive approach in some runs; however, they clearly influence the larger problems. Finally, the LP method tends to be slower than LBP, becomes relatively slower on larger problems, and exhibits significant bias. We have run our algorithm on larger problems, and it converges quickly even on e.g., 100x100 problems, but we are unable to measure accuracy, because accurate baselines are prohibitively slow.

It is worth mentioning that this is a hard problem, and designing sampling schemes for many cases is nontrivial. Indeed, the Gibbs sampler we used will not be ergodic for disjoint cardinality constraints (e.g. only 3 or 10). By contrast, our method is relatively fast, accurate, and applies to a wide variety of constraints.

6.3 Multiple Instance Learning

In multiple instance learning (MIL), we are given "bags" of instances, which are labeled as either "positive" (at least one instance in the bag is positive) or "negative" (all instances in the bag are negative). This can be framed as a problem of learning with weak labels, where the weak labels take the form of a cardinality potential over individual instance labels. Indeed, MIL models where bag labels are modeled as noisy-OR can be seen as exactly this formulation, using the

form of noisy-OR from Eq. (5). However, given the fast algorithms for Recursive Cardinality potentials developed in this work, it becomes tractable to assert other forms of the distribution over within-bag counts. In this section, we experiment with this alternative.

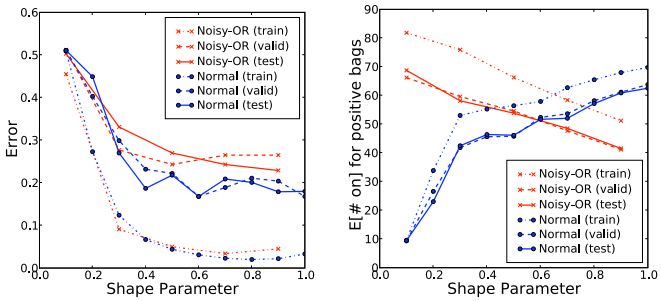
More formally, for variables $\boldsymbol{y}$ appearing in a bag labeled as $t = 0$ (negative) or $t = 1$ (positive), we can rewrite the likelihood of a label as $\sum_{\boldsymbol{y}} p(t, \boldsymbol{y})$, where

$$p(t, \boldsymbol{y}) \propto f^{(t)}(\sum_d y_d) \prod_d \exp\{\theta_d \cdot y_d\}, \qquad (9)$$

where $f^{(t)}$ is some cardinality function that imposes a preference on the number of binary variables turned on for bags labeled as t, and θ_d is a unary potential. All of the required quantities for learning can be computed in two calls to our algorithm (corresponding to $t = 0$ and $t = 1$).

The standard data set for evaluating MIL learners is the Musk dataset from Dietterich et al. (1997). In the data, bags molecules are labeled as to whether they are "musks." A molecule is considered to be a musk if any low energy conformations of the molecule is a musk. The bags in this data set correspond to molecules, and the instances are features of many different low energy conformations. We compared two methods on the musk1 version of the dataset: a standard Noisy-OR model, and a "Normal" model, where $f^{(0)}(c) = \exp\{-(\frac{c}{D})/2\sigma^2\}$ and $f^{(1)}(c) = \exp\{-(\mu - \frac{c}{D})/2\sigma^2\}$.

As is common in the MIL literature, we divide the data into 10 evenly-sized folds and run 10-fold cross-validation. We use 20% of bags for validation, and 10% for testing in each split. To set an $L1$ regularization parameter, and to choose the σ and ϵ parameters in the Normal and Noisy-OR models, respectively, we do a grid search and choose the setting that produced the best average validation error across folds. Fig. 5 (a) reports results showing how error varies as a function of the μ and λ parameters for Normal and Noisy-OR, respectively. We note that the musk1 dataset has only 92 instances, and the standard deviation of errors across validation folds was high, so these results are not statistically significant. However, we do see a trend where varying the f functions can affect performance, and that a Normal cardinality potential out-performs the standard Noisy-OR. For this problem, it appears that setting the μ parameter to be large is beneficial for learning, but in general this should be a parameter that is set via prior knowledge or cross validation. Also note that Gehler and Chapelle (2007) enforced a similar bias on bag counts within an SVM formulation, and their results also suggested that encouraging the model to turn on more binary variables within positive bags improved performance on the musk1 data. In Fig. 5 (b), we show how parameter values in the



(a) Error (b) # on in Positive Bags

Figure 5: Multiple instance learning results.

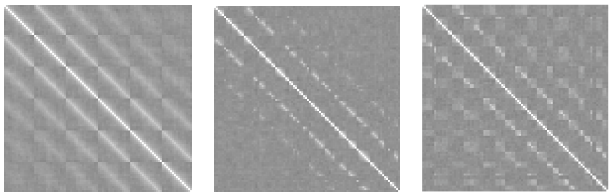
cardinality potential affects the expected number of instances labeled as "on" within positive bags. The trend in this plot shows that varying f does indeed affect how many instances within a positive bag that the model chooses to label as positive examples.

6.4 Generative Image Models

In this set of experiments, we attempt to distinguish between the representational abilities of pairwise tree-structured graphical models, and the RC model. We consider a synthetic dataset that has been sampled from an Ising model near the critical temperature resulting in images with highly correlated regions. The pairwise edges form a grid structure, and the resulting data consists of 1000 16×16 images. In order to compare the pairwise tree and the RC models we first learn their parameters on each dataset using nonlinear conjugate gradients to minimize the negative log-likelihood. After learning each model, we generate the same number of samples as the original dataset and use these to compare statistics. After learning, the log-likelihoods for the pairwise tree model was 145.22, while the log-likelihoods for the RC model was 146.78. We hypothesize that this difference is because the pairwise tree model is able to do a better job at directly capturing pairwise statistics, however the RC model is better at capturing long-range, higher-order statistics. We demonstrate this in our experiments below.

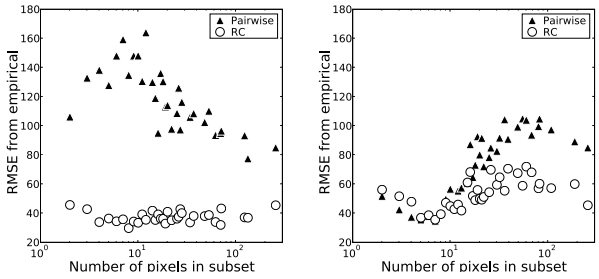
Heuristic choice of transformation structure. For each dataset, in order to determine the structure of the nested cardinality potential, we run hierarchical agglomerative clustering on images, where the similarity between pixels is determined by how often they take the same label. This gives us a full tree structure over the pixels, which we then directly use to give the nested subset structure. We leave a study of other ways for choosing trees to future work.

Pairwise statistics. We analyze the pairwise statistics of the learned distributions in comparison to the original datasets. Fig. 6 shows the pairwise correla-



(a) Actual data (b) Pairwise tree (c) RC

Figure 6: Zoomed pairwise correlation matrices from the Ising dataset.



(a) Adaptive clustering (b) Anti-clustering

Figure 7: Error in higher-order count statistics between the empirical and model distributions as a function of the number of variables in a subset.

tions between each pixel, with lighter shading corresponding to stronger correlation. The Ising model enforces strong correlation between neighbouring pixels, and this is reflected in the banded nature of the correlation matrix. Visually, we can see that the RC model does a better job of capturing long range correlations corresponding to the more faded bands away from the main diagonal.

Higher-order statistics. As a final comparison, we analyze the ability of both models to capture higher-order statistics in the data. Specifically, we test the ability of the RC model to capture higher-order count statistics over subsets that it does not directly encode. To do this, we first build a nested tree over the variables using hierarchical clustering. Using this tree, we look at the count distributions of subsets and compare the empirical and model distributions. In Fig. 7, we show the root mean squared error between the empirical and model distributions as a function of the number of variables in a subset for the Ising dataset. We compare both models on nested trees constructed from two heuristics: the first is the same scheme we use to parameterize the RC model, which we call adaptive clustering. For the second, we attempt to create a "worst case" clustering by taking the distance values used for the adaptive method and negating them before clustering; we call this scheme anti-clustering. We also tried random clusterings, however the results were not qualitatively different from the anti-clustering scheme. Our results show that the RC model does a better job than the pairwise tree model at capturing higher-order count statistics, even if it does not encode them directly. Particularly interesting is that for the adaptive tree, the pairwise model does significantly worse than the RC model. This is due to the confluence of two effects: first, the hierarchical clustering that determines the RC structure seeks to choose subsets of variables that are highly correlated to merge into a subset; second, the spanning tree is unable to model this "all-or-none" effect, and instead tends towards a binomial-like distribution with mode near the 50% full point. This leads to large errors.

7 Discussion

We have presented a new class of high order model where exact inference can be done very efficiently. This model can be used on its own, and it is already able to capture a mix of high and low order statistics that would be difficult for other tractable models. The model can also be used as a subcomponent within outer approximate inference procedures, allowing efficient approximate marginal inference for hard problems that few other approaches are able to tackle.

The general approach—of adding a set of auxiliary variables with highly structured interactions so as to guarantee efficient message passing inference—leads to efficient exact inference in other extended models such as tree structured models augmented with a cardinality potential, and we believe the approach to extend even beyond that.

Finally, we have presented a diverse set of applications and shown how many problems that are not naturally thought of in terms of cardinality potentials can be cast as such. We believe that the highly efficient algorithms discussed here will only increase the number of applications where cardinality potentials are useful.

References

Barlow, R. E., & Heidtmann, K. D. (1984). Computing k-out-of-n system reliability. *IEEE Transactions on Reliability*, *33*, 322-323.

Belfore, L. (1995). An O(n) log2(n) algorithm for computing the reliability of k-out-of-n:G and k-to-l-out-of-n:G systems. *IEEE Transactions on Reliability*, *44*(1).

Chen, S. X., & Liu, J. S. (1997). Statistical applications of the Poisson-Binomial and conditional Bernoulli distributions. *Statistica Sinica*, *7*(4).

de Salvo Braz, R., Amir, E., & Roth, D. (2005). Lifted

first-order probabilistic inference. In *In proceedings of ijcai-05, 19th international joint conference on artificial intelligence* (pp. 1319–1325). Morgan Kaufmann.

Dietterich, T. G., Lathrop, R. H., Lozano-Perez, T., & Pharmaceutical, A. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient belief propagation for early vision. In *CVPR* (pp. 261–268).

Fernandez, M., & Williams, S. (2010). Closed-form expression for the poisson-binomial probability density function. *IEEE Transactions on Aerospace Electronic Systems*, *46*, 803-81.

Fisher, M. E. (1961, Dec). Statistical mechanics of dimers on a plane lattice. *Phys. Rev.*, *124*, 1664–1672. Available from `http://link.aps.org/doi/10.1103/PhysRev.124.1664`

Gail, M. H., Lubin, J. H., & Rubinstein, L. V. (1981). Likelihood calculations for matched case-control studies and survival studies with tied death times. *Biometrika*.

Gallager, R. (1963). *Low-density parity-check codes*. MIT Press.

Ganchev, K., Graça, J., Gillenwater, J., & Taskar, B. (2010). Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, *11*, 2001-2049.

Gehler, P., & Chapelle, O. (2007, March). Deterministic annealing for multiple-instance learning. In Meila, M., & X. Shen (Eds.), *ICML* (p. 123-130).

Gould, S. (2011). Max-margin learning for lower linear envelope potentials in binary markov random fields. In *ICML*.

Gupta, R., Diwan, A., & Sarawagi, S. (2007). Efficient inference with cardinality-based clique potentials. In *ICML* (Vol. 227, p. 329-336).

Hong, Y. (2011). *On computing the distribution function for the sum of independent and non-identical random indicators.*

Huang, B., & Jebara, T. (2007). Loopy belief propagation for bipartite maximum weight b-matching. In *AISTATS*.

Kohli, P., Ladicky, L., & Torr, P. H. S. (2009). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, *82*(3), 302-324.

Marlin, B., Swersky, K., Chen, B., & Freitas, N. de. (2010). Inductive principles for restricted boltzmann machine learning. In *AISTATS*.

Mezuman, E., & Weiss, Y. (2012). Globally optimizing graph partitioning problems using message passing. In *AISTATS*.

Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., & Kaelbling, L. P. (2008). Lifted probabilistic inference with counting formulas. In D. Fox & C. P. Gomes (Eds.), *Aaai* (p. 1062-1068). AAAI Press.

Potetz, B., & Lee, T. S. (2008, Oct). Efficient belief propagation for higher order cliques using linear constraint nodes. *Computer Vision and Image Understanding*, *112*(1), 39–54.

Ramalingham, S., Kohli, P., Alahari, K., & Torr, P. (2008). Exact inference in multi-label CRFs with higher order cliques. In *CVPR*.

Régin, J.-C. (1996). Generalized arc consistency for global cardinality constraint. In *AAAI/IAAI* (p. 209-215).

Rother, C., Kohli, P., Feng, W., & Jia, J. (2009). Minimizing sparse higher order energy functions of discrete variables. In *CVPR* (p. 1382-1389).

Stobbe, P., & Krause, A. (2010). Efficient minimization of decomposable submodular functions. In *NIPS*.

Tarlow, D., Givoni, I., & Zemel, R. (2010). HOP-MAP: Efficient message passing for high order potentials. In *AISTATS*.

Vicente, S., Kolmogorov, V., & Rother, C. (2009). Joint optimization of segmentation and appearance models. In *ICCV*.

Zhao, P., Rocha, G., & Yu, B. (2006). Grouped and hierarchical model selection through composite absolute penalties. *Department of Statistics, UC Berkeley, Tech. Rep*, *703*.

Value Function Approximation in Noisy Environments Using Locally Smoothed Regularized Approximate Linear Programs

Gavin Taylor
Department of Computer Science
United States Naval Academy
Annapolis, MD 21402

Ronald Parr
Department of Computer Science
Duke University
Durham, NC 27708

Abstract

Recently, Petrik et al. demonstrated that L_1-Regularized Approximate Linear Programming (RALP) could produce value functions and policies which compared favorably to established linear value function approximation techniques like LSPI. RALP's success primarily stems from the ability to solve the feature selection and value function approximation steps simultaneously. RALP's performance guarantees become looser if sampled next states are used. For very noisy domains, RALP requires an accurate model rather than samples, which can be unrealistic in some practical scenarios. In this paper, we demonstrate this weakness, and then introduce Locally Smoothed L_1-Regularized Approximate Linear Programming (LS-RALP). We demonstrate that LS-RALP mitigates inaccuracies stemming from noise even without an accurate model. We show that, given some smoothness assumptions, as the number of samples increases, error from noise approaches zero, and provide experimental examples of LS-RALP's success on common reinforcement learning benchmark problems.

1 Introduction

Markov Decision Processes (MDPs) are applicable to a large number of real-world Artificial Intelligence problems. Unfortunately, solving large MDPs is computationally challenging, and it is generally accepted that such MDPs can only be solved approximately. This approximation is often done by relying on linear value function approximation, in which the value function is chosen from a low-dimensional vector space of features. Generally speaking, agent interactions with the environment are sampled, the vector space is defined through feature selection, a value function is selected from this space by weighting features, and finally a policy is extracted from the value function. Each of these steps contains difficult problems and is an area of active research. The long-term hope for the community is to handle each of these topics on increasingly difficult domains with decreasing requirements for expert guidance (providing a model, features, etc.) in application. This will open the door to wider application of reinforcement learning techniques.

Previously, Petrik et al. [12] introduced L_1-Regularized Approximate Linear Programming (RALP) to combine and automate the feature selection and feature weighting steps of this process. Because RALP addresses these steps in tandem, it is able to take advantage of extremely rich feature spaces without overfitting. However, RALP's hard constraints leave it especially vulnerable to errors when noisy, sampled next states are used. This is because each sampled next state generates a lower bound on the value of the originating state. Without a model, the LP will not use the expectation of these next state values but will instead use the max of the union of the discounted values of sampled next states. Thus, a single low probability transition can determine the value assigned to a state when samples are used in lieu of a model. This inability to handle noisy samples is a key limitation of LP based approaches.

In this paper, we present Locally Smoothed L_1-Regularized Approximate Linear Programming (LS-RALP), in which a smoothing function is applied to the constraints of RALP. We show this smoothing function reduces the effects of noise, without the requirement of a model. The result is a technique which accepts samples and a large candidate feature set, and then automatically chooses features and approximates the value function, requiring neither a model, nor human-guided feature engineering.

Standard ALP approaches produce value functions and require a model at policy execution time to determine the optimal action with respect the value function returned by the LP. We extend LS-RALP to produce Q-functions, rather than value functions, thereby removing the dependence on a model from both value function learning and policy execution.

After introducing notation, this paper first examines RALP in Section 3 to demonstrate the effect of noise on solution quality. Section 4 then introduces LS-RALP, and explains the approach. In Section 5, we prove that as samples are added, the error from noise in LS-RALP's value function approximation approaches zero. Finally, in Section 6, we empirically demonstrate LS-RALP's success in high-noise environments, even with a small number of samples. Finally, in Section 7, we make our concluding remarks.

2 Framework and Notation

In this section, we formally define Markov decision processes and linear value function approximation. A *Markov decision process* (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, p, R, \gamma)$, where $\mathcal{S}$ is a measurable subset of Euclidean space, and $\mathcal{A}$ is a finite set of actions. p is a transition kernel, where $p(s'|s,a)$ represents the conditional density for next states. The function $R : \mathcal{S} \mapsto \Re$ is a measurable reward function, and γ is a discount factor.

We are concerned with finding a value function V that maps each state $s \in \mathcal{S}$ to the expected total γ-discounted reward for the process. Value functions can be useful in creating or analyzing a measurable policy function $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$ such that for all $s \in \mathcal{S}$, $\sum_{a \in \mathcal{A}} \pi(s,a) = 1$. The transition and reward functions for a given policy are denoted by P_π and R_π. We denote the Bellman operator for a given policy as $\bar{T}_\pi$, and the max Bellman operator simply as $\bar{T}$. That is, for some state $s \in \mathcal{S}$:

$$\bar{T}_\pi V(s) = R(s) + \gamma \int_{\mathcal{S}} P(s'|s,\pi(s)) V(s') ds'$$

$$\bar{T} V(s) = \max_{\pi \in \Pi} \bar{T}_\pi V(s).$$

We additionally denote the Bellman operator for a policy which always selects action a as

$$\bar{T}_a V(s) = R(s) + \int_{\mathcal{S}} P(s'|s,a) V(s') ds'.$$

If $\mathcal{S}$ is finite, the above Bellman operators can be expressed for all states using matrix notation, in which V and R are vectors of values and rewards for each state, respectively, and P is a matrix containing probabilities of transitioning between each pair of states:

$$\bar{T}_\pi V = R_\pi + \gamma P_\pi V$$

$$\bar{T} V = \max_{\pi \in \Pi} \bar{T}_\pi V$$

$$\bar{T}_a V = R + \gamma P_a V$$

The optimal value function V^* satisfies $\bar{T} V^* = V^*$.

Because $\bar{T}$ computes an exact expectation over next states, we refer to $\bar{T}$ as the Bellman operator *with expectation*. This requires *sampling with expectation*, defined as $\bar{\Sigma} \subseteq \{(s, a, \mathbb{E}[R(s)]), p(\cdot|s,a) | s \in \mathcal{S}, a \in \mathcal{A}\}$. This kind of sampling assumes that the complete density over next states is provided with every sample, a rather unrealistic assumption. It is therefore usually convenient to use *one-step simple samples* $\Sigma \subseteq \{(s,a,r,s') | s, s' \in \mathcal{S}, a \in \mathcal{A}\}$, where s' is a single draw from $p(s'|s,a)$. An individual (s,a,r,s') sample from sets $\bar{\Sigma}$ and Σ will be denoted $\bar{\sigma}$ and σ, respectively. An individual element of a sample σ will be denoted with superscripts; that is, the s component of a $\sigma = (s,a,r,s')$ sample will be denoted σ^s. Simple samples have an associated *sampled* Bellman operator T_σ, where $T_\sigma V(\sigma^s) = \sigma^r + \gamma V(\sigma^{s'})$.

We will also define the optimal Q-function $Q^*(s,a)$ as the value of taking action a at state s, and following the optimal policy thereafter. More precisely, $Q^*(s,a) = R(s,a) + \gamma \int_{\mathcal{S}} P(s'|s,a) V^*(s') ds'$.

We focus on *linear value function approximation* for discounted infinite-horizon problems, in which the value function is represented as a linear combination of *nonlinear basis functions (vectors)* which are assumed to be measurable on S. For each state s, we define a vector $\Phi(s)$ of features. The rows of the basis matrix Φ correspond to $\Phi(s)$, and the approximation space is generated by the columns of the matrix. That is, the basis matrix Φ, and the value function V are represented as:

$$\Phi = \begin{pmatrix} - & \Phi(s_1) & - \\ & \vdots & \end{pmatrix} \qquad V = \Phi w.$$

This form of linear representation allows for the calculation of an approximate value function in a lower-dimensional space, which provides significant computational benefits over using a larger or complete basis; if the number of features is small, this framework can also guard against overfitting any noise in the samples though small features sets alone cannot protect ALP methods from making errors due to noise.

3 L_1-Regularized Approximate Linear Programming

This section includes our discussion of L_1-Regularized Approximate Linear Programming. Subsection 3.1 will review the salient points of the method, while Subsection 3.2 will discuss the weaknesses of the method and our motivation for improvement.

3.1 RALP Background

RALP was introduced by Petrik et al. [12] to improve the robustness and approximation quality of Approximate Linear Programming (ALP) [13, 1]. For a given set of samples with expectation $\bar{\Sigma}$, feature set Φ, and regularization pa-

rameter ψ, RALP calculates a weighting vector w by solving the following linear program:

$$\begin{aligned} \min_{w} \quad & \rho^T \Phi w \\ \text{s.t.} \quad & \bar{T}_{\sigma^a} \Phi(\sigma^s) w \leq \Phi(\sigma^s) w \ \ \forall \sigma \in \Sigma \\ & \|w\|_{1,e} \leq \psi, \end{aligned} \tag{1}$$

where ρ is a distribution over initial states, and $\|w\|_{1,e} = \sum_i |e(i)w(i)|$. It is generally assumed that ρ is a constant vector and $e = \mathbf{1}_{-1}$, which is a vector of all ones but for the position corresponding to the constant feature, where $e(i) = 0$.

Adding L_1 regularization to the constraints of the ALP achieved several desirable effects. Regularization ensured bounded weights in general and smoothed the value function. This prevented missing constraints due to unsampled states from leading to unbounded or unreasonably low value functions. By using L_1 regularization in particular, the well-known sparsity effect could be used to achieve automated feature selection from an extremely rich Φ. Finally, the sparsity in the solution could lead to fast solution times for the LP even when the number of features is large because the number of active constraints in the LP would still be small.

As a demonstration of the sparsity effect, RALP was run on the upright pendulum domain (explained in detail in Section 6) using a set of features for each action consisting of three Gaussian kernels centered around each sampled state, one polynomial kernel based around each sampled state, and a series of monomials calculated on the sampled states. On one randomly-selected run of 50 sample trajectories (a trajectory consists of states reached as a result of randomly selected actions until the pendulum falls over), to represent the Q-values of the first action RALP selected zero of the narrowest Gaussian kernels, nine of the intermediate-width Gaussian kernels, five of the widest Gaussian kernels, five of the polynomial kernels, and zero of the monomials. For the second action, the Q value was represented with zero narrow, seven intermediate, and four wide Gaussian kernels, six polynomial kernels, and one monomial. For the final action, these numbers were one, four, four, seven, and zero. Selecting this same mixture of features by hand would be extremely unlikely.

Petrik et al. [12] demonstrated a guarantee that the error on the RALP approximation will be small, as will be further discussed in the next subsection. Additionally, in experiments, policies based on RALP approximations outperformed policies generated by Least-Squares Policy Iteration [9], even with very small amounts of data.

RALP is not the only approach to leverage L_1 regularization in value function approximation; others include LARS-TD [7], and LC-MPI [6], which both attempt to calculate the fixed point of the L_1-penalized LSTD problem. Unlike RALP, this solution is not guaranteed to exist when samples are drawn off-policy. These methods have the advantage of handling noise more gracefully than LP based approaches, but require on-policy sampling for reliable performance. On-policy sampling is often a challenging requirement for real-life applications.

3.2 RALP Weaknesses

The RALP constraints in LP 1 are clearly difficult to acquire – samples with expectation require a model that provides a distribution over next states, or access to a simulator that can be easily reset to the same states many times to compute an empirical next state distribution for each constraint.

Petrik et al. [12] previously presented approximation bounds for three constructions of RALP, depending upon the sampling regime. In the *Full ALP*, it is assumed every possible state-action pair is sampled with expectation. The *Sampled ALP* contains constraints for only a subset of state-action pairs sampled with expectation. The third form, the *Estimated ALP*, also has constraints defined on sampled state-action pairs, but of the form

$$\Phi(\sigma^s) w \geq T_\sigma \Phi(\sigma^s) w \ \ \forall \sigma \in \Sigma.$$

Note that the Estimated ALP uses one-step simple samples and the sampled Bellman operator.

These three forms are presented in order of decreasing accuracy, but increasing applicability. The Full ALP requires a manageably small and discrete state space, as well as a completely accurate model of the reward and transition dynamics of the system. The Sampled ALP still requires the model, but allows for constraint sampling; this introduces constraint violation error (denoted ϵ_p), as the approximation may violate unsampled constraints. Finally, the Estimated ALP requires neither the model nor experience at every state; because replacing an accurate model with a sample is likely to be imprecise in a noisy domain, this introduces model error (denoted ϵ_s). However, this is the most realistic sampling scenario for learning problems that involve physical systems.

For reference, and to understand the effect of model error, we reproduce the RALP error bounds for the Sampled and Estimated ALP here. We denote the solution of the Sampled ALP $\bar{V}$, and the solution of the Estimated ALP V. ϵ_c is error from states that do not appear in the objective function because they were not samples. Since ρ is often chosen fairly arbitrarily, we can assume the unsampled states have 0 weight and that $\epsilon_c = 0$.

$$\|\bar{V} - V^*\|_{1,\rho} \leq \epsilon + 2\epsilon_c(\psi) + 2\frac{\epsilon_p(\psi)}{1-\gamma} \tag{2}$$

$$\|V - V^*\|_{1,\rho} \leq \epsilon + 2\epsilon_c(\psi) + \frac{3\epsilon_s(\psi) + 2\epsilon_p(\psi)}{1-\gamma} \tag{3}$$

To understand the source of model error ϵ_s, consider a case where a state-action pair is sampled twice. Further suppose that in one of these samples, the agent transitions to a high-value state; in the other, it transitions to a lower-valued state. The resulting value function is constrained by the high-valued sample, while the constraint related to the low-valued experience remains loose; that sample might as well have not occurred. The approximate value at that state will be inaccurately high. This scenario is depicted in Figure 1.

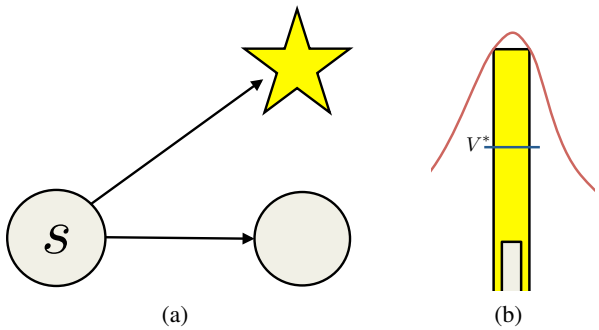


Figure 1: Subfigure 1(a) illustrates two samples beginning at state s; one transitions to the low-valued circle state, while the other transitions to the high-valued star state. Subfigure 1(b) illustrates the resulting constraints and value function approximation. V^* is the actual value of this state, but the red line representing the approximate function is constrained by the max of the two constraints. Under realistic sampling constraints, it is more likely this scenario would occur with two samples drawn from very nearly the same state, rather than both originating from the same sample s.

The inability of LP approaches to value function approximation to handle noisy samples is well known. One proposed solution is Smoothed Approximate Linear Programming (SALP), in which the LP is given a budget with which to violate constraints [2]. Note, however, that there is nothing connecting the constraint violation budget to the problem dynamics unless additional information is provided. In Figure 1, for example, SALP could fortuitously choose the right amount of constraint violation to mix between the next state values for state s in the proportions dictated by the model. However it is also possible for SALP to yield exactly the same value for this state as ordinary ALP, having spent its constraint violation budget on some other state which might not even be noisy.

4 Locally Smoothed L_1-Regularized Approximate Linear Programming

To address the weaknesses introduced in Subsection 3.2, we introduce Locally Smoothed L_1-Regularized Approximate Linear Programming (LS-RALP). The concept behind LS-RALP is simple; by averaging between nearby, similar samples, we can smooth out the effects of noise and produce a more accurate approximation. We define "nearby" to mean two things: one, the states should be close to each other in the state space, and two, the actions taken should be identical. The goal is to achieve the performance of the Sampled ALP with the more realistic sampling assumptions of the Estimated ALP.

We now introduce the smoothing function $k_b(\sigma, \sigma_i)$. A smoothing function spreads the contribution of each sampled data point σ_i over the local neighborhood of data point σ; the size of the neighborhood is defined by the bandwidth parameter b. If $\sigma_i^a \neq \sigma^a$, the function returns zero. We also stipulate that $\sum_{\sigma_i \in \Sigma} k_b(\sigma, \sigma_i) = 1$ for all σ, and that $k_b(\sigma, \sigma_i) \geq 0$ for all σ, σ_i. Smoothing functions of this sort are commonly used to perform nonparametric regression; as sampled data are added, the bandwidth parameter is tuned to shrink the neighborhood, to allow for a balance between the variance from having too few samples in the neighborhood and the bias from including increasingly irrelevant samples which are farther away.

Observation 4.1 *Consider the regression problem of estimating the function f of the response variable $y = f(x) + N(x)$, given n observations $(x_i, y_i)(i = 1, \cdots, n)$, where $N(x)$ is a noise function. Assume the samples x_i are drawn from a uniform distribution over a compact set G, that $f(x)$ is continuously differentiable, and that $N(x)$ satisfies certain regularity conditions. There exists a smoothing function $k_b(x, x_i)$ and associated bandwidth function $b(n)$ such that $\forall x \in G$, as $n \to \infty, \sum_i k_{b(n)}(x, x_i) y_i \to f(x)$, almost surely.*

Smoothing functions that suffice to provide the consistency guarantees in the above observation include kernel estimators [3] and k-nearest-neighbor averagers [4]. Similar results exist for cases without uniform sampling [5, 3].

The regularity conditions on the noise function depend upon the type of estimator and allow for a variety of noise models [3, 4]. In some of these cases, other assumptions in Observation 4.1 can be weakened or discounted altogether.

Literature on nonparametric regression also contains extensive theory on optimal shrinkage rates for the bandwidth parameter; in practice, however, for a given number of samples, this is commonly done by cross-validation.

We now modify our LP to include our smoothing function:

$$\begin{aligned} \min_{w} \quad & \rho^T \Phi w \\ \text{s.t.} \quad & \textstyle\sum_{\sigma_i \in \Sigma} k_b(\sigma, \sigma_i) T_{\sigma^a} \Phi(\sigma_i^s) w \leq \Phi(\sigma_i^s) w \;\; \forall \sigma \in \Sigma \\ & \|w\|_{1,e} \leq \psi. \end{aligned} \tag{4}$$

In this formulation, we are using our smoothing function to estimate the constraint with expectation $\bar{T}\Phi(\sigma^s)w$ by

smoothing across our easily obtained $T_\sigma\Phi(\sigma^s)w$.

Note that unsmoothed RALP is a special case of LS-RALP, where the bandwidth of the smoothing function is shrunk until the function is a delta function. Therefore, LS-RALP, with correct bandwidth choice, can always do at least as well as RALP.

We note that smoothing functions have been applied to reinforcement learning before, in particular by Ormoneit and Sen [11], whose work was later extended to policy iteration by Ma and Powell [10]. In this formulation, a kernel estimator was used to approximate the value function; this differs significantly from our approach of using a smoother to calculate weights more accurately for a parametric approximation. Additionally, because their approximation simply averages across states and does not use features, it will tend to have a poor approximation for novel states unless the space is very densely sampled. More recently, and independently from our work, Kroemer and Peters [8] proposed an approach to policy evaluation that first used kernel density estimation to approximate the dynamics and then solved for the value function of resulting approximate system using dynamic programming.

5 Convergence Proof

The goal of this section is to demonstrate that the application of a smoothing function to the constraints of the Estimated ALP will mitigate the effects of noise. In particular, we show that as the number of samples approaches infinity, the LS-RALP solution using one-step samples approaches the RALP solution using samples with expectation.

We begin by providing an explicit definition of ϵ_s that is consistent with the usage in Petrik et al. [12]. For a set of samples Σ,

$$\epsilon_s = \max_{\sigma\in\Sigma} \left|\bar{T}\Phi(\sigma^s)w - T_\sigma\Phi(\sigma^s)w\right|. \tag{5}$$

We now introduce and prove our theorem. We denote the number of samples $|\Sigma|$ as n.

Theorem 5.1 *If the reward function, transition function, and features are continuously differentiable, samples are drawn uniformly from a compact state space, and the noise models of the reward and transition functions satisfy the conditions of Observation 4.1 that would make kernel regression converge, then there exists a bandwidth function $b(n)$ such that as $n \to \infty$, the LS-RALP solution using one-step samples approaches the Sampled RALP solution for all states in the state space.*

We begin by restating the definition of ϵ_s:

$$\epsilon_s = \max_{\sigma\in\Sigma} \left|\bar{T}\Phi(\sigma^s)w - T_\sigma\Phi(\sigma^s)w\right|.$$

We then expand upon the portion within the max operator for some arbitrary σ:

$$\begin{aligned}&\bar{T}\Phi(\sigma^s)w - T_\sigma\Phi(\sigma^s)w\\ &= \left[R(\sigma^s) + \gamma\int_{\mathcal{S}} P(ds'|\sigma^s,\sigma^a)\Phi(s')w\right] - \left[\sigma^r + \gamma\Phi(\sigma^{s'})w\right]\end{aligned}$$

We now introduce the smoothing function so as to match the constraints of LS-RALP; that is, we replace $T_\sigma\Phi(\sigma^s)w$ with $\sum_{\sigma_i\in\Sigma} k_b(\sigma,\sigma_i)T_\sigma\Phi(\sigma_i^s)w$.

ϵ_s remains the maximum difference between the ideal constraint $\bar{T}\Phi(\sigma^s)w$ and our existing constraints, which are now $\sum_{\sigma_i\in\Sigma} k_b(\sigma,\sigma_i)T_\sigma\Phi(\sigma_i^s)w$.

$$\begin{aligned}&\bar{T}\Phi(\sigma^s)w - \sum_{\sigma_i\in\Sigma} k_b(\sigma,\sigma_i)T_\sigma\Phi(\sigma_i^s)w\\ &= \left[R(\sigma^s) + \gamma\int_{\mathcal{S}} P(ds'|\sigma^s,\sigma^a)\Phi(s')w\right]\\ &\quad - \sum_{\sigma_i\in\Sigma} k_b(\sigma,\sigma_i)\left[\sigma_i^r + \gamma\Phi(\sigma_i^{s'})w\right]\\ &= R(\sigma^s) - \sum_{\sigma_i\in\Sigma} k_b(\sigma,\sigma_i)\sigma_i^r\\ &\quad + \gamma\int_{\mathcal{S}} P(ds'|\sigma^s,\sigma^a)\Phi(s')w - \gamma\sum_{\sigma_i\in\Sigma} k_b(\sigma,\sigma_i)\Phi(\sigma_i^{s'})w\end{aligned}$$

The smoothing function is performing regression on all observed data, rather than just the single, possibly-noisy sample. The first two terms of the above equation are the difference between the expected reward and the predicted reward, while the second two are the difference between the expected next value and the predicted next value. The more accurate our regression on these two terms, the smaller ϵ_s will be.

It follows from Observation 4.1 that as n increases, if $R(s), P(s'|s,a)$, and $\Phi(s)$ are all continuously differentiable with respect to the state space, and if samples are drawn uniformly from a compact set, and the noise model of the reward and transition functions is one of several allowed noise models, there exists a bandwidth shrinkage function $b(n)$ such that $\sum_{\sigma_i\in\Sigma} k_b(\sigma,\sigma_i)T_\sigma\Phi(\sigma_i^s)w$ will approach $\bar{T}\Phi(\sigma^s)w$ almost surely, and ϵ_s will approach 0 almost surely.

When $\epsilon_s = 0$, the smoothed constraints from one-step samples of LS-RALP are equivalent to the constraints with expectation of the Sampled RALP. ∎

The assumptions on $R(s), P(s'|s,a)$, and $\Phi(s)$ required for our proof may not always be realistic, and are stronger than the bounded change assumptions made by RALP. However, we will demonstrate in the next section that meeting these assumptions is not necessary for the method to be effective.

This theorem shows that sample noise, the weak point of all ALP algorithms, can be addressed by smoothing in a principled manner. Even though these results address primarily

the limiting case of $n \to \infty$, our experiments demonstrate that in practice we can obtain vastly improved value functions even with very few samples.

6 Experiments

In this section, we apply LS-RALP to noisy versions of common reinforcement learning benchmark problems to demonstrate the advantage of smoothing. We will use two domains, the inverted pendulum [15] and the mountain-car [14]. While we have proved that LS-RALP will improve upon the value function approximation of RALP as the number of samples grows large, our experiments demonstrate that there is significant improvement even with a relatively small number of samples.

In the mountain-car domain, we approximate the value function, and then apply the resulting greedy policy. We generate the policy via the use of a model; at each state, we evaluate each candidate action, and calculate the approximate value at the resulting state. The action which resulted in the highest value is then chosen in the trajectory.

In the pendulum problem, we extend the method to the approximation of Q-functions, and again apply the resulting greedy policy. When using Q-functions, the model is unnecessary; the action with the highest Q-value is chosen.

6.1 Mountain Car

In the mountain car problem, an underpowered car must climb a hill by gaining momentum. The state space is two-dimensional (position and velocity), the action space is three-dimensional (push, pull, or coast), and the discount factor is 0.99. Traditionally, a reward of -1 is given for any state that is not in the goal region, and a reward of 0 is given for any state that is. For this problem we applied Gaussian noise with standard deviation .5 to the reward to demonstrate the ability of LS-RALP to handle noisy rewards. This is a very large amount of noise that greatly increases the difficulty of the problem.

Features are the distributed radial basis functions suggested by Kolter and Ng [7]; however, because LSPI with LARS-TD requires a distinct set of basis functions for each action, while RALP methods do not, our dictionary is one-third of the size. For sake of a baseline comparison, we note that in the deterministic version of this problem, we repeated experiments by Kolter and Ng [7] and Johns et al. [6] on LARS-TD and LC-MPI, respectively and RALP outperformed both.

Sample trajectories started at a random state, and were allowed to run with a random policy for 20 steps, or until the goal state was reached. After the value function was calculated, it was tested by placing the car in a random state; the policy then had 500 steps for the car to reach the goal state. Unlike the paper which introduced RALP [12], only one action at each state was sampled.

Figure 2 shows the success rate over 50 trials for both RALP and LS-RALP. The regularization parameter for the RALP methods was 1.4, and the smoothing function for LS-RALP was a multivariate Gaussian kernel with a standard deviation spanning roughly 1/85 of the state space in both dimensions. These parameters were chosen using cross validation.

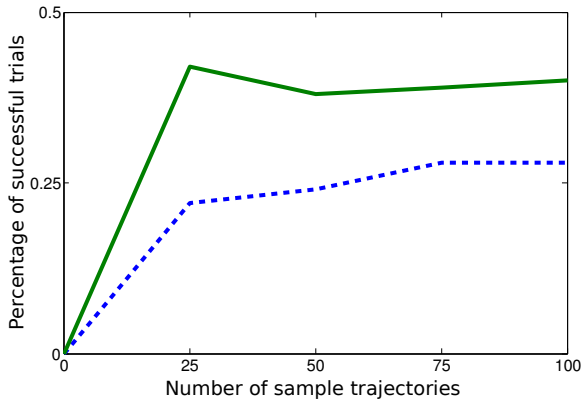


Figure 2: Percentage of successful trials vs. number of sample trajectories for the mountain-car domain with value functions approximated by RALP (blue, dotted) and LS-RALP (green, solid).

6.2 Inverted Pendulum

The focus of this paper is value function approximation, but our theory extends trivially to Q-function approximation. We demonstrate this by using RALP and LS-RALP to approximate Q-functions; this completely removes the need for a model in policy generation.

In the inverted pendulum problem the agent tries to balance a stick vertically on a platform by applying force to the platform. The state space is the angle and angular velocity of the stick, and the action space contains three actions: a push on the table of 50N, a pull on the table of 50N, and no action. The discount factor is 0.9. The goal is to keep the pendulum upright for 3000 steps. The noise in the pendulum problem is applied as Gaussian noise of standard deviation 10 on the force applied to the table. This is significantly greater than the noise traditionally applied in this problem, resulting in much worse performance even by the best controller we could produce.

We compare RALP and LS-RALP, using the same set of features; Gaussian kernels of three different widths, a polynomial kernel of degree 3, $s[1], s[2], s[1] * s[2], s[1]^2, s[2]^2$, and a constant feature, where $s[1]$ is the angle, and $s[2]$ is the angular velocity. For a number of samples n, this was therefore a total of $4n + 6$ features for each action. We also

compare against LSPI, using the radial basis functions used by Lagoudakis and Parr [9], and LARS-TD wrapped with policy iteration, using the same features used by RALP and LS-RALP.

Sampling was done by running 50 trajectories under a random policy, until the pendulum fell over (around six steps). For the LP methods, the regularization parameter was set to 4000, and for LS-RALP, the smoothing function was a multivariate Gaussian kernel. These parameters were chosen by cross validation. 50 trials were run for each method, and the number of steps until failure averaged; if a trial reached 3000 steps, it was terminated at that point. As with the mountain car experiments, only one action for each sampled state was included. Results are presented in Figure 3.

Notice that while LSPI was competitive with the Q-functions approximated with RALP, the addition of the smoother resulted in superior performance even with very little data.

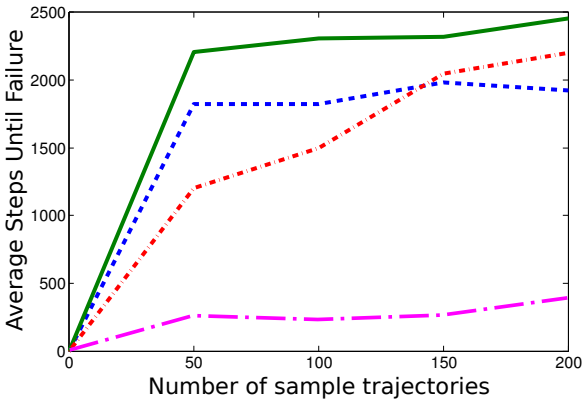


Figure 3: Average number of steps until failure vs. number of sample trajectories for the pendulum domain with Q-functions approximated with LARS-TD with policy iteration (violet, hashed), LSPI (red, hashed), RALP (blue [dark gray], dotted), and LS-RALP (green, solid).

7 Conclusion

In this paper, we have demonstrated the difficulties RALP can have in using noisy, sampled next states, and we have shown that the introduction of a smoothing function can mitigate these effects. Furthermore, we proved that as the amount of data is increased, error from inaccurate samples approaches zero. However, our experiments show drastic improvement with only a small number of samples.

In fairness, we also point out some limitations of our approach: LS-RALP makes stronger smoothness assumptions about the model and features than does RALP (continuous differentiability vs. the Lipschitz continuity necessary for the RALP bounds). LS-RALP also requires an additional parameter that must be tuned - the kernel bandwidth. We note, however, that the RALP work provides some guidance on choosing the regularization parameter and that there is a rich body of work in non-parametric regression that may give practical guidance on the selection of the bandwidth parameter.

The nonparametric regression literature notes that the bandwidth of a smoother can be adjusted not just for the total number of data points, but also for the density of data available in the particular neighborhood of interest [5]. The addition of this type of adaptiveness would greatly increase accuracy in areas of unusual density.

There are further opportunities to make better use of smoothing. First, we note that while smoothing the entire constraint has proven to be extremely useful, it is inevitable that applying two separate smoothing parameters to the reward and transition components would be even more helpful. For example, in the mountain-car experiments of Subsection 6.1, only the reward was noisy; applying smoothing to the reward, and not to the transition dynamics, would improve the approximation. The introduction of an automated technique of setting bandwidth parameters on two separate smoothing functions would likely result in more accurate approximations with less data.

Additionally, while the experiments done for this paper were extremely quick, this was due to the small number of samples needed. For larger, more complex domains, the LP can get large and may take a long time to solve; while LP solvers are generally thought of as efficient, identification of constraints doomed to be loose or features which will not be selected could cause a drastic improvement in the speed of this method.

Acknowledgments

This work was supported in part by NSF IIS-0713435 and NSF IIS-1147641, as well as the Naval Academy Research Council. Opinions, findings, conclusions or recommendations herein are those of the authors and not necessarily those of the sponsors.

References

[1] Daniela Pucci de Farias and Benjamin Van Roy. The Linear Programming Approach to Approximate Dynamic Programming. *Operations Research*, 2003.

[2] Vijay V. Desai, Vivek F. Farias, and Ciamac C. Moallemi. A smoothed approximate linear program. In *Advances in Neural Information Processing Systems*, volume 22, 2009.

[3] Luc P. Devroye. The Uniform Convergence of the Nadaraya-Watson Regression Function Estimate. *The Canadian Journal of Statistics*, 6(2):179–191, 1978.

[4] Luc P. Devroye. The uniform convergence of nearest neighbor regression function estimators and their application in optimization. *IEEE Transactions on Information Theory*, 24(2):142 – 151, Mar 1978.

[5] Christine Jennen-Steinmetz and Theo Gasser. A unifying approach to nonparametric regression estimation. *Journal of the American Statistical Association*, 83(404):1084–1089, 1988.

[6] Jeffrey Johns, Christopher Painter-Wakefield, and Ronald Parr. Linear complementarity for regularized policy evaluation and improvement. In *Advances in Neural Information Processing Systems 23*, pages 1009–1017, 2010.

[7] J. Zico Kolter and Andrew Ng. Regularization and Feature Selection in Least-Squares Temporal Difference Learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 521–528, Montreal, Canada, June 2009. Omnipress.

[8] O.B. Kroemer and J. Peters. A non-parametric approach to dynamic programming. In *Advances in Neural Information Processing Systems*, volume 23, 2011.

[9] Michail G. Lagoudakis and Ronald Parr. Least-Squares Policy Iteration. *The Journal of Machine Learning Research*, 2003.

[10] Jun Ma and Warren B. Powell. Convergence Analysis of Kernel-based On-Policy Approximate Policy Iteration Algorithms for Markov Decision Processes with Continuous, Multidimensional States and Actions. *Submitted to IEEE Transactions on Automatic Control*, 2010.

[11] D. Ormoneit and Ś. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2):161–178, 2002.

[12] Marek Petrik, Gavin Taylor, Ronald Parr, and Shlomo Zilberstein. Feature selection using regularization in approximate linear programs for markov decision processes. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[13] Paul J. Schweitzer and Abraham Seidmann. Generalized Polynomial Approximations in Markovian Decision Processes. *Journal of mathematical analysis and applications*, 110(6):568–582, 1985.

[14] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: an Introduction*. MIT Press, 1998.

[15] Hua O. Wang, Kazuo Tanaka, and Michael F. Griffin. An Approach to Fuzzy Control of Nonlinear Systems: Stability and Design Issues. *IEEE Transactions on Fuzzy Systems*, 4(1):14–23, 1996.

Factorized Multi-Modal Topic Model

Seppo Virtanen[1], **Yangqing Jia**[2], **Arto Klami**[1], **Trevor Darrell**[2]
[1]Helsinki Institute for Information Technology HIIT
Department of Information and Compute Science, Aalto University
[2]UC Berkeley EECS and ICSI

Abstract

Multi-modal data collections, such as corpora of paired images and text snippets, require analysis methods beyond single-view component and topic models. For continuous observations the current dominant approach is based on extensions of canonical correlation analysis, factorizing the variation into components shared by the different modalities and those private to each of them. For count data, multiple variants of topic models attempting to tie the modalities together have been presented. All of these, however, lack the ability to learn components private to one modality, and consequently will try to force dependencies even between minimally correlating modalities. In this work we combine the two approaches by presenting a novel HDP-based topic model that automatically learns both shared and private topics. The model is shown to be especially useful for querying the contents of one domain given samples of the other.

1 INTRODUCTION

Analysis of objects represented by multiple modalities has been an active research direction over the past few years. If the analysis of a single modality is characterized as learning some sort of components that describe the data, the task in analysis of multiple modalities can be summarized as learning components that describe both the variation within each modality but also the variation shared between them (Klami and Kaski, 2008; Jia et al., 2010). The fundamental problem is in learning how to correctly factorize the variation into the shared and private components, so that the components can be intuitively interpreted. For continuous vector-valued samples the problem can be solved efficiently by a structural sparsity assumption (Jia et al., 2010; Virtanen et al., 2011), resulting in an extension of canonical correlation analysis (CCA) that models not only the correlations but also components private to each modality.

One prototypical example of multi-modal analysis is that of modeling collections of images and associated text snippets, such as captions or contents of a web page. When both text and image content can naturally be represented with bag of words -type vectors, the assumptions made by the above methods fail. Instead, such count data calls for topic models such as latent Dirichlet allocation (LDA): several extensions of LDA have been presented for multi-modal setups, including Blei and Jordan (2003); Mimno and McCallum (2008); Salomatin et al. (2009); Yakhnenko and Hovavar (2009); Rasiwasia et al. (2010) and Puttividhya et al. (2011). However, none of these extensions are able to find shared and private topics in the same sense as the CCA-based models do for continuous data. Instead, the models attempt to enforce strong correlation between the modalities, which is a reasonable assumption when analyzing e.g. multi-lingual textual corpora with similar languages but that does not hold for analysis of images associated with free-flowing text. In most cases, the images will contain a considerable amount of information not related to the text snippet, and it is not even guaranteed that the text is related at all to the visual content of the image.

In this work, we introduce a novel topic model that combines the two above lines of work. It builds on the correlated topic models (CTM) by Blei and Lafferty (2007) and Paisley et al. (2011), by modeling correlations between topic allocations and by using a hierarchical Dirichlet process (HDP) formulation for automatically learning the number of the topics. The proposed factorized multi-modal topic model integrates the technical improvements of these single-modality topic models to the multi-modal application, and in particular automatically learns to make some topics

specific to each of the modalities, implementing the factorization idea of Klami and Kaski (2008) and Jia et al. (2010) used for continuous data. The component selection plays a crucial role in implementing this property, implying that the HDP-based technique for automatically selecting the complexity is even more important for factorized multi-modal models than it would be a for a regular topic model.

The primary advantage of the new model is that is does not enforce correlations between the modalities, like the earlier multi-modal topic models do, but instead factorizes the variation into interpretable topics describing shared and private structure. The model is very flexible and does not enforce any particular factorization structure, but instead learns it from the data. For example, the model can completely ignore the shared topics in case the modalities are independent or find almost solely shared topics when they are strongly correlated. In this work we demonstrate the model in analyzing modalities that have only weak relationships, a scenario for which the previous models would not work. In particular, we analyze a collection of Wikipedia pages that consist of images and the whole text on the page. Such a collection has relatively low between-modality correlation and in particular includes considerable amount of text that is not related to the image at all, necessitating topics private to the text modality. The proposed model is shown to clearly outperform alternative HDP-based topic models as well as correspondence LDA (Blei and Jordan, 2003) in the task of inferring the contents of a missing modality.

2 BACKGROUND: TOPIC MODELS

To briefly summarize the topic models and to introduce the notation used in the paper, we describe the standard topic model of Latent Dirichlet Allocation (LDA) (Blei et al., 2003) through its generative process. We assume that words occurring in a document are drawn from K topics. Each topic specifies a multinomial probability distribution over the vocabulary, parameterized through $\boldsymbol{\eta}_k$ drawn from the Dirichlet distribution $\mathrm{Dir}(\gamma\mathbf{1})$, and the topic proportions are multinomial with parameters $\boldsymbol{\theta} \sim \mathrm{Dir}(\nu\mathbf{1})$. The documents are generated by repeatedly sampling a topic indicator $z \sim \mathrm{Multi}(\boldsymbol{\theta})$ and then drawing a word from the corresponding topic as $x \sim \mathrm{Multi}(\boldsymbol{\eta}_z)$.

We will also heavily depend on the concept of correlated topic models (CTM) (Blei and Lafferty, 2007). In the standard LDA the topic proportions $\boldsymbol{\theta}$ drawn from the Dirichlet distribution become independent except for weak negative correlation stemming from the normalization constraint. CTM replaces this choice by logistic normal distribution, first drawing an auxiliary variable from a Gaussian distribution $\boldsymbol{\xi} \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and specifying the topic distribution as $\boldsymbol{\theta} \propto \exp(\boldsymbol{\xi})$. The topics become correlated when $\boldsymbol{\Sigma}$ is not diagonal, and empirical experiments show increased predictive accuracy.

Finally, our model will be formulated through a hierarchical Dirichlet process (HDP) formulation (Teh et al., 2006), to enable automatic choice of the number of topics. As mentioned in the introduction, the choice is even more critical for multi-modal models, since we will have several sets of topics instead of just a single one; specifying the complexity for all of those in advance would not be feasible. Our model will use elements from the recently introduced Discrete Infinite Logistic Normal (DILN) model by Paisley et al. (2011), which incorporates HDP into CTM. The key idea of DILN is that the topic distributions $\boldsymbol{\theta}$ are made sparse by multiplying the $\exp(\boldsymbol{\xi})$ by sparse topic-selection terms. The topic distribution is given by $\boldsymbol{\theta}_k \propto \mathrm{Gamma}(\beta\mathbf{p}_k, \exp(-\boldsymbol{\xi}_k))$, where both β and $\mathbf{p}_k$ come from a stick-breaking process: β is the second level consentration parameter, and $\mathbf{p}_k = V_k \prod_{i=1}^{k-1}(1 - V_i)$, where $V_k \sim \mathrm{Beta}(1, \alpha)$ with α as the first level concentration parameter. The expected value of $\boldsymbol{\theta}_k$ is proportional to $\beta\mathbf{p}_k \exp(\boldsymbol{\xi}_k)$, illustrating the way the different parameters influence the topic weights. For any finite data collection, $\mathbf{p}_k > 0$ only for a finite subset of topics and hence the model automatically selects the number of topics.

3 FACTORIZED MULTI-MODAL TOPIC MODEL

Consider a collection of documents each containing M weakly correlated modalities, where each modality has its own vocabulary. In the application of this paper the two vocabularies are textual and visual words collected from Wikipedia pages with text and a single image (though the model would directly generalize to multiple images). We introduce a novel multi-modal topic model that can be used to learn dependencies between these modalities, enabling e.g. predicting the textual content associated with a novel image. The problem is made particularly challenging by the weak relationship between the modalities; several of the documents will contain large amounts of text not related to the image content.

For modeling the data, we will use M separate vocabularies, so that words (or visual words) for each modality are drawn from separate dictionaries $\boldsymbol{\eta}^{(m)}$ specific to each view m. The topic proportions $\boldsymbol{\theta}^{(m)}$ will also be specific to each modality, whereas the actual words are sampled independently for each modal-

ity given the topic proportions. The essential modeling question is then how the topic proportions are tied with each other, in order to achieve the factorization into shared and private topics. In brief, we will do this by (i) modeling dependencies between topics both within and across modalities and (ii) automatically selecting the number of topics for each type (shared or private to any of the modalities).

The topic proportions $\boldsymbol{\theta}^{(m)}$ are made dependent by introducing auxiliary variables $\boldsymbol{\xi}^{(m)}$, denoting by $\boldsymbol{\xi} = (\boldsymbol{\xi}^{(1)}, ..., \boldsymbol{\xi}^{(M)})$ the concatenation of them, and using the CTM prior $\boldsymbol{\xi} \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. This part of the model corresponds to the 'multi-field CTM with different topic sets' by Salomatin et al. (2009), and the different blocks in $\boldsymbol{\Sigma}$ describe different types of dependencies between the topic proportions. In particular, the blocks around the diagonal describe dependencies between the topic proportions of each modality, whereas the off-diagonal blocks describe dependencies in topic proportions between the modalities.

Having a CTM for the joint topic distribution is not yet sufficient for separating the shared topics from private ones, since we can only control the correlation between the topic proportions. A large correlation between two topics for different modalities would imply that it is shared, but lack of correlation (that is, $\boldsymbol{\Sigma}_{kl} = 0$) would not make either component private. Instead, the weights would simply be determined independently. To create separate sets of shared and private topics we need to be able to switch some of the topics off in one or more of the modalities, similarly to how Jia et al. (2010) and Virtanen et al. (2011) switch off components to make the same distinction in continuous data models. In the case of multi-field CTM this could only be done by driving $\boldsymbol{\mu}_k$ (the mean of the Gaussian prior for $\boldsymbol{\xi}_k$) towards minus infinity, which is not encouraged by the model and is difficult to achieve with mean-field updates.

We implement the shared/private choice by separate HDPs, one for each modality, switching a subset of topics off for each modality separately by a mechanism similar to how the single-view DILN model (Paisley et al., 2011) selects the topics. We introduce $\beta^{(m)}$ and $\mathbf{p}^{(m)}$ for each modality $m = 1, ..., M$, and draw them from separate HDPs, resulting in $\boldsymbol{\theta}^{(m)} \propto \mathrm{Gamma}(\beta^{(m)}\mathbf{p}^{(m)}, \exp(-\boldsymbol{\xi}^{(m)}))$ as the final topic proportions. The topic distributions are still shared through $\boldsymbol{\xi}^{(m)}$ that were drawn from a single high-dimensional Gaussian, but for each modality the stick weights $\mathbf{p}^{(m)}$ select different subsets of topics to be switched off. In the end, a finite number of topics remain for each modality, and the private topics can be identified as ones that have non-zero weight for one modality and are not correlated with topics active in other modalities.

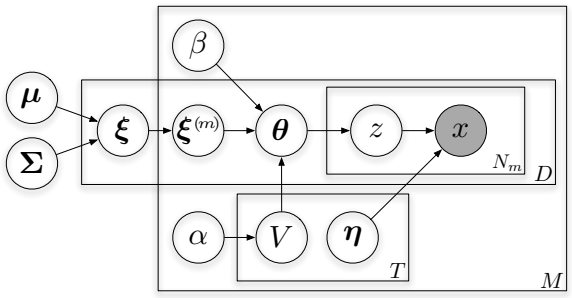


Figure 1: A graphical representation of the factorized multi-modal topic model. The data has D documents described by M modalities. For each modality, the words $\mathbf{x}^{(m)}$ are drawn from dictionary specific to that modality, according to topic proportions $\boldsymbol{\theta}^{(m)}$ also specific to the modality. The topic proportions are generated by logistic transformation of latent variables $\boldsymbol{\xi}^{(m)}$ that model the correlations between the topics both within and across modalities, followed by topic selection with a HDP (denoted by V and β in the plate; see text for details) for each modality. As a result, the model learns both topics modeling correlations between the modalities as well as topics private to each modality.

The final generative model motivated by the above discussion results in a collection of M correlated BOW data sets $\mathbf{X}^{(m)}$, generated as follows (see Figure 1 for graphical representation). For the whole collection we:

- create a dictionary of $T^{(m)}$ topics for each modality by drawing $\boldsymbol{\eta}_k^{(m)} \sim Dir(\gamma^{(m)}\mathbf{1})$ for $k = 1, .., T^{(m)}$
- draw the parameters $\alpha^{(m)}, \beta^{(m)}, V^{(m)}$ of the DILN distribution for each modality from the stick-breaking formulation and construct $\mathbf{p}^{(m)}$.

For each document d we then draw $\boldsymbol{\xi} \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and partition it into the different modalities as $\boldsymbol{\xi} = (\boldsymbol{\xi}^{(1)}, ..., \boldsymbol{\xi}^{(M)})$. For each modality, we then generate the words independently as follows:

- form the topic proportion by drawing $Y_k^{(m)} \sim \mathrm{Gamma}(\beta^{(m)}\mathbf{p}_k^{(m)}, \exp(-\boldsymbol{\xi}_k^{(m)}))$ and set $\boldsymbol{\theta}_k^{(m)} = \frac{Y_k^{(m)}}{\sum_{i=1}^{T^{(m)}} Y_i^{(m)}}$
- draw $N^{(m)}$ words by choosing a topic $z \sim \mathrm{Multi}(\boldsymbol{\theta}^{(m)})$ and drawing a word $x \sim \mathrm{Multi}(\boldsymbol{\eta}_z^{(m)})$

3.1 INFERENCE

For learning the model parameters we use a truncated variational approximation following closely the algorithm given by Paisley et al. (2011), the main difference being that we have M separate sets of $\boldsymbol{\eta}$, β and $\mathbf{p}$, one for each modality. The above generative process is truncated by setting $V^{(m)}_{T^{(m)}} = 1$, forcing the stick lengths beyond the truncation level $T^{(m)}$ to be zero, and the resulting factorized approximation is given by

$$Q = \prod_{m=1}^{M}\prod_{d=1}^{D}\prod_{n_m=1}^{N_m}\prod_{k=1}^{T} q(z^{(m)}_{dn_mk})q(Y^{(m)}_{dn_mk})q(\xi^{(m)}_{dk})q(\boldsymbol{\eta}^{(m)}_{k})$$
$$q(V^{(m)}_{k})q(\alpha_m)q(\beta_m)q(\boldsymbol{\mu})q(\boldsymbol{\Sigma}),$$

where to simplify notation we assume $T_m = T \;\forall\; m$. The algorithm proceeds by updating each factor in turn while keeping the others fixed, using either gradient ascent or analytic solution for maximizing the lower bound of the approximation for each of the terms (see Paisley et al. (2011) for details).

The main difference in the algorithms comes from updating $\boldsymbol{\xi}$, since in our case it goes over M sets of topics instead of just one, yet the activities within each set are governed by separate HDPs. We use a diagonal Gaussian factor $q(\boldsymbol{\xi}) = \mathrm{N}(\tilde{\boldsymbol{\xi}}, \mathrm{diag}(\tilde{\mathbf{v}}))$, where $\tilde{\mathbf{v}}$ denotes the variances of the dimensions, and use gradient ascent for jointly updating the parameters. To simplify notation we use $\boldsymbol{\xi}$ and $\mathbf{v}$ to denote the expectation and variance of the factorial distribution. The relevant part of the lower bound is

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\xi},\mathbf{v}} = &-\sum_{m=1}^{M}\beta^{(m)}{\mathbf{p}^{(m)}}^T\boldsymbol{\xi}^{(m)} \\ &-\sum_{m=1}^{M}\mathbb{E}[\boldsymbol{\theta}^{(m)}]^T\mathbb{E}[\exp(-\boldsymbol{\xi}^{(m)})] \\ &-(\boldsymbol{\xi}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\xi}-\boldsymbol{\mu})/2 \\ &-\mathrm{diag}(\boldsymbol{\Sigma}^{-1})^T\mathbf{v}/2 + \log(\mathbf{v})^T\mathbf{1}/2.\end{aligned} \tag{1}$$

Here $\boldsymbol{\Sigma}^{-1}$ couples the separate $\boldsymbol{\xi}^{(m)}$ terms in the partial derivatives as

$$\begin{aligned}\frac{\partial\mathcal{L}_{\boldsymbol{\xi},\mathbf{v}}}{\partial\boldsymbol{\xi}^{(m)}} = &-\beta^{(m)}\mathbf{p}^{(m)} + \mathbb{E}[\boldsymbol{\theta}^{(m)}]\mathbb{E}[\exp(-\boldsymbol{\xi}^{(m)})] \\ &-(\boldsymbol{\Sigma}^{-1})_{m,m}(\boldsymbol{\xi}^{(m)}-\boldsymbol{\mu}^{(m)}) \\ &-\sum_{j\neq m}(\boldsymbol{\Sigma}^{-1})_{m,j}(\boldsymbol{\xi}^{(j)}-\boldsymbol{\mu}^{(j)}),\end{aligned}$$

with $(\boldsymbol{\Sigma}^{-1})_{i,j}$ denoting a block of $\boldsymbol{\Sigma}^{-1}$ corresponding to modalities i and j. The inverse of $\boldsymbol{\Sigma}$ remains constant during the gradient descent, and hence only needs to be evaluated once for every time the factor $q(\boldsymbol{\xi})$ is updated.

We use maximum marginal likelihood to update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ resulting in closed form updates

$$\boldsymbol{\mu} = \frac{1}{D}\sum_{d=1}^{D}\boldsymbol{\xi}_d$$
$$\boldsymbol{\Sigma} = \sum_{d=1}^{D}\big((\boldsymbol{\xi}_d-\boldsymbol{\mu})(\boldsymbol{\xi}_d-\boldsymbol{\mu})^T + \mathrm{diag}(\mathbf{v}_d)\big)/D.$$

3.2 PREDICTION

The model structure is well suited for prediction tasks, where the task is to infer missing modalities for a new document given that one of them is observed (e.g. infer the caption given the image content). This is because the correlations between the topic proportions provide a direct link between the modalities, and the private topics explain away all the variation that is not useful for predictions.

Here we present the details of the prediction for the special case with just one observed modality (j) and one missing modality (i). Given the observed data we first infer the topic proportions $\hat{\boldsymbol{\theta}}^{(j)}$ and then auxiliary variable $\hat{\boldsymbol{\xi}}^{(j)}$ by maximizing a cost similar to (1), but only using the newly inferred topic proportions of the observed modality and the corresponding part of $\boldsymbol{\Sigma}$. As $\hat{\boldsymbol{\xi}}$ comes from a Gaussian distribution we can infer $\hat{\boldsymbol{\xi}}^{(i)}$ given $\hat{\boldsymbol{\xi}}^{(j)}$ with the standard conditional expectation as

$$\begin{aligned}\hat{\boldsymbol{\xi}}^{(i)} &= \boldsymbol{\mu}^{(i)} + \boldsymbol{\Sigma}_{i,j}\boldsymbol{\Sigma}^{-1}_{j,j}(\hat{\boldsymbol{\xi}}^{(j)} - \boldsymbol{\mu}^{(j)}) \\ &= \boldsymbol{\mu}^{(i)} + \mathbf{W}(\hat{\boldsymbol{\xi}}^{(j)} - \boldsymbol{\mu}^{(j)}).\end{aligned} \tag{2}$$

Here $\mathbf{W}$ involves the corresponding part of the between-topic covariance matrix $\boldsymbol{\Sigma}$ as indicated above, and can be seen as a projection matrix transforming the components of one modality to another. Finally, the newly estimated $\hat{\boldsymbol{\xi}}^{(i)}$ for the missing views is converted back to the expecteed topic proportion $\hat{\boldsymbol{\theta}}$ by exponentiation and multiplying with the corresponding stick lengths $\mathbf{p}^{(i)}$.

3.3 SHARED AND PRIVATE TOPICS

The key novelty of the model is its capability to learn both topics that are shared and that are private to each modality, without needing to specify them in advance. Since the way these topics appear is by no means transparent in the above formulation, we will here discuss the property in more detail. In brief, the distinct nature for the topics comes from an interplay of the correlations between the topics of different modalities and the HDP procedure that turns some of the topics off

for each modality. In particular, neither of these properties alone would be sufficient.

As mentioned already in Section 3, merely having separate $\boldsymbol{\xi}^{(m)}$ drawn from a single Gaussian is not sufficient for finding private topics. At best, the correlation structure can specify that the weights will be independent for the modalities. Next we explain how the other key element of the model, separate selection of active topics for each modality, is not sufficient alone either. We do that by considering a special case of the model that assumes equal $\boldsymbol{\xi} = \boldsymbol{\xi}^{(m)}$ for all views but has separate stick-breaking processes switching some of the topics off for each of the views. We call this alternative model mmDILN, due to the fact how it implementes multi-modal LDA of Blei and Jordan (2003) with DILN-style component selection.

Intuitively, mmDILN model could find private topics simply by setting $\mathbf{p}_k^{(m)}$ to small value for topics that are not needed in that modality. However, it cannot make correct predictions from one modality to another, and hence fails in achieving one of the primary goals for shared-private factorizations. If $\mathbf{p}_k^{(m)}$ is small then the model has no information for inferring $\boldsymbol{\xi}_k$ from that view, and hence also all other elements $\boldsymbol{\xi}_l$ that correlate with $\boldsymbol{\xi}_k$ will be incorrect. If $\boldsymbol{\xi}_k$ was an important topic for the other view, the predictions will be severely biased. Our model avoids this issue by having the separate $\boldsymbol{\xi}^{(m)}$ parameters, leading to correct across-modality predictions as described in the previous section. In the experimental section we will empirically compare the proposed model with mmDILN, demonstrating how mmDILN indeed has very poor predictive accuracy despite modeling the training data almost as well. Hence, even though the structure is in principle sufficient for learning private topics, the model has no practical value as a shared-private factorization.

In order to recognize the nature of each of the topics, we need to look at both the covariance $\boldsymbol{\Sigma}$ between the topic weights and the modality-specific stick weights $\mathbf{p}_k^{(m)}$. Since the topics can be (potentially strongly) correlated both within and across modalities, we can identify private topics only by searching for topics that do not correlate with any topic that would be active in any other modality. In the experiments we demonstrate how the topics can be ranked according to how strongly they are shared with another modality, by inspecting the elements of $\boldsymbol{\Sigma}$.

4 RELATED WORK

In this section we relate the model to other approaches for modeling multi-modal count data.

4.1 MULTI-MODAL TOPIC MODELS

The multi-modal extension of LDA (mmLDA) by Blei and Jordan (2003) and its non-parametric version mmHDP by (Yakhnenko and Hovavar, 2009) assume all modalities to share the same topic proportions, and essentially extend LDA only by having separate dictionaries for each modality and generating the words for the domains independently. For many real world data sets the assumption of identical topic proportions is too strong, and the model tries to enforce correlations even when they do not exist. While the assumption may help in picking up topics that would be weak in either modality alone, it makes identifying the true correlations almost impossible.

Such models fail especially when modeling data having strong private topics in one modality. Since the topic proportions are shared, the topic must be present in other modalities as well and becomes associated with a dictionary that merely replicates the overall distribution of the words. Such topics are particularly harmful for prediction tasks. When the dictionary of a topic matches that of the background word distribution, it will be present in every document in that modality. For example, when predicting text from images we could learn to associate politics (a strong topic private to the text modality) with the overall visual word distribution, resulting in all of the predictions including terms from the politics topic.

Salomatin et al. (2009) took a step towards our model with their multi-field CTM. It extends CTM by introducing separate $\boldsymbol{\xi}^{(m)}$ for each modality, similarly to our model. However, as described in the previous section the separate topic proportions are not yet sufficient for separating the shared topics from private ones.

4.2 CONDITIONAL TOPIC MODELS

Lots of recent work on multi-modal topic modeling framework has focused on building conditional models, largely for image annotation task. Correspondence LDA (corrLDA) proposed simultaneously to mmLDA in (Blei and Jordan, 2003) is a prominent example, assuming that the image is generated first and the text depends on the image content. Both modalities are assumed to share the same topic weights. While such models are very useful for modeling the conditional relationship, they do not treat the modalities symmetrically as in our model. Recently Puttividhya et al. (2011) proposed an extension of corrLDA, replacing the identical topic distributions with a regression module from image topics to the textual annotation topics. The added flexibility results in better predictive performance, but the model remains a directional one,

in contrast to our model that generates all modalities with equal importance. For applications treating only two modalities and having a specific task that makes one of them more important (say, image annotation) the conditional models often work well. However, they do not easily generalize to multiple modalities and are not flexible in terms of the eventual application.

Other conditional models focus on conditioning on meta-data, such as author or link structure (Mimno and McCallum, 2008; Hennig et al., 2012). Such models allow integrating data that are not necessarily in count format, but the same distinction of directional versus generative applies. However, this family of models could be integrated with our solution, incorporating a meta-data link into our multi-modal model. In essence, the choice of whether meta-data is modeled or not is independent of the choice of how many count data modalities the data has.

4.3 CANONICAL CORRELATIONS

As described earlier, the model bears close resemblance to how CCA models correlations between continuous data, the similarities being most apparent with the recent re-interpretations of CCA as shared-private factorization (Klami and Kaski, 2008; Jia et al., 2010). The technical details of the solutions are, however, very different as the normalization of topic proportions makes the techniques used for continuous data not feasible for topic models.

Despite the mismatch of data types, CCA can be used for modeling count data as well. The most promising direction would be to apply kernel-CCA, but there are no obvious choices for the kernel function that would directly match the analysis of image-text pairs. As one practical remedy, (Rasiwasia et al., 2010) combined CCA and LDA directly by first estimating a separate LDA model for each modality and then combining the resulting topic proportions with CCA. Our approach does not rely on two separate analysis steps that do not result in directly interpretable private topics.

5 EXPERIMENTS AND RESULTS

5.1 DATA AND MEASURES

We validate the model on real data collected from Wikipedia[1]. We constructed a data collection with $D = 20,000$ documents, each consisting of a single image represented with 5000 SIFT patches and text (the contents of the whole Wikipedia page) represented with a vocabulary of 7500 most frequent terms, after stopword removal. We make a random 50/50-split into test and train data. To demonstrate the ability of the proposed model to correctly model the relationships between the two modalities, we evaluate the model with conditional perplexity of a missing modality for a new sample:

$$\mathcal{P}^{(m)}_{\text{train}} = \exp\left(- \frac{\sum_{d \in D_{train}} \log p(\mathbf{x}^{(m)}_d)}{\sum_{d \in D_{train}} \mathbf{x}^{(m)}_d} \right)$$

$$\mathcal{P}^{(i)|(j)}_{\text{test}} = \exp\left(- \frac{\sum_{d \in D_{test}} \log p(\mathbf{x}^{(i)}_d | \mathbf{x}^{(j)}_d)}{\sum_{d \in D_{test}} \mathbf{x}^{(i)}_d} \right),$$

where $\mathbf{x}^{(m)}_d$ denotes concatenation of $N^{(m)}_d$ words. These quantities measure how well the model can relate the visual content to the textual content, corresponding to the document completion task of Wallach et al. (2009) but computed across modalities.

We compare our model to three alternatives representing various kinds of multi-modal topic models: mmDILN (Section 3.3), mmHDP (Section 4.1) and corrLDA (Section 4.2). Both mmDILN and mmHDP are comparable to our model in making automatic topic number selection and modeling both modalities symmetrically. Consequently, the experiments will focus on demonstrating the importance of finding the correct factorization into shared and private topics. The corrLDA is included as an example of a conditional model that gives an alternative approach to solving a similar prediction task. Note that we need to learn two separate corrLDA models, one for predicting text from images and one for the other direction, whereas the other models can do both types of predictions. For corrLDA we use 100 topics (the threshold we used for nonparametric models).

5.2 INFERENCE SPEED

First we show that the variational approximation used for inference is efficient. Figure 2 shows how the algorithm converges for both $N = 400$ and $N = 10000$ documents already after some tens of iterations. For both experiments we used a maximum of $T = 100$ topics. The convergence of mmHDP and mmDILN is similar (not shown).

5.3 PREDICTING TEXT FROM IMAGES AND VISE VERSA

Figure 3 shows the evaluation for training and test sets for the proposed model and the comparison methods, measured as the perplexity on training data and the conditional perplexity of images given the text and text given the images. The proposed method, which is more flexible than the alternatives, reaches better

[1] Available from `http://www.eecs.berkeley.edu/~jiayq/`

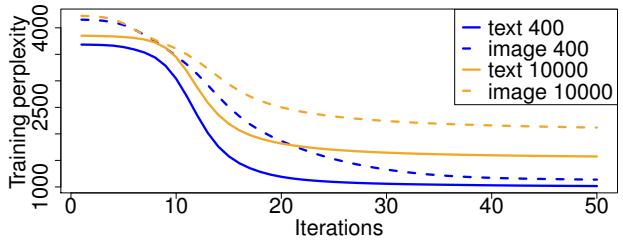


Figure 2: Training perplexity as function of algorithm iterations.

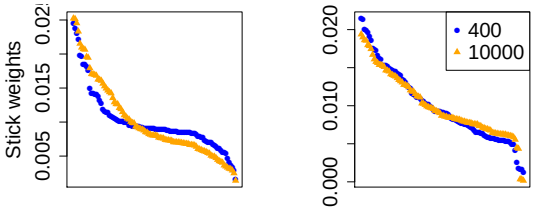
(lower) perplexity on the training and testing data due to being able to describe both variation not shared by the other modality without needing to introduce noise topics.

A notable observation is that the baseline methods perform worse at predicting text from images as the amount of training data increases. This illustrates clearly the fundamental problem in modeling multi-modal collections without separate private topics. Since the text documents are easier to model than the images, the alternative models start to focus more and more on modeling the text when there is large amount of data. The dominant topics start describing the text alone, yet they are also active in the image modality but with a topic that does not contain any information. Given a new image sample, the estimated topic proportions will be arbitrary and hence do not enable meaningful prediction. The proposed model, however, learns to make those textual topics private to the text modality, while capturing weaker correlations between the two modalities with shared topics. The model still cannot predict textual information not correlated with the image content, but it learns correctly not to even attempt that and manages to make accurate predictions for the aspects that are correlated.

5.4 SHARED AND PRIVATE TOPICS

To illustrate how the HDP-formulation chooses the topics, we visualize the stick parameters $\mathbf{p}$ in Figure 4. First, we notice that the last sticks have close to zero weight, indicating that the chosen truncation level $T = 100$ is sufficient. More importantly, we see that the weights for the text and image topics are different (the image topics are more spread out), motivating the choice of separate weights for the modalities.

To further understand how the proposed model is able to find both shared and private topics, we explore the nature of the individual topics. Since the SIFT vocabulary is not easily interpretable by visual inspection, we illustrate the property for the textual topics. For each textual topic we measure the amount of correlation between the other modality by inspecting the correlation structure in $\mathbf{\Sigma}$, and then rank the topics according to this measure. This results in a ranked list of the text topics, the first ones being strongly shared by the two modalities while the last ones are private to the text modality.



(a) Our model: text (b) Our model: image

Figure 4: Visualization of stick parameter $\mathbf{p}$ of the proposed model for the text modality (a) and the image modality (b) reveals how they are not identical for the two modalities. Both figures show the weights for two models learned with 400 and 10,000 documents, revealing how the distribution is learned fairly accurately already from a small collection.

More specifically, denoting the separate blocks in the covariance matrix as

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{t,t} & \mathbf{\Sigma}_{t,i} \\ \mathbf{\Sigma}_{i,t} & \mathbf{\Sigma}_{i,i} \end{pmatrix}, \quad (3)$$

we convert it to a correlation matrix, $\mathbf{\Omega}$, threshold small values out (we used a threshold of 0.2) and extract the cross-correlation between textual (rows) and visual topics (columns), to get $\mathbf{\Omega}_{t,i}$. Then for each textual topic we define visual relevance, $\boldsymbol{\rho}_k$, as row mean of absolute values of $\mathbf{\Omega}_{t,i}$, written as $\boldsymbol{\rho} = \frac{1}{T}|(\mathbf{\Omega}_{t,i})|\mathbf{1}$ [2]. This quantity captures general and rich visual combinations that co-occur with the textual topics, and it is worth noticing how the measure is very general: It allows multiple visual topics to correlate with one textual topic (and vise versa), and includes both positive and negative correlations that are typically equally relevant (negative correlation can be seen as absence of a visual component) (See Figure 5 for demonstration).

The textual topics are ranked according to $\boldsymbol{\rho}$ in Figure 6 . There are a few very strong shared topics between text and image modalities, and at the end of the list we have several topics private to the text modality, indicated by zero correlation with the image modality. This matches with the intuition that the full text of a Wikipedia page cannot be mapped to the image content in all cases. Table 1 summarizes the six text topics most strongly correlating with the image modality, as well as six topics that are private

[2]We also tried using the maximum element instead of the mean; it results in fairly similar ranking.

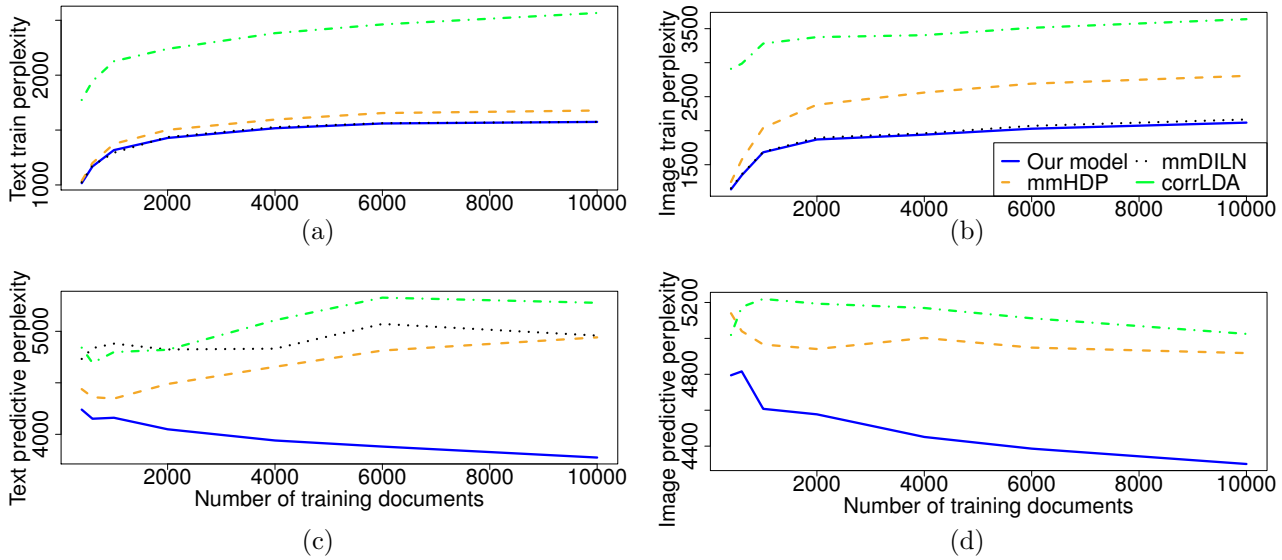


Figure 3: Training and test perplexities (lower is better) for the two modalities. For training data we show the perplexity of modeling the text (a) and images (b) separately. For test data, we show the conditional perplexity of predicting text from images (c) and predicting images from text (d), corresponding to the document completion task used for evaluating topic models. The proposed method outperforms the comparison ones in all respects. The comparison methods mmHDP, mmDILN and corrLDA that are not able to extract topics private to either modality are not able to learn good predictive models, demonstrated especially by the error increasing as a function of training samples in (c). The image prediction perplexity for mmDILN is outside the range depicted in (d), above 5400 for all training set sizes.

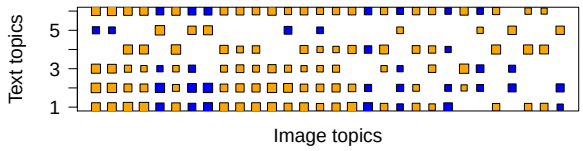


Figure 5: Illustration of part of cross-correlation between text topics and image topics corresponding to subset of $\mathbf{\Omega}_{(t,i)}$, where yellow represents positive correlations, and blue represents negative ones. The size of the boxes corresponds to the absolute value.

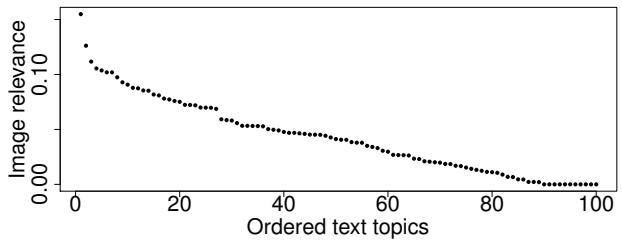


Figure 6: Text topics ordered according to visual relevance $\boldsymbol{\rho}$. We see that there are a few strongly correlating topics, and that the model has found roughly 10 topics that are private to the text domain. Note that such topics may still be important for modeling the whole multi-modal corpus, whereas they do not contribute to the cross-modal information transfer.

to the text modality, revealing very clear interpretations. The most strongly correlating topic covers airplanes, which are known to be easy to recognize from the images due to the distinct shapes and background. The second topic is about maps that also have clear visual correspondence, and the other strongly correlated topics also cover clearly visual concepts like buildings, cars and railroads. The topics private to the text domain, in turn, are about concepts with no clear visual counterpart: economy, politics, history and research. In summary, the model has separated the components nicely into shared and private ones, and provides additional interpretability beyond regular multi-modal topic models.

6 DISCUSSION

Our paper ties together two separate lines of work for analysis of multi-modal data. In particular, we created a novel multi-modal topic model which extends earlier tools for analysis of multi-modal count data by incorporating elements found useful in the continuous-valued case. We explained how learning topics private to each modality is of crucial importance while modeling modalities with potentially weak correlations, and

Table 1: Text topics ranked according to visual relevance, summarized by the words with highest probability. The topic indices match the ranking in Figure 6. The shared topics have clear visual counterparts, whereas the private ones do not relate with any kind of visual content.

Shared topics
T1 airport flight airlines air international aircraft aviation terminal passengers airline boeing flights airways service airports passenger accident
T2 format dms latd dm longm latm longs lats launched mi broken mill sold renamed dec captured rapids class feet coordinates built lake located
T3 building house built buildings street hall st century tower houses west designed design castle south north east side main square large end site
T4 car engine cars model models ford engines race rear series front racing wheel year driver speed vehicles vehicle production hp motor drive
T5 retrieved album song music video released single awards number billboard chart top release mtv songs media love show uk jackson hot albums
T6 line railway station rail trains train service lines bus transport services system railways stations built railroad passenger main metro transit
Topics private to the text domain
T95 president washington post united american national states secretary december november september times military dc kennedy press security
T96 ottoman turkish turkey kosovo armenian war greek serbia bulgarian serbian government border bulgaria turks forces croatian albanian republic
T97 research science development institute university management scientific technology design world national engineering work human international
T98 government state national european policy council international states members act union political countries system nations article parliament
T99 nuclear weapons anti power protest bomb people protests united protesters government strike peace states march reactor atomic april test
T100 economic trade economy world production industry oil million growth development government agricultural market agriculture industrial

demonstrated empirically how such a property can only be obtained by combining two separate elements: modeling correlations between separate topic weights for each modality, and learning modality-specific indicators switching unnecessary topics off. For implementing these elements we combined state-of-art techniques in topic models, integrating the DILN distribution (Paisley et al., 2011) into a model similar to the multi-field correlated topic model of Salomatin et al. (2009), to create an efficient learning algorithm readily applicable for relatively large document collections.

Acknowledgements

AK and SK were supported by the COIN Finnish Center of Excellence and the FuNeSoMo exchange project. AK was additionally supported by Academy of Finland (decision number 133818) and PASCAL2 European Network of Excellence.

References

Blei, D., Ng, A. and Jordan, M. (2003). Latent Dirichlet allocation. *JMLR*, 3:993–1022.

Blei, D. and Jordan, M. (2003). Modeling annotated data. In *SIGIR*.

Blei, D. and Lafferty, J. (2007). A correlated topic model of science. *Annals of Applied Sciences*, 1:17–35.

Hennig, H., Stern, D., Herbrich, R. and Graepel,T. (2012). Kernel topic models. In *AISTATS*.

Jia, Y., Salzmann, M. and Darrell, T. (2010). Factorized latent spaces with structured sparsity. In *NIPS 23*.

Klami, A. and Kaski, S. (2008). Probabilistic approach to detecting dependencies between data sets. *Neurocomputing*, 72(1-3):39–46.

Mimno, D. and McCallum A. (2008). Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In *UAI*.

Paisley, J., Wang C. and Blei, D. (2011). Discrete infinite logistic normal distribution. In *AISTATS*.

Puttividhya, D., Attias, H. and Nagarajan, S. (2011). Topic-regression multi-modal latent Dirichlet allocation for image annotation. In *CVPR*.

Puttividhya, D., Attias, H. and Nagarajan, S. (2009). Independent factor topic models. In *ICML*.

Rasiwasia, N., Pereira, J., Covielho, E., Doyle, G., Lanckriet G., Levy, R. and Vasconcelos N. (2010). A new approach to cross-modal multimedia retrieval. In *ACM Multimedia*.

Salomatin, K., Yang, Y. and Lad, A. (2009). Multi-field correlated topic modeling. In *SDM*.

Teh, Y., Blei, D. and Jordan, M. (2006). Hierarchical Dirichlet processes. JASA, 101(476):1566–1581.

Virtanen, S., Klami, A. and Kaski, S. (2011). Bayesian CCA via structured sparsity. In *ICML*.

Wallach, H.M., Murray, I., Salakhutdinov, R. and Mimno, D. (2009). Evaluation methods for topic models. In *ICML*.

Yakhnenko, O. and Honavar, V. (2009). Multi-modal hierarchical Dirichlet process model for predicting image annotation and image-object label correspondence. In *SDM*.

Latent Dirichlet Allocation Uncovers Spectral Characteristics of Drought Stressed Plants

Mirwaes Wahabzada[1*°], **Kristian Kersting**[12*], **Christian Bauckhage**[1], **Christoph Römer**[2], **Agim Ballvora**[3], **Francisco Pinto**[4], **Uwe Rascher**[4], **Jens Léon**[3], **Lutz Plümer**[2]
[1]Fraunhofer IAIS, Sankt Augustin, Germany. [2]Institute of Geodesy and Geoinformation, University of Bonn, Germany. [3]Institute of Crop Science and Resource Conservation, Plant Breeding, University of Bonn, Germany. [4]Institute of Bio- and Geosciences, IBG-2: Plant Sciences, Forschungszentrum Jülich, Germany.

Abstract

Understanding the adaptation process of plants to drought stress is essential in improving management practices, breeding strategies as well as engineering viable crops for a sustainable agriculture in the coming decades. Hyper-spectral imaging provides a particularly promising approach to gain such understanding since it allows to discover non-destructively spectral characteristics of plants governed primarily by scattering and absorption characteristics of the leaf internal structure and biochemical constituents. Several drought stress indices have been derived using hyper-spectral imaging. However, they are typically based on few hyper-spectral images only, rely on interpretations of experts, and consider few wavelengths only. In this study, we present the first data-driven approach to discovering spectral drought stress indices, treating it as an unsupervised labeling problem at massive scale. To make use of short range dependencies of spectral wavelengths, we develop an online variational Bayes algorithm for latent Dirichlet allocation with convolved Dirichlet regularizer. This approach scales to massive datasets and, hence, provides a more objective complement to plant physiological practices. The spectral topics found conform to plant physiological knowledge and can be computed in a fraction of the time compared to existing LDA approaches.

1 Introduction

Water scarcity is a principal global problem that causes aridity and serious crop losses in agriculture. It has been estimated that drought can cause a depreciation of crop yield up to 70% in conjunction with other abiotic stresses (Boyer, 1982; Pinnisi, 2008). Climate changes and a growing human population in parallel thus call for a sincere attention to advance research on understanding of plant adaptation under drought. A deep knowledge of the adaptation process is essential in improving management practices, breeding strategies as well as engineering viable crops for a sustainable agriculture in the coming decades. Accordingly, there is a dire need for crop cultivars with high yield and strong resistance against biotic and abiotic stresses.

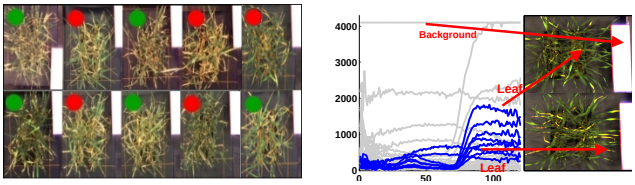


Figure 1: What are the specific spectral characteristics of plants suffering from drought stress? (Left) A collection of hyper-spectral images (projected to RGB space) within the flowering period. Stressed plants are indicated by red dots, control plants by green dots. Visually it is difficult to distinguish between control and stressed plant; compare e.g. the 2nd and 3rd image in the bottom row. (Right) Example spectral signatures taken from a hyper-spectral image for leaf and background pixels. (Best viewed in color)

Unfortunately, understanding stress is not an easy task. Stress resistance is the result of a complex web of interactions between the genotype and the environment leading to phenotypic expressions. It is contributed by a number of related traits that are controlled mostly by polygenic inheritance. In the past, a slow progress in the development of improving cultivars was mainly due to poor understanding of genetic factors that impact tolerance to drought (Passioura, 2002). Recently, progress has been made in under-

*Both authors contributed equally. °Contact author: mirwaes.wahabzada@iais.fraunhofer.de

standing the genetic basis of drought related quantitative trait loci (QTL), see e.g. (Lebreton et al., 1995; McKay et al., 2008). More recently, OMICS approaches have offered a direct molecular insight into drought tolerance mechanism, see e.g. (Rabbani et al., 2010; Guo et al., 2010; Abdeen et al., 2010). However, genetic and biochemical approaches are time consuming and still fail to fully predict the performance of new lines in the field. In recent years it is discussed that phenomic approaches, that measure the structural and functional status of plants may overcome the limited predictability and some authors have attributed this lack of high throughput phenomic data as the "phenomic bottleneck" (Richards et al., 2010).

Hyper-spectral imaging provides a particularly promising approach to sensor-based phenotyping. Its measurements were observed to contain early indicators of plant stress, see e.g. (Rascher et al., 2007; Rascher and Pieruschka, 2008). In contrast to conventional cameras, which record only 3 wavelengths per pixel, hyper-spectral cameras record a spectrum of several hundred wavelengths ranging from approximately 300nm to 2500nm resulting in big data cubes. These spectra contain information as to changes of the pigment composition of leaves which are the result of metabolic processes involved in plant responses to biotic or abiotic stresses. This information can be used e.g. using SVMs for classification of hyper-spectral signatures and in turn for prediction of biotic stress before symptoms become visible to the human eye, see e.g. (Rumpf et al., 2010; Römer et al., 2010), or for finding archetypical signatures (Kersting et al., 2012).

More important for the present study, hyper-spectral imaging was proven to be successful in discovering relationship between the spectral reflectance properties of vegetation and the structural characteristics of vegetation and pigment concentration in leaves (Govender et al., 2009); the spectral characteristics of vegetation are governed primarily by scattering and absorption characteristics of the leaf internal structure and biochemical constituents, such as pigments (e.g. chlorophyll a and b), water, nitrogen, cellulose and lignin, see (Curran et al., 1990; Gitelson and Merzlyak, 1996; Blackburn, 2007; Govender et al., 2009) and references in there). For instance, Gitelson and Merzlyak (1996) observed a increase of reflectance for the band 670nm if the amount of chlorophyll in the leaf was dropped as it is case in the presence of drought stress.

However, since stress reactions are the result of a complex web of interactions between the genotype and the environment, indices involving very few distinct wavelengths only run the risk of providing a oversimplified and actually wrong spectral characterization of drought reactions. In this study, we present a data-driven approach to discovering spectral drought stress indices, treating it as an unsupervised labeling problem at massive scale and solving it using an online variational Bayes approach (Hoffman et al., 2010; Wahabzada and Kersting, 2011) to latent Dirichlet allocation (Blei et al., 2003). Although, there are other data-driven approaches, such as low-rank matrix factorization based on sub-sampling (Mahoney and Drineas, 2009; Sun et al., 2008) or computing extreme data points (Thurau et al., 2012), these methods have a major limitation: they are not part-based. Part-based methods such as NMF, however, do not easily deal with additional information, e.g. relational information and short range dependencies.

Using a topic model is a sensible idea since it is common practice in plant physiology to talk about integral wavelength-reflectance pairs only (the words) within signatures (the documents). However, we have to be a little bit more careful. Signatures are still "curves" showing important short range dependencies among spectral wavelengths. In order to preserve the dependencies, we develop an online variational Bayes approach with convolved Dirichlet regularizer (Newman et al., 2011). Indeed, one may argue that subsampling is a valid alternative to deal with the massive amount of data at hand. However, plant physiologists often do not get used to the idea of throwing away information and actually do not trust the results of sample-based methods. In contrast, our regularized online VB scales well to massive datasets and does not throw away information, hence, provides a more objective complement to plant physiological practices. Moreover, the spectral topics found by LDA conform to plant physiological knowledge and can be computed in a fraction of the time compared to existing LDA approaches.

We proceed as follows. We start off by reviewing online variational Bayes for LDA. Afterwards, we develop the regularized variant. Before concluding we will present our main experimental evaluation on hyper-spectral images of plants with two treatments (control and stressed) as well as supplemental evaluations on two additional real world datasets, the network of human diseases and Wikipedia articles.

2 Online Variational Bayes for LDA

LDA is a Bayesian probabilistic model of collections of text documents (Blei et al., 2003). It assumes a fixed number of K underlying topics in a document collection. Topics are assumed to be drawn from a Dirichlet distribution, $\beta_k \sim Dir(\eta)$, which is a convenient conjugate to the multinomial distribution of words appearing in documents. According to LDA, documents

are generated by first drawing topic proportions according to $\theta_d \sim Dir(\alpha)$, where α is the parameter of the Dirichlet prior on the per-document topic distributions. Then for each word i a topic is chosen according to $z_{di} \sim Mult(\theta_d)$ and the observed word w_{di} is drawn from the selected topic, $w_{di} \sim Mult(\beta_{z_{di}})$.

In this paper, we focus on variational Bayesian (VB) inference. Here, the true posterior is approximated using a simpler, fully factorized distribution q. Following Blei et al. (2003); Hoffman et al. (2010), we choose $q(z, \theta, \beta)$ of the form $q(z_{di} = k) = \phi_{dw_{di}k}$, $q(\theta_d) = Dir(\theta_d, \gamma_d)$, and $q(\beta_k) = Dir(\beta_k, \lambda_k)$. The variational parameters ϕ, γ, and λ are optimized to maximize the Evidence Lower BOund (ELBO)

$$\log p(w \mid \alpha, \eta) \geq \mathcal{L}(w, \phi, \gamma, \lambda)$$

$$\triangleq \mathbb{E}_q[\log p(w, z, \theta, \beta \mid \alpha, \eta)] - \mathbb{E}_q[\log q(z, \theta, \beta)],$$

which is equivalent to minimizing the Kullback - Leibler divergence between $q(z, \theta, \beta)$ and the true posterior $p(z, \theta, \beta \mid w, \alpha, \eta)$.

Based on VB, Hoffman et al. (2010) have introduced an online variant that we here present for the batch case running over mini-batches (chunks of multiple observations). That is, we assume that the corpus of documents has been sorted uniformly at random and chunked into l mini-batches $B_1, B_2, \ldots, B_l$ of size S. That is, the ELBO $\mathcal{L}$ is set to maximize $\mathcal{L}(w, \phi, \gamma, \lambda) \triangleq \sum_{B_i} \sum_{d \in B_i} \ell(n_d, \phi_d(n_d, \lambda), \gamma_d(n_d, \lambda), \lambda)$, where n_d is the word count vector and $\ell(n_d, \phi_d(n_d, \lambda), \gamma_d(n_d, \lambda), \lambda)$ denotes the contribution of document d to the ELBO. As Hoffman et al. (2010) have shown this mini-batch VB-LDA corresponds to a stochastic natural gradient algorithm on the variational objective $\mathcal{L}$. Using mini-batches reduces the noise in the stochastic gradient estimation as we consider multiple observations per update: $\tilde{\lambda}_{kw} = \eta + D/S \sum_{s \in B_i} n_{sw} \phi_{skw}$ where n_{sw} is the s-th document in the i-th mini-batch and D denote the number of documents. The rate of change ρ_t is set to $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$ where $\tau_0 \geq 0$ and $\kappa \in (0.5, 1]$.

3 Regularized Variational Bayes

For introducing a structured prior to regularize the word-topic probabilities, we are inspired by the recent regularized Gibbs (regGS) approach due to Newman et al. (2011), who have demonstrated that regularization improves the topic coherence. Specifically, we view each topic as a mixture of word probabilities given by the word-pair dependency matrix C (a $W \times W$ matrix, where W denotes the size of vocabulary and $C_{ij} \geq 0$), that is

$$\beta_k \propto C b_k \text{ where } b_k \sim Dir(\eta), \tag{1}$$

In VB the true posterior is approximated using fully factorized distributions q. Consequently, we parametrize the word probabilities b by introducing a new variational parameter ν, i.e. $q(b_k) = Dir(b_k, \nu_k)$. The per-word topic assignments z are parametrized by ϕ, and the posterior over the per-document topic weights θ are parametrized by γ, as for the standard LDA (Blei et al., 2003). With this, the part of the likelihood including the specific parameter ν can be written as

$$\begin{aligned}\mathcal{L}_{[\nu]} =& \mathbb{E}_q[\log p(w \mid z, C, b)] \\ &+ \mathbb{E}_q[\log p(b \mid \eta)] - \mathbb{E}_q[\log q(b)] \,. \end{aligned} \tag{2}$$

The remaining part of the ELBO does not change. To approximate the first term of Eq. (2) we adapt the lower bound on the log-sum-exp function (Boyd and Vandenberghe, 2004, page 72), $\mathbb{E}_q[\log \sum_i X_i] \geq \log \sum_i \exp(\mathbb{E}_q[\log X_i])$ (for a detailed proof see e.g. (Paisley, 2010)) to our case, which follows by applying Jensen's inequality: $\mathbb{E}_q[\log p(w \mid z = k, C, b)]$

$$\begin{aligned} &= \sum_i^W \Phi_{ik} \mathbb{E}_q[\log \sum_j^W C_{ij} b_{jk}] \\ &\geq \sum_i^W \Phi_{ik} \log \sum_j \exp(\mathbb{E}_q[\log C_{ij} b_{jk}]) \\ &= \sum_i^W \Phi_{ik} \log \sum_j C_{ij} \exp(\mathbb{E}_q[\log b_{jk}]) \end{aligned} \tag{3}$$

where $\sum_w^W \Phi_{wk} = \sum_d^D \sum_w^W \phi_{dwk}$. This is still a lower bound, so maximizing it will improve the ELBO. The expectation of $\log b$ under the distribution q is: $\mathbb{E}_q[\log b_{wk}] = \Psi(\nu_{wk}) - \Psi(\sum_s \nu_{sk})$. The remaining terms of the Eq. (2) (for a topic k) are

$$\begin{aligned} \mathbb{E}_q[\log p(b \mid \eta)]_{[k]} &= \log \Gamma(W\eta) - W \log \Gamma(\eta) \\ &\quad + \sum_w^W (\eta - 1)(\Psi(\nu_{wk}) - \Psi(\sum_s^W \nu_{sk})) \,, \\ \mathbb{E}_q[\log q(b)]_{[k]} &= \log \Gamma(\sum_s^W \nu_{wk}) - \sum_w^W \log \Gamma(\nu_{wk}) \\ &\quad + \sum_w^W (\nu_{wk} - 1)(\Psi(\nu_{wk}) - \Psi(\sum_s^W \nu_{sk})) \,. \end{aligned}$$

To derive a VB approach, we compute the derivative of Eq. (2) with respect to the variational parameter ν_{wk}. After applying the chain rule and rearranging terms, this gives e.g. $\partial \mathbb{E}_q[\log p(w \mid z, C, b)] / \partial \nu_{wk} =$

$$\begin{aligned} &= \Psi_1(\nu_{wk}) \sum_i \Phi_{ik} \frac{C_{iw} \exp(\mathbb{E}_q[\log b_{ik}])}{\sum_j C_{ij} \exp(\mathbb{E}_q[\log b_{jk}])} \\ &\quad - \Psi_1(\sum_s v_{sk}) \sum_i \Phi_{ik} \,. \end{aligned}$$

Taking the derivatives for all terms together we arrive at: $\partial \mathcal{L} / \partial \nu_{wk} =$

$$\begin{aligned} &\Psi_1(\nu_{wk})(\sum_i^W \Phi_{ik} \frac{C_{iw} \exp(\mathbb{E}_q[\log b_{ik}])}{\sum_j^W C_{ij} \exp(\mathbb{E}_q[\log b_{jk}])} + \eta - \nu_{wk}) \\ &- \Psi_1(\sum_s^W \nu_{sk}) \sum_i^W (\Phi_{ik} + \eta - \nu_{ik}) \,. \end{aligned} \tag{4}$$

Setting the above derivative to zero, we get the following fixed point update:

$$\nu_{wk} = \eta + \sum_i^W \Phi_{ik} \frac{C_{iw} \exp(\mathbb{E}_q[\log b_{ik}])}{\sum_j^W C_{ij} \exp(\mathbb{E}_q[\log b_{jk}]))} \ . \quad (5)$$

This is a proper generalization of the standard VB approach. To see this simply set the word-pair dependency matrix C to the identity matrix. Then, it follows that $\nu_{kw} = \eta + \sum_d n_{dw}\phi_{dwk}$; see also (Blei et al., 2003) for more details.

To derive a learning algorithm, i.e., to actually optimize $\mathcal{L}$, we follow a coordinate ascent on the variational parameters ϕ, γ and ν. Given the word topic probabilities β from Eq. (1), this yields the following per-document updates for ϕ and γ in the E-step:

$$\phi_{dwk} \propto \beta_{wk} * \exp(\mathbb{E}_q[\log \theta_{dk}]) \ , \quad (6)$$

$$\gamma_{dk} = \alpha + \sum_w^W n_{dw}\phi_{dwk} \ . \quad (7)$$

In the M step, we perform fixed point updates, as given in Eq. (5), and compute the values β_{wk} using Eq. (1) as follows:

$$\beta_{wk} \propto \sum_i^W C_{iw} \exp(\Psi(\nu_{ik}) - \Psi(\sum_s^W \nu_{sk})) \ . \quad (8)$$

However, recall that one of our main goals is the application of regularised VB to hyper-spectral images of plants. Since a single image can already consists of hundreds of thousands of signatures (documents) so that several images (as in our experiments) easily scale to several million documents, batch VB is likely to be overtaxed in terms of running time. Consequently, we will develop an online variant of regularized VB (regVB) that scales well to massive datasets.

4 Online Regularized VB

Since setting the word-dependency matrix C to identity matrix results in standard VB, it is intuitively clear that we can extend the regularized VB to the online case (regOVB) by adapting (Hoffman et al., 2010). Specifically, the variational lower bound for the regVB can be written as $\mathcal{L} =$

$$\sum_d^D \Big\{ \mathbb{E}_q[\log p(w_d \mid \theta_d, z_d, C, b)] + \mathbb{E}_q[\log p(z_d \mid \theta_d)] - \mathbb{E}_q[\log q(z_d)] + \mathbb{E}_q[\log p(\theta_d \mid \alpha)] - \mathbb{E}_q[\log q(\theta_d)] + (\mathbb{E}_q[\log p(b \mid \eta)] - \mathbb{E}_q[\log q(b)])/\mathbf{D} \Big\}$$

$$\triangleq \sum_d^D \ell(n_d, \phi_d, \gamma_d, \boldsymbol{C}, \boldsymbol{\nu}) \ , \quad (9)$$

where $\ell(n_d, \phi_d, \gamma_d, \boldsymbol{C}, \boldsymbol{\nu})$ is the dth document's contribution to the variational bound. The per-corpus terms

Algorithm 1: Online regularized LDA. The changes to online LDA are highlighted using blue fonts.

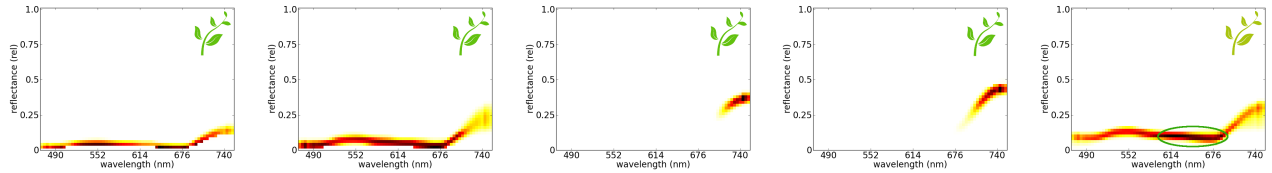
Input: D (documents), S (batchsize), $RegIter$ (fixed updates in M step), C (word-dependency matrix)

Define $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$ with $\kappa \in (0.5, 1]$;
Initialize $\boldsymbol{\nu}$ randomly and set $t = 0$;
repeat
  Select S documents randomly forming the mini-batch $\tilde{D}$;
  `/* Compute E step */`
  foreach *document* d *in* $\tilde{D}$ **do**
    repeat
      Set $\phi_{dwk} \propto \beta_{wk} * \exp(\mathbb{E}_q[\log \theta_{dk}])$;
      Set $\gamma_{dk} = \alpha + \sum_w^W \phi_{dwk} n_{dw}$;
    until $\frac{1}{K}\sum_k |change\ in\ \gamma_{dk}| < 0.00001$;
  `/* Compute M step */`
  Initialize $\tilde{\nu}$ randomly;
  for $i = 1 : RegIter$ **do**
    $\tilde{\nu}_{wk} = \frac{D}{S} \sum_{d \in \tilde{D}} \sum_i^W \phi_{dik} \frac{C_{iw} \exp(\mathbb{E}_q[\log b_{wk}])}{\sum_j C_{ij} \exp(\mathbb{E}_q[\log b_{jk}]))} + \eta$;
  Set $\boldsymbol{\nu} = (1 - \rho_t)\boldsymbol{\nu} + \rho_t \tilde{\boldsymbol{\nu}}$;
  $\beta_{wk} \propto \sum_i^W C_{iw} \exp(\Psi(\nu_{ik}) - \Psi(\sum_s \nu_{sk}))$;
  Increment $t := t + 1$;
until *converged* ;

are summed together and divided by the number of documents D. Doing so allows one to derive the online approach since the optimal ν is the one for which $\mathcal{L}$ maximized after fitting the per-document parameter. In other words, we can use the regularized updates in a per-document manner. This is summarized in Alg. 1. That is, we start off by randomly selecting documents form entire dataset by forming a mini-batch $\tilde{D}$. Then an *E step* is performed to find locally optimal values of γ and ϕ holding β fix. In the *M step* several fixed point updates for $\tilde{\nu}$ are computed using $\tilde{\nu}_{wk} =$

$$\frac{D}{S} \sum_{d \in \tilde{D}} \sum_i \phi_{dik} \frac{C_{iw} \exp(\mathbb{E}_q[\log b_{ik}])}{\sum_j C_{ij} \exp(\mathbb{E}_q[\log b_{jk}]))} + \eta \quad (10)$$

given the document-specific parameter ϕ_d with $d \in \tilde{D}$ (currently observed mini-batch), where we rescale by $\frac{D}{S}$ to update as though we would have seen all documents. Multiple documents are used per update to reduce variance. The parameter $\boldsymbol{\nu}$ is updated through a weighted average of its previous value, and $\tilde{\nu}$ (computed for the current mini-batch using fixed point updates as in Eq. (10)). Furthermore, the new values of β are computed given $\boldsymbol{\nu}$ and word-dependency matrix C. Following Hoffman et al. (2010), the rate of change ρ_t is set to $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$ with $\kappa \in (0.5, 1]$ in order to guarantee convergence. Note, as in the non-regularized case, we recover regularized batch VB when setting the batch size to $S = D$ and $\kappa = 0$.



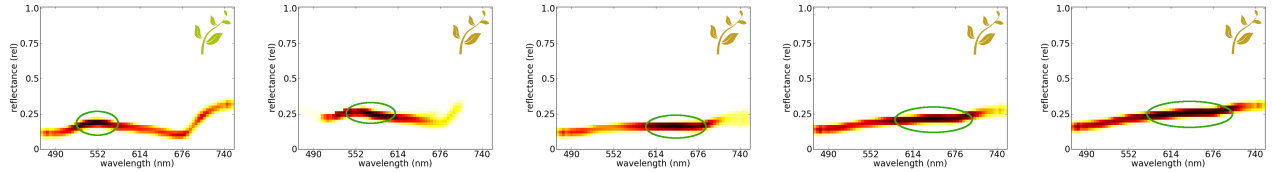
(a) Low reflectance for wavelengths <680nm with a small peak at 550nm, which indicate high chlorophyll content (Gitelson and Merzlyak, 1996)

(b) Low reflectance for wavelengths <680nm with a small peak at 550nm, which indicate high chlorophyll content (Gitelson and Merzlyak, 1996)

(c) "Red-Edge": High reflectance for near-infrared spectral bands >700. Sensitive to chlorophyll content (Curran et al., 1990; Seager et al., 2005)

(d) "Red-Edge": High reflectance for near-infrared spectral bands >700. Sensitiv to chlorophyll content (Curran et al., 1990; Seager et al., 2005)

(e) Top words for chlorophyll specific wavebands between 600nm-680nm (Blackburn, 1998).



(f) High sensitivity to chlorophyll concentration near wavelength 550nm (Blackburn, 2007; Gitelson and Merzlyak, 1996).

(g) High sensitivity to chlorophyll concentration near wavelength 550nm (Blackburn, 2007; Gitelson and Merzlyak, 1996).

(h) High reflectance for spectral bands between 600nm-680nm indicate low chlorophyll content (Gitelson and Merzlyak, 1996).

(i) High reflectance for spectral bands between 580nm-680nm indicate low chlorophyll content (Gitelson and Merzlyak, 1996).

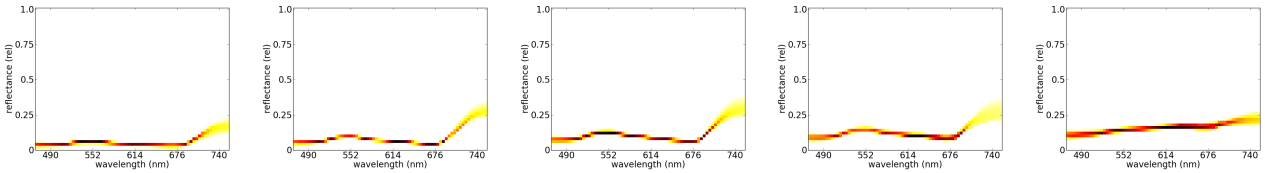
(j) High reflectance for spectral bands between 580nm-680nm indicate low chlorophyll content (Gitelson and Merzlyak, 1996).

Figure 2: Example topics learned by regularized online VB (K = 15). Most of the topics can be found to reflect results received by experts (see Table 1 in (Govender et al., 2009) and references in there). Top words (here in dark red) consists mostly of highly correlated spectral bands. Furthermore, given the reflectances one can identify whether a topic represent a healthy (e.g. low reflectance in chlorophyll specific bands, large reflectance in near infrared wavebands (Blackburn, 2007)) or non-healthy signature. (Best viewed in color)

5 Uncovering Spectral Drought Stress Characteristics

In our main experiment, drought stress was applied to barley cultivar Scarlett. Hyper-spectral images of the 5 stressed and 5 control Barleys were taken with a resolution of 640x640 pixels, where each pixel is a vector with 120 recorded wavelengths from the range of 394-891nm with approximately 4nm spectral resolution, using the SOC-700 hyper-spectral imaging system (Rascher et al., 2007), manufactured by Surface Optics. A normalization of the images was done by calculating the spectral reflectance for each pixel. For that, the spectrum of a pixel was divided by the spectrum of the incoming radiation estimated from a white reference panel that exhibits Lambertian reflectance located in each scene. Because the monitoring started with the flowering time of Barley, both the control and stressed Barleys developed senescent leaves. In our experiments we used 7 measurements for 10 plants, which were done every 3-4 day. Five control plants grew in a fully water capacity of the substrate conditions while the 5 stressed plants were exposed to 50% water supply reduction (at BBCH 30; BBCH is a scale used to identify the phenological development stages of a plant). This yielded 70 data cubes of resolution 640x640x120. Although the SOC-700 measured from 394nm to 890nm, the wavelengths below 470nm and above 750nm were discarded because they appeared to be very noisy. The reason for this is most likely an unstable source of illumination for these frequencies. Therefore, only the bandwidths from 470nm to 750nm were used. We transform each cube into a dense 'pixel x spectra' matrix. This resulted in 70 data matrices of soze 69x409,600; overall a dataset with about 30 Million signatures.

Furthermore, for our analysis we are interested in finding not just specific wavelength patterns but also reflectance values. For that reason, and to get the data sparse, we discretize the corresponding signatures in the following way: we decompose the space covering the full signatures (with $Wl = 69$ spectral bands) additionally into $R = 50$ possible reflectance words. Thus, in a signature each wavelength can consist of one of the R distinct reflectance words, which results in a total number of $Wl \times R$ different possible *spectral words*. The use of discretised values instead of continuous is also motivated by the fact that according to plant physiologist small difference in reflectance values are not of great importance. They instead study the spectral characteristics of plants in order to get spectral



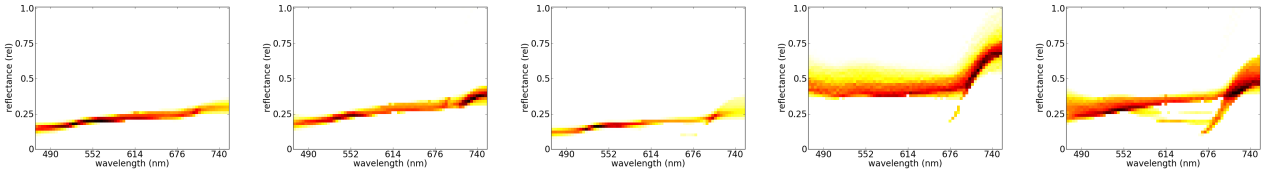
(a) Low reflectance for wavelengths <680nm with a small peak at 550nm, which indicate high chlorophyll content (Gitelson and Merzlyak, 1996)

(b) Low reflectance for wavelengths <680nm but **no** small peak at 550nm

(c) Peak at 550nm but **no** low reflectance for for wavelengths <680nm

(d) Top words for chlorophyll specific wavebands between 600nm-680nm **but** also for lower waveband

(e) High reflectance for spectral bands between 600nm-680nm indicating low chlorophyll content **but** also for other spectral bands



(f) Increasing slope but high reflectance for spectral bands **below** 600nm-680nm.

(g) **Mixture** of "red edge", peak at 550nm and increasing slope.

(h) **Mixture** of top words for chlorophyll specific wavebands between 600nm-680nm and peak at 550nm.

(i) Junc topic with elements of "red edge"

(j) Junc topic with elements of "red edge" and peak at 550nm

Figure 3: Topics learned by standard online VB (K = 15). As one can see the topics are less distinct compared to the regularized ones and actually mix known indices in particular for stressed and controlled plants.

indices more sensitive to pigment content. Furthermore, we excluded all biologically implausible signatures (*"background"* signatures) using Simplex Volume Maximization (SiVM) (Thurau et al., 2012) algorithm, by taken out all signatures with a high coefficient to an extreme spectra which was identified as *"background"* (non-leaf). This resulted in a dataset with about $9,2$ Million signatures (documents).

The word-dependency matrix C was created using pointwise mutual information (PMI) (Newman et al., 2011). For plants, in order to also get the cooccurrences between different reflectance's within a wavelength, we proceeded as follows: we first aggregated the signatures in the images within each non-overlapping squares of 5x5 pixel. *Spectral word* cooccurrences were computed using a sliding window of length 1 in each direction (wavelength and reflectances) in the aggregated signatures. We compute the PMI only for the 1000 words with the highest frequency. The matrix C was computed from 8 plants (56 images, with approx. $7,3$ Million signatures), for the remaining two plants (one control and one stressed, with approx. $1,9$ Million signatures) an regularized/non-regularized online LDA was learned on non aggregated signatures. For both methods we set the batchsize $S = 1024$, κ close to 0.5, $\alpha = 0.01$, $\eta = 0.01$ and (for the regularized online VB) 10 fixed point iterations in the M step. For both methods we stopped when each signature (document) was seen once. The computation took for both methods less than one hour (for $K = 15$) on a standard Intel-Quadcore 3.4 GHz computer, but only using a single core. In contrast, running batch LDA on two plants took more than 15 hours.

5.1 Uncovered Characteristics

Fig. 2 shows examples of topics discovered by regularized online VB (K = 15). The topics conform to common plant physiological knowledge (see also Table 1 in (Govender et al., 2009) and references in there). For example topics (a), (b), (f) and (g) clearly show that high probable words appear for the spectral band 550nm. This wavelength was found to have a maximum sensitivity to a wide range of chlorophyll contents (Blackburn, 2007; Gitelson and Merzlyak, 1996), which are the most important pigments as they are necessary for photosynthesis. Furthermore, topics (a) and (b) show a high probability wavelengths close to 500nm and 670nm. As mentioned by Gitelson and Merzlyak (1996), these bands/wavelength have high correlation for yellow-green to dark-green leaves. Additionally Gitelson and Merzlyak report that the reflectance in 550nm does not exceed 0.1 in green leaves (for Maple and Chestnut). This is also supported by topics (a) and (b) but now for Barley. An up-rising slope between bands 690nm and 750nm is known as the "red edge" and is due to the contrast between the

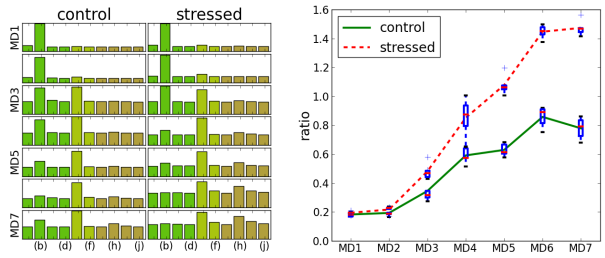
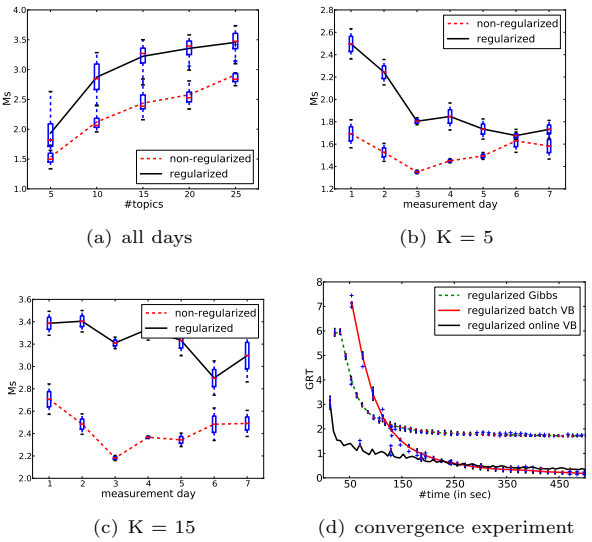


Figure 4: (**Left**) Evolution of "non-healthy" and "healthy" topics shown in Fig.2 over time. Each row stands for different measurement days 1-7. The larger the bar the more prominent a topic is. As one can see, the most prominent topics for control plants consists of "healthy" yellow-green topics, whereas for the stressed plants the probabilities of "healthy" topics drop rapidly and "stress" topics (brown) become more likely in latter days. (**Right**) The ratio of relative importance of a "non-healthy" and "healthy" topic. Beginning by measurement day 3 we have a significant difference (paired t-test, p = 0.05) between stressed and control plants. (Best viewed in color)



(a) all days (b) K = 5

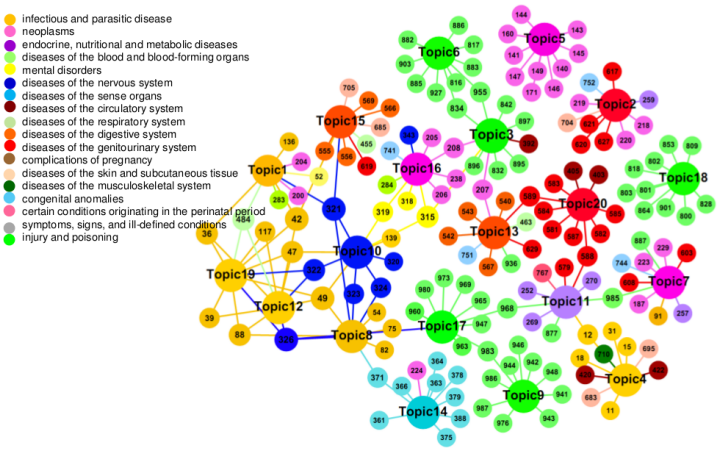
(c) K = 15 (d) convergence experiment

Figure 5: Regularization produces more specific topics. Shown are the averages per topic KL-distance to a "background topic" when taking (a) all days as a function of the number of topics, (b) for 5 topics and (c) 15 topics as a function of measurement days. The topic proportions of the signatures from each images were taken separately in order to compute the distance to "background topic". As one can see the regularized topics are more specific than non-regularized ones; they have significantly larger KL-distance. Interestingly, for the last three measurement days the picture gets more diverse. Actually, regularized topics capture the spectral characteristics less well for smaller (b) than for large number of topics (c). This indicates that their is a diverse set of spectral characteristics required to capture drought stress. Furthermore, (d) regularized VB converges significantly faster than regularized semi-collapsed Gibbs-Sampling (Wikipedia, K = 20). Shown are the expected change of word probabilities in topics over time. This speed-up puts high throughput hyper-spectral topic models of several plants per day for several weeks in reach. (Best viewed in color)

strong absorption of chlorophyll and the otherwise reflective leaf (Seager et al., 2005). The "red edge" is detected by topics (c) and (d). The discovered important wavelengths in the topics (h)-(j) (between 580nm-680nm) also closely mirror known indices. Blackburn (1998) define the "optimal" individual bands for pigment estimation as 680 nm for chlorophyll a and 635 nm for chlorophyll b. Gitelson and Merzlyak (1996) observed a increase of reflectance for the band 670nm if the amount of chlorophyll in the leaf drops, as it is the case for stressed plants. Taking both results together, topics (h)-(j) clearly describe "non-healthy" resp. "drought-stressed" topics and also conform to plant physiological knowledge.

The discovered topics/indices can be used in various ways to investigate drought stress reactions. For instance, we can computed the distributions of topics for plants over time. Fig. 4 (**left**) shows this topic distribution over all the measurement days 1-7 (recall that images were taken every 2-3 days so we cover several weeks). Since, LDA is giving us a topic distribution per signature only, we actually estimated the concentration parameters of the Dirichlet distribution over the topics induced by all signatures of single images (Minka, 2000). Then, we used the expected probability of each topic for visualization.

As one can see, the probability of "non-healthy" topics increases for the stressed plant over time, whereas for the control plant "healthy" yellow-green to green topics have higher weights. Furthermore, Fig. 4 (**right**) show the ratio between the expected probability of a "non-healthy" and "healthy" topic (ratio = h/b, for topics (b) and (h) in Fig. 4 (**left**)). For this experiment we also computed the distribution of topics for all plants and measurement days using the model shown in Fig. 2. Beginning by measurement day 3 this ratio shows a significant difference (paired t-test, p = 0.05) between stressed and control plants. This are yet another validations that the discovered topics capture drought stress relevant characteristics.



Topic 20: **kidney disease**	
581	Nephrotic syndrome
583	Nephritis and nephropathy
588	Disorders resulting from impaired renal function
403	Hypertensive kidney disease
585	Chronic kidney disease (ckd)
405	Secondary hypertension
589	Small kidney of unknown cause
582	Chronic glomerulonephritis
587	Renal sclerosis unspecified
584	Acute kidney failure
Topic 8: **viral infections**	
326	Late effects of intracranial abscess or pyogenic infection
49	Other non-arthropod-borne viral diseases
75	Infectious mononucleosis
323	Encephalitis myelitis and encephalomyelitis
321	Meningitis due to other organisms
88	Other arthropod-borne diseases
54	Herpes simplex
82	Tick-borne rickettsioses
371	Corneal opacity and other disorders of cornea
324	Intracranial and intraspinal abscess

Figure 6: Regularized VB LDA can discover composite topics in Human Disease Network: (Left) A graph representing topics learned by regularized VB (K = 20). Topics (big circles) are connected with top 10 diseases (small circles). The different colors indicate different categories of ICD9 codes. As one can see, most of the topics are dominated by one of the categories. But there are also mixed topics of common diagnosed diseases across several categories. (Right) Two example topics learned with regularized VB LDA about "kidney diseases" and "viral infections". (Best viewed in color)

5.2 Regularization improves the topic coherence

To accommodate the qualitative results so far with quantitative ones, we compared the results produced by regularized and non-regularized oLDA in terms of their topic coherence. A topic is specific or has authentic identity if it is far from generating words in wide range of documents. To measure this, Alsumait et al. (2009) define a "background topic" to be a topic that is equally probable in all the documents, i.e., $P(d_m \mid \xi) = 1/D$ for $m \in (1, 2, ..., D)$. In turn, the distance between a topic and the "background topic" can be viewed as measure for how much background or common the information is provided by the topic. Since we are interested in overall performance of a model, we measured the average per topic distance to the "background topic" using KL-Divergence:

$$Ms = \frac{1}{K} \sum_k D_{kl}(\boldsymbol{\theta}_k, \xi) \text{ where } \boldsymbol{\theta}_k = (\theta_{1k}, \theta_{2k}, ..., \theta_{Dk}) \ .$$

The results are summarized in Fig.5 (a)-(c). As one can see, the regularized topics are more specific than non-regularized ones (a), i.e. they have larger KL-distance . More interestingly, for the last three measurement days regularized online LDA does worst for lower number of topics (b) than for large number of topics (c). This can be due to wide range of different signatures in the images (in terms of "non-healthy" signatures). This can be also seen in Fig.4 where for the first days there just a few topics with high probability, whereas for the latter days more topics are needed to represent the data.

6 Supplementary Evaluation

To further investigate the performance of regularized VB works even if LDA fails to learn anything meaningful, and that it is comparable with semi-collapsed Gibbs Sampling inference by Newman et al. (2011) we provide experiments with two additional datasets.

Human Disease Network: Here we investigate the question: can regularized VB learn meaningful and interpretable topics models even if "no information" in documents are available? To answer this we make use of human disease network dataset (Hidalgo et al., 2009), which consist originally of approximately 32 Million inpatient claims, pertaining 13039018 individuals of 65 years and older patients. The corresponding human disease network includes cooccurrences of diagnosis (specified by ICD9 codes) and up to 9 secondary diagnosis.

Here we applied regularized VB by using disease dependency matrix which was computed using PMI. We used only diseases with more than 500 occurrences, resulted in a total of 742 diseases. In order to learn a regularized topic model, we created a synthetic datasets of D = W documents each consisting of one word (actually an identity matrix with $W \times W$ dimensionality). We used the following settings: $\alpha = 0.01$, $\eta = 0.01$ and 10 fixed point iterations in the M step. The results of the regularized VB (for K = 20) are shown in Fig.6. As one can see, regularized VB LDA can discover coherent topics in human disease network, even if LDA would fail. Topics (big circles) are connected with top 10 diseases (small circles). The different colors indi-

cate different categories of ICD9 codes. As one can see, most of the topics are dominated by one of the categories. But there are also mixed topics of common diagnosed diseases across several ICD9 categories, as shown in the table on Fig.6, with topic about "kidney diseases" and "viral infections".

Wikipedia Dataset: Further, to compare regularized VB with regularized Gibbs Sampling we used a small set of D = 5000 Wikipedia articles, with W = 7312 words in the vocabulary and N ≈ 750000 total number of terms. The word cooccurrences were computed using an extern data of 3441010 titles of Wikipedia articles. The titles include naturally the short range word dependencies of words. Here we used the normalized cooccurrences (matrix C) for all regularized methods. We set $\alpha = 0.05\frac{N}{DK}$ (as suggested by Newman et al. (2011)), $\eta = 0.01$ and the number of topics was set to K = 20, and in each M step 10 fixed point iterations were applied. For the online case the batchsize was set to S = 500, κ close to 0.5 and $\tau_0 = 1024$. The regularized Gibbs LDA was run for 1250 iterations where we applied regularization (10 fixed point updates) every 50 iterations. The VB methods were run until each document was seen 50 times. The results are shown in Table 1. As one can see, all methods produce topic models of similar quality in terms of the interpretability. Additionally, to show convergence of the different methods, we computed the change of the probabilities of words $\boldsymbol{\beta}$ between two iterations when $\boldsymbol{\beta}$ was computed. To measure this, we use: $GRT = \sum_w \sum_k \mid \beta_{kw}^t - \beta_{kw}^{t-1} \mid$, where t indicate the current iteration. The results are represented in Fig.5 (d). As one can see, the regularized batch VB converges faster than semi-collapsed Gibbs-Sampling. Moreover, regularized online VB outperforms regularized batch VB and is comparable in terms of the interpretability of the topics.

music		
regGS	music, band, song, released, live, new, single, rock, songs, records	+
regVB	music, band, song, released, live, new, songs, single, rock, records	+
regOVB	music, band, song, released, single, songs, rock, live, records, track	+
league		
regGS	league, team, season, game, first, years, games, club, two, new	+
regVB	league, team, club, world, cup, years, first, national, won, season	+
regOVB	league, season, team, club, years, new, state, career, played, born	+
university		
regGS	university, new, american, college, school, science, research, professor, national, born	+
regVB	university, new, united, born, states, american, national, school, first, party	+
regOVB	university, book, research, professor, science, published, work, new, first, school	+
church		
regGS	school, high, church, new, schools, district, year, students, education, college	+
regVB	church, war, king, first, century, great, new, history, city, catholic	+
regOVB	church, house, building, century, city, town, new, village, built, old	+

Table 1: Gibbs, batch VB and online VB for regularized LDA produce qualitatively comparable topics: An example of topics (for Wikipedia Dataset) represented by 10 words with highest weights (K = 20).

7 Conclusion

Understanding drought stress in plants is not an easy task. In this context, hyper-spectral image sensors are an established, sophisticated method for discovering spectral stress indices. However, they gather massive, high dimensional data clouds over time, which together with the demand of physical meaning of the prediction model present unique computational problems in scale and interpretability. Motivated by this, we developed a regularized variational Bayes approach to latent Dirichlet allocation and presented the — to the best of our knowledge — the first application of probabilistic topic models to discovering drought stress characteristics from hyper-spectral image sequences. Our experimental results on a large-scale plant phenotyping dataset demonstrate that the estimated spectral characteristics are meaningful, conform to existing plant physiological knowledge, and are fast to compute. In contrast to indices established in plant physiology, topics are not based on single or few wavelengths but provide a distributional view on the characteristics of complete signatures. Overall, our results are an encouraging sign that the vision of high throughput precision phenotyping is not insurmountable. Detailed measurements of plant characteristics can be analysed at massive scale to collectively provide estimates of trait phenotypes for many of the underlying genotypes that comprise a typical plant breeding population.

Our work provides several interesting avenues for future work. Next to experiments under field conditions e.g. in an experimental agricultural site, one should aim at improving the topics quality even further by applying hierarchical, (semi-)supervised and relational versions of topic models. Active LDA approaches could speed up computations even further. Ultimately, the models should be used to identify the most relevant moment when biologists have to gather samples for invasive, molecular examinations.

Acknowledgements: The authors thank the anonymous reviewers for their valuable comments, Edwin Bonilla, Wray Buntine, and Zhao Xu for helful discussions on regluarized LDA, and Anja Pilz and Hannah Lanzrath for helping with the data for the supplementary evaluation. The work was partially supported by the Fraunhofer ATTRACT fellowship STREAM and by the German Federal Ministry of Education and Research BMBF/315309/CROP.SENSe.

References

A. Abdeen, J. Schnell, and B. Miki. Transcriptome analysis reveals absence of unintended effects in drought-tolerant transgenic plants overexpressing the transcription factor abf3. *BMC Genomics*, 11, 2010.

L. Alsumait, D. Barbará, J. Gentle, and C. Domeniconi. Topic significance ranking of lda generative models. In *Proceedings of ECML*, pages 67–82, 2009.

G. A. Blackburn. Quantifying chlorophylls and carotenoids at leaf and canopy scales: an evaluation of some hyperspectral approaches. *Remote Sens. Environ*, 66:273–285, 1998.

G. A. Blackburn. Hyperspectral remote sensing of plant pigments. *Journal of Experimental Botany*, 58(4):855–867, 2007.

D.M. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

J.S. Boyer. Plant productivity and environment. *Science*, 218:443–448, 1982.

P.J. Curran, J.L. Dungan, and H.L Gholz. Exploring the relationship between reflectance red edge and chlorophyll content in slash pine. *Tree Physiology*, 7:33–48, 1990.

A.A. Gitelson and M.N. Merzlyak. Signature analysis of leaf reflectance spectra: algorithm development for remote sensing of chlorophyll. *Plant Physiol.*, 148:494–500, 1996.

M Govender, P J Dye, I M Weiersbye, E T F Witkowski, and F Ahmed. Review of commonly used remote sensing and ground-based technologies to measure plant water stress. *WaterSA*, 35(5):741–752, 2009.

P. Guo, M. Baum, S. Grando, S. Ceccarelli, G. Bai, R. Li, M. von Korff, R.K. Varshney, A. Graner, and J. Valkoun. Differentially expressed genes between drought-tolerant and drought-sensitive barley genotypes in response to drought stress during the reproductive stage. *Journal of Experimental Botanic*, 60:3531–3544, 2010.

C.A. Hidalgo, N. Blumm, A.L. Barabasi, and N.A. Christakis. A dynamic network approach for the study of human phenotypes. *PLoS Comput Biol*, 5 (4), 04 2009.

M. Hoffman, D.M. Blei, and F. Bach. Online learning for latent dirichlet allocation. In *Proceedings of Neural Information Processing Systems (NIPS-10)*, 2010.

K. Kersting, M. Wahabzada, C. Roemer, C. Thurau, A. Ballvora, U. Rascher, J. Leon, C. Bauckhage, and L. Pluemer. Simplex distributions for embedding data matrices over time. In I. Davidson and C. Domeniconi, editors, *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*, Anaheim, CA, USA, April 26–28 2012.

C. Lebreton, V. Lazic-Jancic, A. Steed, S. Pekic, and S.A. Quarrie. Identification of qtl for drought responses in maize and their use in testing causal relationships between traits. *Journal of Experimental Botanic*, 46:853–865, 1995.

M.W. Mahoney and P. Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 106(3):697–703, 2009.

J.K. McKay, J.H. Richards, S. Sen, T. Mitchell-Olds, S. Boles, E.A. Stahl, T. Wayne, and T.E. Juenger. Genetics of drought adaptation in arabidopsis thaliana ii. qtl analysis of a new mapping population , kas-1 x tsu-1. *Evolution*, 62:3014–3026, 2008.

T. Minka. Estimating a Dirichlet distribution. In *A note publically available from the author's homepage*. 2000.

D. Newman, E. Bonilla, and W. Buntine. Improving topic coherence with regularized topic models. In *Proceedings of NIPS*, 2011.

J. Paisley. Two useful bounds for variational inference. Technical report, Department of Computer Science, Princeton University, Princeton, NJ, 2010.

J.B. Passioura. Environmental biology and crop improvement. *Functional Plant Biology*, 29:537–554, 2002.

E. Pinnisi. The blue revolution, drop by drop, gene by gene. *Science*, 320:171–173, 2008.

M.A. Rabbani, K. Maruyama H. Abe, M.A. Khan, K. Katsura, Y. Ito, K. Yoshiwara, M. Seki, K. Shinozaki, and K. Yamaguchi-Shinozaki. Monitoring expression profiles of rice genes under cold, drought, and high-salinity stresses and abscisic acid application using cdna microarray and rna gel-blot analyses. *Plant Physiology*, 133:1755–1767, 2010.

U. Rascher and R. Pieruschka. Spatio-temporal variations of photosynthesis: The potential of optical remote sensing to better understand and scale light use efficiency and stresses of plant ecosystems. *Precision Agriculture*, 9:355–366, 2008.

U. Rascher, C.L. Nichol, C. Small, and L. Hendricks. Monitoring spatio-temporal dynamics of photosynthesis with a portable hyperspectral imaging system. *Photogrammetric Engineering and Remote Sensing*, 73:45–56, 2007.

R.A. Richards, G.J. Rebetzke, M. Watt, A.G. Condon, W. Spielmeyer, and R. Dolferus. Breeding for improved water productivity in temperate cereals: phenotyping, quantitative trait loci, markers and the selection environment. *Functional Plant Biology*, 37(2):85–97, 2010.

C. Römer, K. Bürling, T. Rumpf, M. Hunsche, G. Noga, and L. Plümer. Robust fitting of fluorescence sprectra for presymptomatic wheat leaf rust detection with Support Vector Machines. *Computers and Electronics in Agriculture*, 79(1):180–188, 2010.

T. Rumpf, A.-K. Mahlein, U. Steiner, E.-C. Oerke, and L. Plümer. Early Detection and Classification of Plant Diseases with Support Vector Machines Based on Hyperspectral Reflectance. *Computers and Electronics in Agriculture*, 74(1):91–99, 2010.

S. Seager, E.L. Turner, J. Schafer, and E.B. Ford. Vegetations red edge: A possible spectroscopic biosignature of extraterrestrial plants. *Astrobiology*, 5(3): 372–390, 2005.

J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Sparse graph mining with compact matrix decomposition. *Statistical Analysis and Data Mining*, 1(1):6–22, 2008.

C. Thurau, K. Kersting, M. Wahabzada, and C. Bauckhage. Descriptive matrix factorization for sustainability: Adopting the principle of opposites. *Journal of Data Mining and Knowledge Discovery*, 24(2):325—354, 2012.

M. Wahabzada and K. Kersting. Larger residuals, less work: active document scheduling for latent dirichlet allocation. In *Proceedings of ECML PKDD*, 2011.

Dynamic Teaching in Sequential Decision Making Environments

Thomas J. Walsh
Center for Educational Testing and Evaluation
University of Kansas
Lawrence, KS 66045

Sergiu Goschin
Department of Computer Science
Rutgers University
Piscataway, NJ 08854

Abstract

We describe theoretical bounds and a practical algorithm for teaching a model by demonstration in a sequential decision making environment. Unlike previous efforts that have optimized learners that watch a teacher demonstrate a static policy, we focus on the teacher as a decision maker who can dynamically choose different policies to teach different parts of the environment. We develop several teaching frameworks based on previously defined supervised protocols, such as Teaching Dimension, extending them to handle noise and sequences of inputs encountered in an MDP. We provide theoretical bounds on the learnability of several important model classes in this setting and suggest a practical algorithm for dynamic teaching.

1 Introduction

In situations where one agent teaches another, the AI community has largely focused on teachers that demonstrate a static policy to the learning agent [Abbeel and Ng, 2005; Walsh et al., 2010]. However, demonstration of even an optimal policy is often not the best way to teach. For instance, consider a learner being trained to play billiards by observing a near-optimal player. Since a hallmark of good play is simplifying the next shot, an optimal player will likely only demonstrate easy shots. But if we really want to teach the learner to shoot pool, we need to show it difficult shots and novel situations that the optimal policy will rarely encounter. More generally, teachers can improve learning efficiency by showing highly informative examples that a static policy might not often encounter. In this paper, we show how to cast the problem of optimal teaching as a decision problem in its own right, and provide bounds on the teachability of several important model classes in Reinforcement Learning (RL) [Sutton and Barto, 1998].

Our approach uses lessons from the supervised-learning community, which has established a number of frameworks for quantifying the teachability of a domain class. Specifically, we extend the classical *Teaching Dimension* framework (TD) [Goldman and Kearns, 1992] and the recently described *Subset Teaching Dimension* (STD) [Zilles et al., 2011], which quantify the teachability of deterministic hypothesis classes. We adapt these frameworks for the type of data encountered in RL, particularly handling noise and sequential inputs. This allows us to characterize the teachability of several RL concept classes such as Bernoulli distributions, k-armed bandits and DBNs in a supervised setting.

These supervised algorithms and analyses form the foundation for our teaching algorithms in the sequential decision making setting, where teachers are constrained in the examples they can select by the dynamics of their environment and their current state. We show how to cast the problem of optimal teaching as a decision problem in its own right and that exact optimization is often intractable. We then use our results from the supervised setting to construct approximations to the optimal teaching strategy that are successful in practice. The end result is a general algorithm for a teacher in a Markov Decision Process (MDP) [Puterman, 1994] that, in contrast to work on demonstrating static policies, dynamically chooses highly informative examples to teach the model.

2 Alternate Approaches

We now describe competing protocols for teaching in the RL setting. We will cover previous work on teaching in the supervised setting in later sections.

Static policies for teaching RL agents have been studied in Inverse Reinforcement Learning (IRL) [Abbeel

and Ng, 2004, 2005] and Apprenticeship Learning [Walsh et al., 2010]. However, in both of these settings, the emphasis is on better learning algorithms to interpret a static teaching policy repeated over many episodes. By contrast, we are providing algorithms for the teaching side so that the teacher can show trajectories built to teach, not just to demonstrate. Also, learning efficiency with static demonstrators is measured by either the time needed to achieve the same policy as the teacher (IRL) or perform as well or better than the teacher (Apprenticeship). In our current work we will focus on algorithms to teach the *entire model* of a given environment rather than a specific policy. We note this is a different problem because the learner cannot rely on a policy bias towards the teacher demonstrations.

There has been significant work on human teachers guiding RL agents, including by direct demonstration [Argall et al., 2009] and providing shaping rewards [Knox and Stone, 2010]. Most of these works focus on either optimizing the learner for human inputs or studying how different human interfaces or reward signals impact the learner. An area of future work is studying how similar our optimal teaching policies are to human teachers, as results indicate these behaviors may be very different [Khan et al., 2011].

In some domains, teaching a model may be less efficient than directly teaching the optimal policy or value function. For instance, in the bandit setting, teaching the model (the payout of all arms) requires pulling each arm a number of times, but teaching the optimal policy directly might require (if the learner knows only one arm is optimal) only pulling the optimal arm once. This situation is similar to behavioral cloning [Bain and Sammut, 1995; Khardon, 1999], where supervised learning techniques are employed in the policy space, though the focus of that field is on learning algorithms, not teaching. We focus on teaching a model because in large structured domains, the number of parameters in the model is usually much smaller than the number of parameters needed to encode the value function or policy, though our techniques could be extended to teach in the policy space.

3 Teachers and Learners

In this section, we begin by describing several teaching frameworks from the supervised-learning literature. We then describe modifications to these protocols to accommodate teaching in an RL setting. Specifically, we need to account for (1) noise in the labels and (2) sequences of instances rather than sets. We deal with the first problem in this section and then describe several domain classes that can be taught in this noisy supervised model.

3.1 Supervised Teaching Frameworks

We now describe two frameworks that can be used to measure the sample complexity of teaching in a supervised setting where a teacher is picking the examples. These frameworks differ considerably in how much the teacher and learner can infer about one another. First, we consider the classical *Teaching Dimension* model [Goldman and Kearns, 1992] where the teacher is providing helpful examples to the learner, but the learner is unaware that it actually has a teacher.

Definition 1. *The* Teaching Dimension *of a concept class* $\{c \in C\} : X \mapsto Y$ *over instance space* X *and labels* Y *is* $TD(C) = \max_{c \in C} \min_{|S|}(Cons(S, C) = \{c\})$, *with* $Cons$ *being the concepts in* C *consistent with* S. *This makes* S *a* teaching set *TS(c,C) for c.*

Intuitively, TD represents the minimum number of examples needed to uniquely identify any $c \in C$. Because the learner may not know it is being taught and the teacher does not know what learner it is providing samples to, TD teachers may not act optimally for a specific learner. However, a natural extension is a protocol where the learner and teacher both understand they are interacting with a shared goal (for the learner to realize a concept). Such notions were formalised first by Balbach and Zeugmann [2009] who devised a protocol where the learner would make inferences about hypotheses that the teacher was clearly not trying to teach based on the size of the teaching set. This reasoning between the teacher and learner was further formalised in the *Subset Teaching Dimension* (STD) [Zilles et al., 2011] where the learner and teacher both assume the other is optimal.

Definition 2. *The* Subset Teaching Dimension *(STD) of a hypothesis class* $\{c \in C\} : X \mapsto Y$ *(*$STD(C)$*) is equal to the worst case number of samples needed to teach a learner when both the teacher and the learner know that the other is performing their task optimally. This is done by iteratively constructing Subset Teaching Sets* $STS^i(c, C)$ *starting from* $STS^0(c, C) = TS(c, C)$, *such that* $STS^i \subseteq STS^{i-1}$ *and* $STS^i(c, C)$ *is not a subset of* $STS^{i-1}(c', C)$ *for* $c' \neq c$ *(a "consistent subset").*

Intuitively, STD captures the fixed point behavior of a learner and teacher that recursively assume the other is doing their job optimally. One can think through the process of reaching this fixed point iteratively starting with the teaching sets for all concepts in the original TD framework, denoted $\text{STS}^0 = TS$. In the first step of reasoning, the learner assumes it is being taught using STS^0, and therefore it can assume the instances it will see will come from an optimal teaching set. This

in turn allows it to notice when it has seen subsets of instances that must be leading to one (and only one) set from STS^0. The next step of reasoning goes back to the teacher, who infers it can teach using one of these subsets from STS^1. The process repeats until there are no more changes to STS. The uniqueness and reachability of this fixed point is detailed by Zilles et al. While the reasoning process may be complicated, the resulting behavior can be quite intuitive. For instance, in the original TD setting, when teaching any singleton concept over n variables with the empty set also a possibility, n negative instances are needed to show the empty hypothesis is correct. However in STD, all hypotheses can be taught using just one example (positive for the singleton, negative for empty).

3.2 Teaching Frameworks for Noisy Concepts

In this section, we extend the protocols above to the setting where the concept being learned is noisy. We begin by defining a stochastic concept and unordered collection (a "set" that may contain duplicates):

Definition 3. *A* stochastic concept $c : X \mapsto D(Y)$ *maps each possible instance $x \in X$ to a distribution over the label space ($D(Y)$). An* unordered example collection *U of a concept c consists of (potentially overlapping) pairs of inputs and labels $\{x_0, y_0 ... x_n, y_n\}$ with each y_i drawn from $c(x_i)$.*

Next, we need to address the consistency of the learner, which can no longer be expected to predict the exact label of every instance. Instead, we consider learners that predict a distribution of labels for a given input.

Definition 4. *A* distribution consistent learner *with parameters ϵ and δ makes predictions for input x based on the current unordered collection U in the form of a predicted distribution $D'(x)$ over the label space Y. Consistency means $||\hat{D}(x) - D'(x)||_{TV} \leq \epsilon$ with probability $1 - \delta$, where $|| \cdot ||_{TV}$ is the total variation (half the L1-norm) over the labels and $\hat{D}$ is the distribution observed in U (in most cases $\hat{D}$ is the distribution determined by the maximum-likelihood estimate).*

Now we can extend the notion of a teaching set from Definition 1 in three ways. First, instead of sets, we consider collections with duplicates. Second, we assume the teacher does not control the label associated with a given input, but can see the label (produced by the true concept) once an instance is added to the teaching collection. Therefore, the teacher can choose instances one at a time to add to the collection and always knows how all the current examples have been labeled. In practical terms, the teacher will then be able to choose when to execute a *stop* action to declare the collection finished *but* notice that the learner will not be aware of the order in which examples were added. Finally, we say such a collection is a teaching collection (denoted TS in an abuse of notation) if any distribution consistent learner must have $||D(x) - \hat{D}(x)|| \leq \epsilon$, with probability $1 - \delta$ after seeing that sequence.

We now have all of the components to define the noisy teaching dimension (NTD):

Definition 5. *The* noisy teaching dimension (NTD) *with parameters ϵ and δ of a stochastic concept class C (*NTD*(C)) is the maximum size minimum teaching collection $\tau \in TS(c)$ over all concepts $c \in C$. That is,* $\max_c min_{\tau \in TS(c)} |\tau|$

Next, we consider the case of a Noisy Subset Teaching Dimension (NSTD), which will extend STD. Here, we need to redefine the notion of subset used in the original STD to account for duplicates and the incremental construction of the teaching collection. To do so we introduce the *consistent subcollection* relation:

Definition 6. *A collection U' is a* consistent subcollection *of another collection U ($U' \subseteq U$) if (1) every element of U' is mapped by a function to an element of U and (2) the range and probability of distributions represented by U and U' are the same.*

Replacing the original "consistent subset" requirement from Definition 2 with a requirement that each iteration of reasoning must produce a consistent subcollection and assuming the learner is distribution consistent gives us a full definition of NSTD. In most cases the original NTD collection (NSTD^0) will not be improved upon in the worst case, but in many cases the teachers ability to *stop* constructing the collection will cause a significant decrease in the *expected* teaching time for NSTD over NTD (see Theorems 2 and 3). While the definitions above assumes the teacher's construction ordering is hidden from the learner, we will later see the sequential setting reveals this order and allows stronger inference.

4 Concept Classes

We now describe several concept classes that are relevant for RL agents. Our analysis in this section is carried out in the supervised setting.

4.1 Monotone Conjunctions

Conjunctions of terms are a simple but important model for pre-conditions of actions and simple conditional outcomes. A monotone conjunction over an input space of n boolean terms is labeled 1 if all of the relevant variables are 1, and 0 otherwise. In the TD setting, the complexity of learning a monotone conjunction is $O(n)$ [Goldman and Kearns, 1992], but the types of examples are slightly different than in the

classical mistake-bound [Littlestone, 1988] case, which only requires positive examples. This is because in TD we cannot assume anything about how the learner defaults in its predictions. Instead, the TD teacher should present the most specific positive example (only relevant variables set to 1), and then negative examples with each *relevant* variable alone set to 0.

In the STD protocol, the bound actually becomes 1 [Zilles et al., 2011]. The optimal STD strategy is to show the positive example from the TD case because the learner can infer all of the other negative examples. These results provide intuition about how preconditions or conditional effects can be taught in the sequential setting, a topic we return to in Section 5.3.

4.2 Coin Learning

One of the fundamental noisy hypothesis classes in model-based RL is the Bernoulli distribution learned through observation outcomes, (i.e. determining the bias (p^*) of a coin by observing flips). In the NTD case, because the teacher has to provide samples that may be interpreted by any type of consistent learner, the following strategy produces the optimal teaching set with high probability.

Theorem 1. *In the NTD setting, the proper teaching strategy is to collect* $m = H(\epsilon, \delta) = \frac{1}{2\epsilon^2} ln(\frac{2}{\delta})$ *samples of the Bernoulli distribution and then say* stop.

Proof. This strategy produces enough samples for learners that requires m samples before they make any predictions (e.g. KWIK learners [Li et al., 2011]). Suppose a learner existed that made inaccurate ($> \epsilon$ error) predictions with probability $> \delta$ after seeing these samples with empirical mean $\hat{p}$. Then by Definition 4 this learner would be inconsistent because Hoeffding's inequality states $Pr[|\hat{p} - p^*| > \epsilon] < \delta$. □

This is not surprising since this is also the sample complexity as an autonomous coin learner [Li et al., 2011]. However, in NSTD, the teacher has a significant impact. The following two theorems lay out (1) the reasoning that leads to the different behavior and (2) the sample complexity of coin learning in this protocol.

Theorem 2. *The general NSTD teaching policy for teaching the probability of a weighted coin is to flip the coin at most* $H(\epsilon, \delta)$ *times, but the teacher can* stop *building the collection whenever the empirical mean* $\hat{p}$ *of the coin's bias is within* $\epsilon/2$ *of the true probability.*

Proof. After processing a teaching collection of some size m, a learner can construct a confidence interval (using Hoeffding's inequality), denoting the upper and lower limits of p^* (the true probability of heads) with probability $1 - \delta$. We call this region the *consistent* region and the hypotheses outside of this region are deemed *inconsistent*. In addition, we can define the empirical mean $\hat{p} = \sum_{i=1}^{m} y_i/m$ and a region $[\hat{p} - \frac{\epsilon}{2}, \hat{p} + \frac{\epsilon}{2}]$ that we call the *instantaneous* region.

In NTD (NSTD^0) we just saw the teacher's only recourse is to provide $m = H(\epsilon, \delta)$ samples. For NSTD^1, where the learner is aware it is being taught, suppose the teacher presents a collection of size $m' < m$ to the learner. If p^* is in the inconsistent region, then the label set has probability $< \delta$, so the teacher was correct in stopping early. If p^* was in the consistent region but *not* in the instantaneous region, then the teacher would not have stopped construction, because treating this as a consistent subcollection will not guarantee the learner picks a distribution within ϵ of p^*. Therefore, in the case where the teacher stops the construction while the consistent region is still larger than the instantaneous region, the teacher's current collection is a consistent subcollection of a possible NTD collection and the learner can pick any hypothesis from the instantaneous region and be correct. □

We will now prove that the expected time for teaching a distribution in the NSTD framework is significantly smaller than the standard Hoeffding bound ($H(\epsilon, \delta)$) that determines the number of samples needed to learn in the NTD protocol. We believe the result is interesting in its own right as it describes a useful fact about the expected time the empirical average needs to first hit an interval of interest centered at the true expected value of a random variable.

The key technique we will use is formulating the evolution of the empirical average as a random walk and reducing the problem of hitting the desired interval to the problem of computing the mean first passage time through a fixed barrier. The models we introduced so far use Bernoulli as the standard example of learning a noisy concept. In the interest of technical brevity, we will actually prove the result for the case where the noise model is a Normal distribution. The reason is that the proof is more insightful for a continuous distribution and it is known in the literature [Redner, 2001; Feller, 1968] that the first passage time properties in the continuous case approximate the discrete version well.

Theorem 3. *Given a Normal distribution with unknown mean p, and the ability to sample from it, the expected number of samples a teacher needs to teach p in the NSTD protocol scales with* $O(\frac{1}{\epsilon})$ *(the variance is considered known).*

Proof. The above theorem statement is equivalent to stating that the expected first time the empirical av-

erage of a sequence of i.i.d. samples from a $D = Normal(p, 1)$ distribution hits the interval $[p-\epsilon, p+\epsilon]$ is $O(\frac{1}{\epsilon})$. Let $X_i, i = 1, 2, ...$ be i.i.d. samples from distribution D and let $S_t = \sum_{i=1..t} X_i$ be a random walk on $\mathbb{R}$ (with step values sampled from D). The empirical average at any time $t > 0$ is of course $\hat{X}_t = \frac{S_t}{t}$. Let $Y_i = X_i - p$ (thus $Y_i \sim Normal(0, 1)$) and let $W_t = \sum_{i=1..t} Y_i$. We can thus write $S_t = pt + W_t, \forall t > 0$.

Proving a bound on the expected time it takes $\hat{X}_t$ to first hit interval $[p - \epsilon, p + \epsilon]$ is equivalent to showing a bound on the expected time it takes the random walk S_t to hit the dynamic interval $[pt - \epsilon t, pt + \epsilon t]$. Let $A_t = pt + \epsilon t$ be a process that encodes the time evolution of the upper bound of the dynamic interval.

We will first focus on the expected time it takes for the random walk to first be inside the interval from the perspective of the upper bound. Let $B_t = A_t - S_t = \epsilon t - W_t = \epsilon t + W_t$ where the last equality follows from W_t being a symmetric stochastic process around 0. Since we are looking for the expected time t that A_t first becomes larger than S_t, this is equivalent to asking what is the expected first time that B_t hits the origin if it first starts on the negative side (if it starts on the positive side, this first time will be 1).

We will now approximate the discrete-time stochastic process B_t by a continuous-time process with the goal of getting a qualitative result about the expected mean time. This technique is commonly used to study first passage-time properties of discrete time random processes (see for instance the Integrate-and-Fire Neuron model in section 4.2 in [Redner, 2001]). Viewed from this perspective, $B_t = \epsilon t + W_t$ is actually the standard Brownian motion with positive drift ϵ. But it is well known that the mean first passage time over a fixed positive constant α for a Brownian motion with positive drift ϵ is governed by the **Inverse Gaussian Distribution (IG)** with parameters $IG(\frac{\alpha}{\epsilon}, \alpha^2)$ [Chhikara and Folks, 1989]. The expected value of the IG distribution is $\frac{\alpha}{\epsilon}$ and if we take $\alpha = 1$ we get that the expected first time that B_t will become larger that 1 is $\frac{1}{\epsilon}$. Since the expected time to first hit the origin if started on the negative side is naturally upper bounded by the expected time to hit 1, we get a bound on the expected time A_t needs to first 'catch up' with the random walk S_t.

Symmetrically we can show that the expected time for S_t to become larger than the lower bound of the dynamic interval (the process $pt - \epsilon r$) is also $\frac{1}{\epsilon}$. □

Figure 1a shows empirical validation of this result in the coin-flipping case for increasingly smaller values of ϵ (1000 runs, $\delta = .05$). While NTD grows quadratically in accordance with $H(\epsilon, \delta)$, the NSTD expected time grows only linearly in $\frac{1}{\epsilon}$.

4.3 k-armed Bandits

A natural extension of coin-learning is teaching a complete model in the k-armed bandit case [Fong, 1995], that is teaching the expected payout of all k arms, each of which may have a different noisy (but bounded $[0, 1]$) payout function. We note again that we are teaching the full model here, not the optimal policy. Each arm can be treated as a Bernoulli distribution that needs to be learned within ϵ with probability δ/k to ensure total failure probability at most δ . Note ϵ does not change with k because each arm corresponds to a different input parameter x (a different action). Hence for k arms, the NTD solution is to pull each arm $H(\epsilon, \delta/k)$ times, giving us an NTD bound of $\frac{k}{2\epsilon^2} ln(\frac{2k}{\delta})$. For NSTD, the teacher can again pull the arms in some ordering, but can stop pulling an arm when it has either (1) been pulled $m = H(\epsilon, \delta/k)$ times or (2) its empirical average is within $\epsilon/2$ of its true payout. The expected savings over NTD is a factor of $\frac{1}{\epsilon}$ *for each arm*, so the speedup effect is actually multiplied across the arms. Figure 1b illustrates this in the bandit setting for increasing k (1000 runs, $\epsilon = 1/45$, $\delta = .05$). The algorithms just described are labeled NTD-IND and NSTD-IND (for "individual" pulls). The growth of NTD-IND is actually $kln(k)$ but quickly diverges from the other approaches here, showing the increasingly better (than NTD-IND) expected performance for NSTD-IND.

While not applicable in RL, the complexity of teaching by pulling all of the arms in parallel will be informative in the next section, so we investigate it here. The goal is still to learn each arm's expected payout with ϵ-accuracy, but the teacher has access to an action that pulls all of the arms at once and reports their individual payouts. NTD in this case (NTD-PAR in Figure 1b) performs $H(\epsilon, \delta/k)$ parallel pulls, saving a factor of k over the individual NTD above. But in NSTD, the parallel pulls introduce a tension between the speedup from parallelizing and the previously noted speedup from being able to stop pulling an arm when its empirical mean is close to the true mean. Now if some arms are close to their empirical mean but others are not, a parallel pull may disturb the empirical means of the "learned" arms. Figure 1b shows the NSTD-PAR strategy that is forced to either perform a parallel pull or *stop*, which it only does when *all* the empirical payouts are within $\epsilon/2$ of their true payout or have been pulled $H(\epsilon, \delta/k)$ times. For small k, the sequential pulls are actually more efficient, but for a larger number of arms, the parallel pulls have a significant benefit despite the danger of "unlearning" an arm.

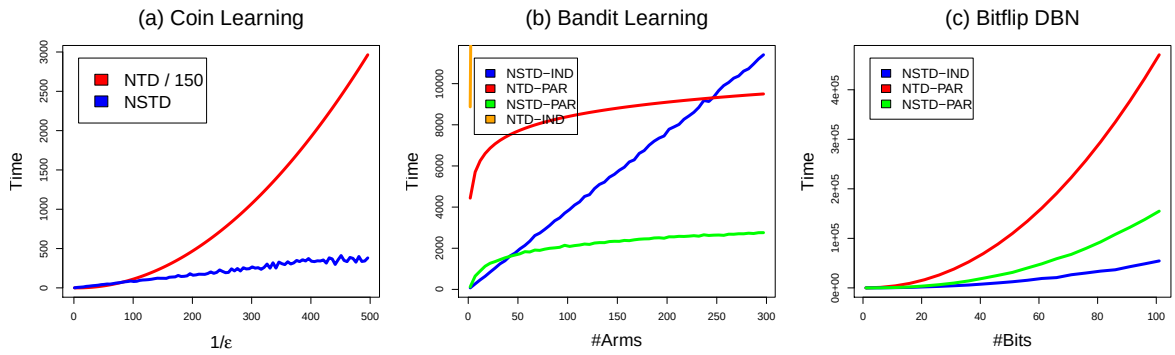


Figure 1: TD and STD comparisons for (a) coin flipping, (b) k-armed bandits and (c) the Bitflip DBN

4.4 Dynamic Bayesian Networks

We now consider Dynamic Bayesian Networks [Dean and Kanazawa, 1989], or DBNs, where multiple noisy factors are learned in parallel. This case is similar to the parallel bandit case but here because the total error (ϵ) is an aggregate of the subproblem errors, the NSTD strategy will be dramatically different. A Dynamic Bayesian Network is composed of n discrete valued factors. At every timestep, the next value for factor x_i is determined by a probability distribution $P(x_i|\rho(x_i))$ where ρ maps the factor to its k "parent factors". We assume here that $k = O(1)$ and we only consider the binary case (all factors are either 0 or 1) as the bounds generalize to ordinal values from 1 to D with extra terms including D^{k+1} (see [Li et al., 2011]).

Mapping this back to our learning protocols, the input space is all possible configurations of the DBN factors and predictions should be made on the distribution of the factors at the next timestep. As a simple example, consider the case where every variable is the parent of only one other variable. Assume that the structure is known and the DBN is *deterministic*, so all that has to be learned is the relationship between each parent value and its child value. In the traditional mistake bound setting, where the teacher has no control over the inputs, the worst case learning bound is $O(n)$, because each example after the first one could show the same parent (input) pattern as the first, except with a single bit flipped. Because of the connection between mistake bound learnability and teaching by demonstration [Walsh et al., 2010] this is the worst-case bound for teaching this type of DBN by demonstration of a static policy. However, with our dynamic teaching protocols we can do much better. Specifically, in the TD case the sample complexity is $O(1)$, because the teacher can pick the parent values. In the first example, the teacher can pick an arbitrary setting of the factors and show the result. Then all it needs to do is show the complement of this bit string in the second example. Thus, deterministic binary DBNs with $k = 1$ are teachable with 2 examples. The STD bound is the same in this case and the generalization to multiple parents adds only a constant term.

In the stochastic setting, the consequences of teaching the factor probabilities in parallel are more complicated than the deterministic case. Unlike the bandit case, the total error in predicting the next state probability is based on the aggregate error (total variation by Definition 4) of all the subproblems. So each factor needs to be predicted with ϵ/n accuracy. This means that for an NTD teacher, we need $H(\epsilon/n, \delta/n^k) = O(\frac{n^2}{\epsilon^2} kln(\frac{n}{\delta}))$ samples to learn the probabilities.

The stochastic setting is even more complicated in the NSTD case, where once again there is competition between the desire to teach factor probabilities in parallel and the desire to stop showing certain conditions once they have been taught. The parallel and individual teaching styles in the bandit case manifest themselves here in the following ways. NSTD-PAR tries to teach multiple factors at once, using the same "complement" input selection as NTD but can stop providing samples whenever *all* of the factors are $\epsilon/2n$ accurate. The worst case time for this approach is bounded by NTD's bound above but again on expectation this will usually perform better than NTD. However, if one wishes to minimize expected teaching time when there is background knowledge that says certain conditions are already known, a variant of the NSTD-IND strategy is likely better. In NSTD-IND, the teacher presents inputs with only one unknown condition for a single factor, and all other factors have their parent values set to configurations matching the background knowledge. As in the bandit case, this avoids "unteaching" but sacrifices parallelization. Once the factor outcome being taught has its empirical distribution within $\epsilon/2n$ of the true probability, the algorithm moves on to the next condition. The worst-case time for NSTD-IND is $O(\frac{n^3}{\epsilon^2} kln(\frac{n}{\delta}))$, worse than TD and NSTD-PAR, but

its expected time is only $O(\frac{n^2}{\epsilon} kln(\frac{n}{\delta}))$ since it targets individual conditions.

An empirical demonstration of these teaching strategies is shown in Figure 1c for a DBN representation of a noisy Bitflip domain [Givan et al., 2003]. The input for this DBN is simply a bit string of length n and there are two actions, one to flip bit 0 and one to *shift* each bit up (with a 0 always incoming to bit 0). However, the effect of the shift is noisy for each bit: with probability p_i the shift to bit i is successful, and otherwise the bit retains the same value it currently has. Figure 1c shows the number of steps needed to teach all of the p_i's for $\epsilon = .3$ (aggregate error) and $\delta = .05$ for the three teaching protocols described above (500 runs). NTD teaches by setting the inputs (current state) to an alternating string of 0's and 1's ending in a one, so that each bit's shift probability is observed on every step (ensuring parallel teaching). NSTD-PAR uses the same alternating bit string for every example, but stops building the collection if all of the p_i values are within $\epsilon/2n$ of their true value. NSTD-IND sets up the state to have all 1's from bit 0 up to the highest bit it has not yet taught, and then all $0's$ above that. This teaches the probability of shifting each bit one at a time with deterministic outcomes for the other bits. We see here that unlike the bandit case, NSTD-IND dominates the other strategies, even as the number of bits increases. This is because the continually shrinking accuracy requirements increase the chances of the parallel strategy unteaching a factor. However, NSTD-PAR still outperforms NTD, so in situations where no background knowledge is available (for NSTD-IND) and the teacher and learner are aware of each other, this may be the preferred strategy.

Finally, we note Bitflip also showcases the benefit of teaching versus demonstration from an optimal policy. Consider Bitflip with a reward only when all bits are 1. The optimal policy is to flip if bit 0 is a 0, otherwise shift. However, this strategy produces very few useful samples because once a bit is turned to 1 in an episode, it will never be a 0 (even if the shift fails the bit retains its current value). Even if $p_i = .5$, the expected number of useful samples in an episode for bit i is only 2, and the probability of more samples drops off exponentially. So the optimal *performance* policy will almost always take far more steps to teach the domain than any of the teaching protocols described above.

5 Teaching in an MDP

Above, we established the teachability of several RL concept classes in a supervised setting. However, we are interested in teachers acting out lessons in an MDP, where a transition function $T : X, A \mapsto Pr[X]$ governs agent movement, and therefore the teacher may not have access to all possible states at every timestep. We now extend the previously defined frameworks to the sequential setting, forcing them to handle teaching *sequences* drawn from an MDP. We then describe the teaching process in an MDP as a planning problem in its own right and present and demonstrate a heuristic solution based on our supervised learning results.

5.1 Sequential Teaching Frameworks

In sequential domains, the teacher may not be able to access all possible states at every timestep, but instead may have to take multiple steps (each of which will be seen by the learner) to reach its target teaching state. Hence, we now adapt the TD and STD definitions to consider *sequences* of states and actions rather than randomly accessed inputs and labels. More formally, we need each protocol to handle the following constrained teaching sequences.

Definition 7. *An* MDP teaching sequence $MTS(c, C, M)$ *for MDP* $M = \langle X, A, T, R, \gamma \rangle$ *and* $c : X, A \mapsto Pr[X]$ *or* $c : X, A \mapsto \Re$ *is a sequence* $\langle \langle x_0 a_0 r_0 \rangle ... \langle x_T, a_T, r_T \rangle \rangle$ *with each* $\langle x_i, a_i, r_i, x_{i+1} \rangle$ *reachable by taking action* a_i *from* x_i *in* M*, starting from* $x_0 = s_0$*. After witnessing this sequence, the* $\hat{c}$ *learned by a consistent learner must either be* c *(deterministic) or distributionally consistent (Definition 4) to* c *with high probability (stochastic concepts).*

In the definition above, c is a concept that may be part of the transition or reward function (or both) of M. For TD, instead of considering a set of instances and labels, one now needs to consider a sequence of states and actions drawn from M based on the teacher's potentially non-stationary policy π. We denote the distribution of such sequences as $MTS^{\pi}(c, C, M)$. Formally, we define Sequential Teaching Dimension (TD_S) in an MDP as:

Definition 8. *The* sequential teaching dimension $TD_S(C, M)$ *of concept class* C *being taught in MDP* M *with accuracy parameters* ϵ *and* δ *is the maximum of the minimum expected length of an MDP teaching sequence for any individual concept in the class,* $TD_S(C, M) = \max_c \min_{\pi} E[MTS^{\pi}(c, C, M)]$*, where* π *is an arbitrary, potentially non-stationary, policy (any possible teaching algorithm).*

This definition is very general and using it constructively in the full stochastic case is a topic of future work, but there are cases where the optimal dynamic teaching policy is clear. For instance, when the transition function is deterministic (including cases where we are teaching a noisy reward function), the teacher simply needs to find the shortest-length teaching sequence starting from s_0 that contains enough samples

to teach c to any consistent learner. Also, heuristic solutions for approximating optimal teaching defined in this manner (e.g. Algorithm 1) can successfully teach in both the deterministic and stochastic setting.

The conversion of Zille's STD protocol to the sequential setting is even more complicated because in STD the learner makes inferences about why the teacher is showing certain instances, but now not every instance is accessible at every step. The main intuition we can use to resolve these difficulties in the case where T is deterministic is to instead apply the recursive reasoning to the teaching sequences defined above. Intuitively, this should allow the learner to reason that a teacher is trying to teach a concept c when the teacher chooses a path that leads to a state consistent with c instead of other paths that would teach different concepts. This saves the teacher from walking the entire path. We can formalize this using the notion of an example subsequence and prefix.

Definition 9. *An* example subsequence Σ' *of example sequence* Σ *is a contiguous series of inputs and labels* $\langle x_i, y_i ... x_j ... y_j \rangle$ *for* $i \geq 0$, $j \leq T$. *A* prefix *(*$\Sigma' <_{pre} \Sigma$*) is a subsequence with* $i = 0$ *and a* suffix *is a subsequence with* $j = T$.

Using this definition, we can replace previous definitions of subsets and collections in STD and NSTD and define a SubSequence Teaching Dimension (SSTD) in a natural way for MDPs with a deterministic transition function (see the appendix).

As a concrete example of the more powerful reasoning about sequences in SSTD and NSSTD (its noisy version), consider coin learning, but instead of predicting a probability, assume the learner only needs to predict whether the coin is biased more towards heads or tails (assuming $p^* \neq 0.5$). Without loss of generality assume the coin is biased towards tails but comes up heads on the first trial. In NSTD, because the ordering of examples is hidden to the learner (Def. 3), we need to keep flipping the coin until we have observed more tails than heads. But in NSSTD we only need one more flip and then the teacher can end teaching, no matter what the outcome of the second flip (and even though the empirical distribution will not indicate a tails bias). This is because if the coin was biased heads the teacher would not have made the second flip; it would have stopped after the first heads observation. Similar reasoning can be done when predicting p^* as every choice to flip indicates p^* is *not* in the previous step's instantaneous region (see Theorem 2).

5.2 Practical Teaching in MDPs

If teaching is to be done in an MDP, the best policy will directly optimize the TD_S or $SSTD$ criteria described above. However, even for TD_S in the deterministic setting, the problem of determining the exact optimal teaching sequence may be intractable.

Theorem 4. *Determining the optimal teaching sequence for a concept in a deterministic MDP under the conditions of Definition 8 is NP-Hard.*

Proof. We can encode the graph of a Traveling Salesman Problem (TSP) with integer edge costs as a deterministic MDP with unit costs by adding in the same number of "dummy" states between cities as their distance in the original graph. Then consider teaching a reward function that is known to be 0 at every non-city state but has an unknown value at each of the cities. The shortest tour for teaching such a reward function provides a solution to the original TSP. □

This is in line with previous results on the intractability of determining teaching sets in the supervised setting [Servedio, 2001]. However, tractable approximations such as Algorithm 1 are certainly possible. There we perform two approximations: first we use a greedy approximation of the supervised (random access) teaching collection construction to find a set of instances that will teach the target concept and then greedily construct a tour of all instances in this set.

Algorithm 1 Heuristic teaching in an MDP

input: Concept $c \in C$ to be taught, MDP M with start state s_0, Learning protocol P
$TS(c, C) = \emptyset$
$R = \langle s, a, r, s' \rangle \in M$ reachable from s_0
while $TS(c, C)$ is not a teaching set for c in P **do**
  $X = \langle s, a, r, s' \rangle \in R$ and $\notin TS(c, C)$ that teaches the most parameters of c
  $TS(c, C) = \{X\} \cup TS(c, C)$
while $TS(c, C) \neq \emptyset$ **do**
  $X' =$ closest $X \in TS(c, C)$ to current state, reachable fastest with policy π
  Demonstrate π until X' is demonstrated
  $TS(c, C) = TS(c, C) \backslash X'$

In the SSTD setting, the first approximation by itself could be problematic because the inference process is based on optimality assumptions shared by the learner and teacher. If the teacher uses a suboptimal teaching sequence, then the learner can make the wrong inference. So for extending SSTD or NSSTD to the practical setting, any approximations used by the teacher to construct the teaching set or the touring policy need to be shared between the teacher and the learner, allowing them to simulate each other's behavior. The motivation behind this sharing is to engender natural teacher-student interactions by having the learner and teacher share a set of common assumptions about

one another (such as optimality or consistency in the original STD and TD).

Finally, we consider noise in the transition function for states and actions that are not part of the target concept being taught. In that case, we would ideally like to create a tour of instances with the shortest stochastic path length (in expectation), but this is certainly as hard as the deterministic touring case. However, we can modify the heuristic algorithm used above to simply find the state with the shortest stochastic path and then focus on reaching that state. This allows the heuristic to teach concepts in arbitrary MDPs.

5.3 An Example in the Taxi Domain

To demonstrate our heuristic teaching algorithm we conducted experiments with the Object-Oriented MDP encoding of a Taxi Domain [Diuk et al., 2008]. This environment consists of a square grid (5 by 5 for us) and a controllable taxi with actions {up, down, left, right}. There are also several *landmarks* in the grid, one of which initially contains a *passenger* and another of which is the passenger's *destination*. The taxi's goal is to go to the passenger, execute its *pickup* action, transport her to the destination, and then execute *dropoff*. In the OOMDP encoding of this domain, every state has a set of predicates (e.g. *Wall-ToRight*(taxi)) associated with it, and actions have parameters stipulating which objects are in its scope(e.g. *pickup*(T,P,L) for taxi, passenger and landmark). The predicates in this domain include indicators of walls and clearness in every direction from an object, as well as predicates indicating when two objects are in the same square and when the passenger is in the taxi. Also, each action has a conjunction over the variablized predicates that serves as a *pre-condition* for this action. We focus on teaching these pre-conditions.

We perform our experiments in a deterministic Taxi domain, starting with a modified TD-style conjunction teacher as the set constructor for Algorithm 1. The modification to the conjunction learner from Section 4.1 is that instead of using the most specific positive example, it may have to use one that contains irrelevant predicates (like *WallNorth* for *putDown*) because of state-space restrictions. The teacher must show more positive examples to discredit these irrelevant predicates in addition to the negative examples (action failures) to indicate the relevant variables.

To demonstrate the benefits of STD, we used an approximation of the STD behavior described in Section 4.1 that shows the most specific state it can, then shows positive examples for all of the irrelevant variables and then stops demonstrating (since the learner can infer all other variables are relevant). Table 5.3 shows the number of steps needed to fully teach the pre-conditions of several sets of actions with all others being known. The agent in all cases starts from the middle of the grid and there are 2 landmarks in the bottom-left and top-right corners.

Table 1: Teaching steps for selected action sets in taxi

Action(s)	TD	STD Approximation
pickup	20	15
pickup / putdown	23	17
movement	37	27
all	63	45

Several interesting behaviors were observed, including the teacher using the landmarks in the corners as areas to gather many positive examples (because the irrelevant predicates can be dispelled en masse there). The teacher also made use of the lack of type constraints on most of the variables to create negative instances (where the pre-conditions failed) by swapping the order of arguments to indicate specific relations were relevant. These results compare favorably to previous result on teaching taxi conditions by static-policy demonstration [Walsh et al., 2010] but static policy approaches can be made to look arbitrarily bad in this situation because they will not teach subtle aspects of the domain (like how the corners work) unless they are actually on the way to the goal. By contrast, our approach ignores the current goal and focuses on being the best teacher, not the best performer.

We also experimented with a sequential Bitflip domain (Section 4.4) where all but two of the bits (the middle and one from the end) had deterministic shift outcomes. Our experiments again showed the approximations using NSTD significantly outperforming those using NTD. The average steps to teach the 10 bit version were 362.35 (NSTD-Par), 869.61 (NSTD-IND) and 14610.41 (NTD-PAR). The parallel versions were all more efficient despite flipping every other action.

6 Conclusions

We have extended two supervised learning frameworks (TD and STD) to handle noisy observations and sequential data. These frameworks provide efficient ways to teach several RL classes, including DBNs and Bernoulli distributions. We also presented a practical heuristic algorithm for leveraging TD and STD to perform efficient teaching in an MDP and demonstrated its effectiveness in the taxi domain. Unlike previous efforts at teaching domains with a static policy, these new algorithms actually target the individual parameters of a domain in a task-independent manner, leading to agents that truly teach, rather than just show.

References

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *ICML*.

Abbeel, P. and Ng, A. Y. (2005). Exploration and apprenticeship learning in reinforcement learning. In *ICML*.

Argall, B., Chernova, S., Veloso, M. M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483.

Bain, M. and Sammut, C. (1995). A framework for behavioural cloning. In *Machine Intelligence*.

Balbach, F. J. and Zeugmann, T. (2009). Recent developments in algorithmic teaching. In *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications*, LATA '09. Springer-Verlag.

Chhikara, R. S. and Folks, J. L. (1989). *The inverse gaussian distribution: theory, methodology, and applications*. Marcel Dekker, Inc., New York, NY, USA.

Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational intelligence*, 5:142–150.

Diuk, C., Cohen, A., and Littman, M. L. (2008). An object-oriented representation for efficient reinforcement learning. In *ICML*.

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley.

Fong, P. W. L. (1995). A quantitative study of hypothesis selection. In *ICML*.

Givan, R., Dean, T., and Greig, M. (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147((1-2)):163–223.

Goldman, S. A. and Kearns, M. J. (1992). On the complexity of teaching. *Journal of Computer and System Sciences*, 50:303–314.

Khan, F., Zhu, X., and Mutlu, B. (2011). How do humans teach: On curriculum learning and teaching dimension. In *NIPS*.

Khardon, R. (1999). Learning to take actions. *Machine Learning*, 35(1):57–90.

Knox, W. B. and Stone, P. (2010). Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *AAMAS*.

Li, L., Littman, M. L., Walsh, T. J., and Strehl, A. L. (2011). Knows what it knows: A framework for self-aware learning. *Machine Learning*, 82(3):399–443.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound. *Machine Learning*, 2:285–318.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York.

Redner, S. (2001). *A guide to first-passage processes*. Cambridge University Press, Cambridge.

Servedio, R. A. (2001). On the limits of efficient teachability. *Information Processing Letters*, 79.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Walsh, T. J., Subramanian, K., Littman, M. L., and Diuk, C. (2010). Generalizing apprenticeship learning across hypothesis classes. In *ICML*.

Zilles, S., Lange, S., Holte, R., and Zinkevich, M. (2011). Models of cooperative teaching and learning. *Journal of Machine Learning Research*, 12.

Fast Graph Construction Using Auction Algorithm

Jun Wang and Yinglong Xia
IBM T. J. Watson Research Center
1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA
{wangjun, yxia}@us.ibm.com

Abstract

In practical machine learning systems, graph based data representation has been widely used in various learning paradigms, ranging from unsupervised clustering to supervised classification. Besides those applications with natural graph or network structure data, such as social network analysis and relational learning, many other applications often involve a critical step in converting data vectors to an adjacency graph. In particular, a sparse subgraph extracted from the original graph is often required due to both theoretic and practical needs. Previous study clearly shows that the performance of different learning algorithms, e.g., clustering and classification, benefits from such sparse subgraphs with balanced node connectivity. However, the existing graph construction methods are either computationally expensive or with unsatisfactory performance. In this paper, we utilize a scalable method called auction algorithm and its parallel extension to recover a sparse yet nearly balanced subgraph with significantly reduced computational cost. Empirical study and comparison with the state-of-art approaches clearly demonstrate the superiority of the proposed method in both efficiency and accuracy.

1 INTRODUCTION

Graph based data representation treats data points as graph nodes and builds pairwise edges between these nodes which are often weighted by the similarity or correlations between the corresponding data samples. In the past decade, graph based data representation has become very popular and widely used in many machine learning algorithms due to both practical needs and theoretical advantage. First, under many practical situations, the data come in a natural way of graph representation, where the links or connections between items can be treated as directed or undirected edges. Typical examples include social networks, the Internet, and citations networks [13, 21]. Second, for the conventional vector space based data representation, a data graph can be easily generated using pre-defined similarity or correlation measurement.

Given the popularity of data graph representation, many graph based machine learning algorithms have been developed, including graph based clustering and semi-supervised learning algorithms. For instance, representative graph based clustering approaches include those formulated as a graph cut problem [16, 20]. In addition, graph has also been integrated into semi-supervised learning paradigm, resulting in a wide range of approaches, such as graph Laplacian regularization framework [2, 23, 25] and random walk based methods [1, 22]. Other popular methods include graph based ranking algorithms [17, 24]. In many real applications and systems, graph based approaches have shown empirical success, especially for those with agnostic setting and given no prior knowledge of the data distributions.

As indicated in [12, 15], constructing a suitable data graph is the vital step for ensuring the learning performance. The sparse structure of the constructed graph is desired for learning algorithms to remain efficiency to memory usage and robustness to noisy samples. Given an adjacency graph, a typical way to recover a sparse graph without losing the global structure of the data is to build a neighborhood graph. The objective is to connect each data point with its nearest neighbors and prune all other connections. However, since the nearest neighborhood method prune the edges through a greedy process, it can only produce an asymmetric graph with some adhoc symmetrization processes. As a result, the constructed neighborhood graph is often irregular and imbalanced, where

the learning performance could be significantly degraded [12]. Therefore, this drawback of nearest neighbor method motivated one to consider matching technique, such as b-matching, to build a balanced graph structure [11, 12]. Instead of achieving symmetrization by post-processing, b-matching sets the symmetry as additional constraints and directly extracts a symmetric solution. But this modification imposes much high computation and memory cost and the one of the known fast approximated solution using belief propagation has the space complexity of $O(n)$ and the time complexity $O(n^{2.5})$ [10], where n is the number of graph nodes (refer to Table 1).

Considering the trade off between computational efficiency and performance, in this paper, we propose a fast and scalable solution for constructing nearly balanced graphs using the auction algorithm proposed in [3]. In order to remain performance superiority and reduce computation cost, our method tends to extract an approximately balanced graph though preserving the regularity as much as possible. We show that this method has the space complexity $O(n)$ and time complexity $O(|\mathbf{E}| \cdot \max |a_{ij}|/\epsilon)$, where $\max |a_{ij}|/\epsilon$ is a constant determined by the data and the learning parameter ϵ. It is much less expensive than the existing regular graph construction methods, e.g. b-matching technique. In addition, the auction algorithm based graph construction is naturally parallelizable and therein the computational efficiency can be further improved using multiple processes. We provide extensive experiments and comparison study with the state-of-the-arts on both synthetic and real benchmark datasets. The experimental results clearly show that the graphs created by parallel auction algorithm (PAA) provide superior performance than neighborhood graphs for different machine learning algorithms. Although PAA graphs have slightly lower quality than b-matching graphs, we observe that PAA method reduces the computational cost by hundreds of times than b-matching method. Therefore, the PAA method is very suitable for handling large scale applications with massive data graphs.

The remainder of this paper is organized as follows. In Section 2, we give a brief introduction of related work on graph construction process. A new graph construction and sparsification method using a parallel auction algorithm is presented in Section 3. The experiments are reported in Section 4 followed by conclusions and discussions in Section 5.

2 RELATED WORKS

Although graph based machine learning methods have been extensively studied and explored, there have been limited efforts for building effective data graphs in the community. Although some recent methods use sampling techniques to approximate large scale graphs [14], it still remains as a challenging issue to efficiently construct large scale graphs with good qualities, such as sparsity and regularity. In this section, we will provide a brief introduction about related work for graph construction and then motivate our proposed method.

Given a data set $\mathbf{X}$, graph construction aims in converting the data to a weighted graph $G = \{\mathbf{X}, \mathbf{E}, \mathbf{W}\}$ where graph nodes are the samples and $\mathbf{W} \in \mathbb{R}^{n \times n}$ is a weight matrix of edges $\mathbf{E}$. As summarized in [12], two important steps for graph constructions are sparsifying the adjacency matrix $\mathbf{A} = \{a_{ij}\} \in \mathbb{R}^{n \times n}$ and assigning weights to edges. In many practical settings, the adjacency matrix is computed using a similarity function for all pairs of nodes, i.e. $a_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j)$. For other applications, such as social network analysis and relational learning, the adjacency or connection matrix $\mathbf{A}$ might be given in a natural way without the above process of computing pair-wise similarity. In the second step, graph sparsification tries to recover a sparse binary matrix $\mathbf{B} = \{b_{ij}\} \in \mathbb{B}^{n \times n}$ to preserve the global structure of data using local information. The final edge matrix can be easily derived as $\mathbf{W} = \mathbf{B} \bullet \mathbf{A}$, where the symbol $\bullet$ indicates element-wise multiplication. Such an extract sparse subgraph often leads better learning performance and requires less storage and computation cost [11, 12].

The most popular method for constructing a sparse graph is the nearest neighbor (NN) approach, including different variants such as k-nearest neighbor and ϵ-nearest neighbor methods. Since ϵ-nearest neighbor often generate fragile subgraphs and does not provide satisfactory performance, kNN graph remains more popular in many applications [15]. Briefly speaking, kNN graph construction performs a greedy search process to select the edges with the most significant weights for each node, and remove all the other insignificant ones. The objective of kNN approach indeed solves the following maximization problem

$$\begin{aligned} &\max_{b_{ij}} \sum_i \sum_j b_{ij} a_{ij} \\ \text{s.t.} \quad &\sum_j b_{ij} = k, \\ &b_{ii} = 0, b_{ij} \in \{0,1\}, \forall i, j = 1, \cdots, n \quad . \end{aligned} \tag{1}$$

Since the greedy search for k nearest neighbors does not guarantee to produce symmetric $\mathbf{B}$, it usually needs some postprocessing to achieve symmetrization by $\max(\mathbf{B}, \mathbf{B}^\top)$ or $\min(\mathbf{B}, \mathbf{B}^\top)$. However, this adhoc symmetrization often leads to imbalanced connectivity for nodes, which could seriously degrade the learning

Table 1: Comparison of the time and space complexity for different graph construction approaches. Assume the input adjacency graph has n nodes and $|\mathbf{E}|$ edges. The compared methods are: Loopy Belief Propagation (LBP) [9], Sufficient Selection Belief Propagation (SSBP) [10], and the auction algorithm.

Method	Time	Space
LBP Method	$O(n\|\mathbf{E}\|)$	$O(n+\|\mathbf{E}\|)$
SSBP Method	$O(n^{2.5})$	$O(n)$
Nearest Neighbor Method	$O(n^2)$	$O(1)$
Auction Method	$O(\|\mathbf{E}\| \cdot \max\|a_{ij}\|/\epsilon)$	$O(n)$

performance, as observed in [12].

As a valuable alternative for extracting sparse graph structure, weighted b-matching technique [18] has been applied to build graph with balanced node connectivity. Similarly to the objective of kNN approach, b-matching recovers a sparse graph with the maximized edge weights. Furthermore, b-matching enforces the symmetry of connectivity through directly imposing a set of constraints $b_{ij} = b_{ji}$. Essentially, b-matching is a generalized version of the well-known linear assignment problem, which can be solved by Hungarian method with a cubic complexity $O(n^3)$ ($n = |\mathbf{X}|$) in time. Various methods have been proposed to solve b-matching problem with less memory and time cost, including the reduction based method [8] and belief propagation based approaches [9, 10], for which the computational complexity is summarized in Table 1. Although b-matching method yields clear advantage for different machine learning algorithms, including spectral clustering and semi-supervised classification [11, 12], it is a much more expensive procedure than the intuitive kNN method. Due to both time and memory constraints, the existing b-matching based graph construction is infeasible for large scale datasets.

3 AUCTION ALGORITHM FOR GRAPH CONSTRUCTION

In this section, we describe the procedure to utilize an auction algorithm to identify bn significant edges from the given bipartite graph corresponding to the adjacency matrix $\mathbf{A}$. In particular, each node is connected with at most b edges. If given an arbitrary graph, it is straightforward to convert it to a bipartite graph though creating a shadow node for each graph node and only connect the original graph nodes to the shadow nodes. In order to utilize distributed computing systems to process massive data graphs, we also describe the parallelization technique for the proposed graph construction method.

3.1 Auction Algorithm for Linear Assignment

Here, we first describe the solution for a single-edge matching problem using an auction algorithm, and then provide general extensions. For a standard maximum 1-matching problem (also known as a linear assignment problem) over a bipartite graph with a non-negative weight matrix $\mathbf{A}$, the objective can be written as:

$$\max_{\mathbf{B}} \ \mathrm{tr}(\mathbf{A}^\top \mathbf{B}) \quad \text{s.t.} \ \ \mathbf{B}\vec{1}^T = \vec{1}, \ \mathbf{B}^T\vec{1} = \vec{1}, \ \mathbf{B} \in \mathbb{B}^{n\times n}, \tag{2}$$

where $\vec{1} = (1, 1, \cdots, 1)^T$. This linear program can be interpreted as selecting a permutation matrix $\mathbf{B}$, so that after permutation of $\mathbf{A}$, the sum of entries on the diagonal is maximized. The edges corresponding to the permutation matrix $\mathbf{B}$ is as known as a matching in $\mathbf{A}$. Thus, the solution of this program can be used to sparsify $\mathbf{A}$ with constraints of regularity. To generalize the standard linear assignment problem to the objective of extracting a balanced graph where each node connects to b edges, we can simply replace $\vec{1}$ with $\vec{b} = b \cdot \vec{1}$. However, this solution for such a b-matching problem can be approximated by solving the above linear assignment problem for $b/2$ times and removing the selected edges after each iteration.

As discussed earlier, we seek fast and scalable solutions for graph sparsification due to the demand of processing large scale graphs in many domains. Conventional approach for solving Eq. 2 is based on increasing augmenting paths, which leads to low concurrency and limited scalability [3, 19]. Thus, we propose to use the auction algorithm introduced by Bertsekas in [3]. Since the constrained problem in Eq. 2 can be converted into a dual form, the auction algorithm deals with the above linear programming by solving its dual problem. In particular, we are solving the following dual problem of the original primal linear program:

$$\min_{p,\pi} \ \vec{1}^T p + \vec{1}^T \pi \quad \text{s.t.} \ \ \vec{1}p^T + \pi\vec{1}^T \geq \mathbf{A}, \tag{3}$$

where $p = (p_1, p_2, \cdots, p_n)^T$ and $\pi = (\pi_1, \pi_2, \cdots, \pi_n)^T$ are known as the dual vectors. The constraints $\vec{1}p^T + \pi\vec{1}^T \geq \mathbf{A}$ indicate the element wise relationship between the matrices $\vec{1}p^T + \pi\vec{1}^T$ and $\mathbf{A}$, i.e. $(\vec{1}p^T + \pi\vec{1}^T)_{ij} \geq a_{ij}, i, j = 1, \cdots, n$. The auction algorithm computes the dual variables through an iterative *auction* process. Intuitively, we view each row i as a buyer and each column j as an object; a_{ij} is the benefit of object j to buyer i; p_j is the price of object j (initialized to 0 at the beginning); π_i is the profit for buyer i, defined as $(a_{ij} - p_j)$, in case object j is assigned to i. The auction algorithm proceeds as follows: Each buyer i bids for an adjacent object j that offers the highest profit, i.e., $\max_{j \in \text{adj}(i)}(a_{ij} - p_j)$, where $\text{adj}(i)$ denotes the set of adjacent nodes of i. The bid is the difference between the highest profit and the second highest profit. An object j is assigned to the buyer offering the highest bid, say buyer i. Thus, row i and column j are matched and price p_j is increased by the bid. Price increment makes an object expensive for competitors, so that they can choose other objects. The above steps are repeated for all unmatched rows until every row is matched or the matching does not change.

The naive auction algorithm described above has a defect caused by *price war* [3]. When the highest and second highest profits are equal for some buyers, the bid becomes 0 and the object price remains unchanged. Therefore, buyers can indefinitely compete for the same object without increasing its price. The object will be alternately assigned to them and the auction will not progress. To overcome this issue, a small positive scalar ϵ is added to the price of an object once it is assigned to a buyer. A smaller ϵ leads to better accuracy; a greater ϵ results in an accelerated auction process.

3.2 Auction for Graph Sparsification

The auction algorithm described in Section 3.1 selects a single edge for each node while maximizing the total weight of the selected edges. For selecting b edges, there are two straightforward approaches to generalize the auction. The first approach is to perform auction multiple times. Every time the algorithm selects an edge for each node, if exists, without having two edges connect to the same node. The other approach is to select up to b edges in each iteration process, following the same criteria for edge selection. The details are discussed in below.

Note that the above algorithm does not ensure that if a node i connects to a shadow node j', node j will also connect to node i'. This can cause adverse impact on graph regularity. We propose a heuristic strategy to reduce such negative impact. When such inconsistence occurs, saying node j selects node $\tilde{i}'$ instead of node i', we compare the percentile of edge $(j, \tilde{i}')$ with respect to all incident edges of j and the percentile of (i, j') with respect to all incident edges of i. By percentile, we mean the index of the selected edge among all incident edges sorted in descending order. We accept the edge with smaller percentile while reject the other one. The intuition that we select edges according to the percentile instead of edge weight is that all incident edges for the boundary nodes in a graph can be weighed much higher than even the lightest incident edge of an internal node.

Algorithm 1 Fast Graph Construction Using Parallel Auction Algorithm

Graph partitioning: partition graph G into L disjoint subgraphs $\{G_l\}_{l=1}^L$ with weight matrices $\{\mathbf{A}_l\}_{l=1}^L$, where $\mathbf{A}_l = \{a_{ij}^l\}$

Initialization: Dual vectors $p = \vec{0}$, initial $\pi = \vec{0}$, the set of selected edges $\mathcal{M} = \emptyset$

while price keeps changing **do**

[1]. For each subgraph corresponding to the weight matrix $\mathbf{A}_l = \{a_{ij}^l\}$, compute for nodes in X_l that have not identified b edges:

$$j_1^l = \arg\max(a_{ij}^l - p_j^l),\ \forall j \in \text{adj}(i)$$

.....

$$j_{b+1}^l = \arg\max(a_{ij}^l - p_j^l),\ \forall j \in \text{adj}(i), j \neq j_1, \cdots j_b$$

[2]. Remove earlier selected edges that conflict

$$\mathcal{M} = \mathcal{M} \backslash \{(i', j_t^l) \in \mathcal{M}, \forall 1 \leq t \leq b, i' \in X_l\}$$

[3]. Add current selected edges

$$\mathcal{M} = \mathcal{M} \cup \{(i, j_t^l) \in \mathcal{M}, \forall 1 \leq t \leq b$$

[4]. Update price

$$p_{j_t^l}^l = a_{ij_t^l}^l - a_{ij_{t+1}^l}^l + p_{j_{t+1}^l}^l + \epsilon, \forall t = 1, \cdots, b$$

[5]. Synchronize price globally $p_j^l = \max_{t=1}^P p_j^t$

end while

3.3 Parallelization

In this subsection, we discuss a parallelization strategy of the auction algorithm for graph sparsification. Given the weight matrix $\mathbf{A}$ of a bipartite graph, we first partition $\mathbf{A}$ into L disjoint sub-matrices, where each sub-matrix $\mathbf{A}_l = \{a_{ij}^l\}$ is of size $n_l \times n$, $\sum_{l=1}^L n_l = n$. Therefore, sub-matrix $\mathbf{A}_l$ is the weight matrix of a bipartite subgraph G_l, which contains disjoint node sets X_l and Y_l. Although $\mathbf{A}$

Table 2: Average running time (seconds) on the synthetic bipartite and unipartite graphs using different methods for constructing sparse graphs. The graphs have nodes from 100 to 2000 and the value of b is set as 10 uniformly. The compared methods include Loopy Belief Propagation (LBP) [9], kNN, and the proposed parallel auction algorithm (PAA).

n	**Bipartite**				**Unipartite**			
	$\|\mathbf{E}\|$	LBP	kNN	PAA	$\|\mathbf{E}\|$	LBP	kNN	PAA
100	5×10^3	0.2	8×10^{-4}	4×10^{-3}	9.9×10^3	0.3	4×10^{-3}	1.1×10^{-2}
200	2×10^4	2.8	3×10^{-3}	1.5×10^{-2}	3.98×10^4	0.7	0.01	0.04
500	1.25×10^5	18.7	0.02	0.11	2.495×10^5	6.2	0.07	0.28
1000	5×10^5	37.9	0.11	0.53	9.99×10^5	23.2	0.39	1.26
2000	2×10^6	609.7	0.47	2.42	3.998×10^6	111.4	1.31	5.39

can also be partitioned by columns or into checkerboard structures [4], the above horizontal partitioning along row direction is more natural since large scale sparse graphs/matrices are often stored in the compress sparse rows format. After partitioning, we can rewrite the linear assignment problem in Eq. 2 as:

$$\begin{aligned} \max & \sum_{l=1}^{L} \sum_{i \in X_l} \sum_{j \in Y_l} a_{ij} b_{ij} \qquad (4) \\ \text{s.t.} & \sum_{j \in Y_l} b_{ij} = 1, \ \forall i \in X_l, l = 1, \cdots, L \\ & \sum_{l=1}^{L} \sum_{i \in X_l} b_{ij} = 1, \ \forall j \in Y_l, \quad b_{ij} \geq 0, \ \forall i, j. \end{aligned}$$

Eq. 4 implies a straightforward parallelization of auction algorithm for graph sparsification. In Algorithm chart 1, we summarize the major steps of parallel auction for graph sparsification. Here we set the initial values for $\vec{p}$ and $\vec{\pi}$ are both $\vec{0}$. The maximum number of edges allowed for a node is denoted by b and the set for selected edges is $\mathcal{M}$, which is initialized as an empty set $\emptyset$.

The above algorithm allows concurrent auction on a maximum of L processors, each of which handles one single bipartite subgraph. The only communication between processors occurs when synchronizing price vectors in step 5 of the above parallel algorithm. Note that the solution of the other dual variable $\vec{\pi}$ is updated implicitly, which is given by $\pi_i^l = a_{ij}^l - p_j^l$, where (i, j) is selected. The parallel auction typically accelerates computation due to the concurrent auction activities in each sub-matrix. However, it does not change the complexity. That is, in the worst case (the input graph is a chain), it is equivalent to a serial auction process.

3.4 Complexity Analysis

The time complexity for auction algorithm depends on the number of auction iterations and the workload for each iteration. In an auction process, a price of node increases at least for ϵ [3], i.e., the minimum bid increment. Since the profit for bidder i that matched with object j is given by $(a_{ij} - p_j)$, the profit becomes negative after $|a_{ij}|/\epsilon$ iterations, i.e., i will seek other objects for possibly higher profit. In each iteration, we visit at most $|\mathbf{E}|$ edges (the number typically decreases dramatically after a few iterations), so the overall complexity is $O(|\mathbf{E}| \max |a_{ij}|/\epsilon)$. During the auction process, the only information we must keep is the price of objects. There are n objects, so the space complexity is $O(n)$. Note that the space for storing the data or graph itself is not counted for evaluating space complexity. In Table 1, we summarize the time and space complexity of the proposed parallel auction algorithm based graph construction process and compare with other methods. From this analysis, it is easy to see the proposed parallel auction algorithm (PAA) based method is extremely efficient for large scale sparse graph data, which is indeed very common in real applications, such as social network and geometry analysis [7].

4 EXPERIMENTS

In this section, we provide empirical study of the proposed parallel auction algorithm (PAA) based graph construction method and compare with the-state-of-art approaches. In particular, we are interested in comparing with the belief propagation (BP) based methods [9, 10] since these method tends to generate high quality graphs with balanced node connectivity, similar as our proposed method. Note that there are two variants of the BP approach, loopy belief propagation (LBP) and a speed-up version, sufficient selection belief propagation (SSBP). However, in the empirical study, we notice that these two methods share very similar computational performance. Hence, we only report the results of LBP for comparison. In addition, we also implement the nearest neighbor search based graph construction using the standard sorting pro-

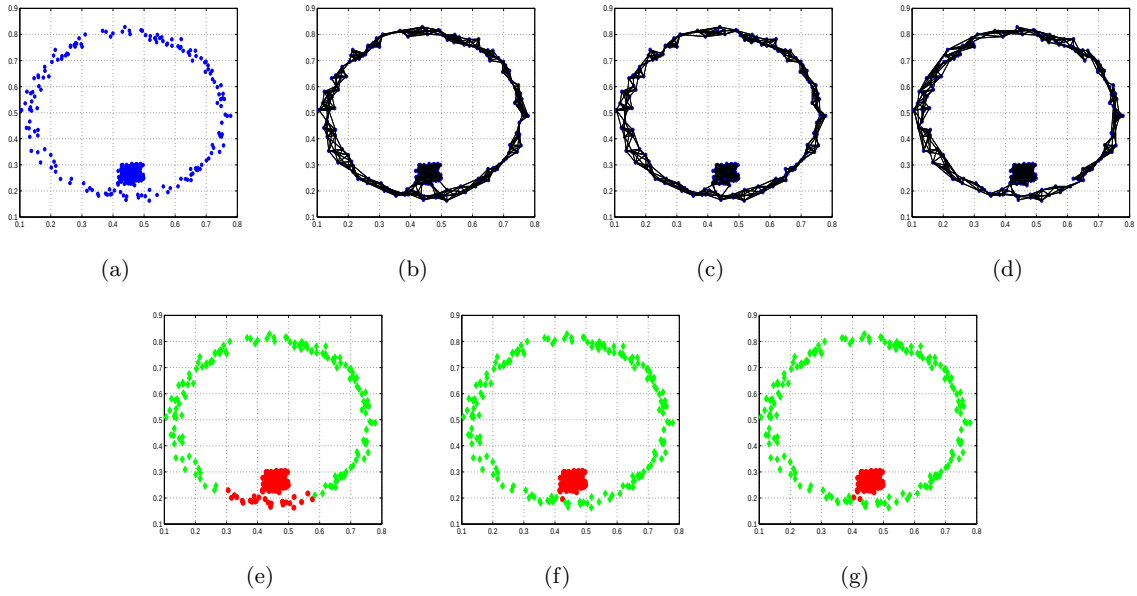


Figure 1: Illustration of the constructed graphs and clustering results over the toy datasets using different methods: a) original data; b) k-nearest neighbor graph ($k = 8$); c) b-matching graph ($b = 8$); d) PAA graph. Different color indicate the extract clusters using: e) kNN graph; f) b-matching graph; g) PAA graph.

cess as another baseline. Extensive experimenters over both synthetic and real benchmark datasets are performed to evaluate the computational efficiency and performance for different learning algorithms, as discussed below.

4.1 Speed Comparison

We first evaluate the running time of different graph construction methods. Similar to the settings in [9], here we randomly generate bipartite and unipartite graphs with $100 \leq n \leq 2000$ nodes, where the weights of graph edges are independently sampled from a uniform distribution $U(0, 1)$. For bipartite graph, each of the two disjoint node sets has $n/2$ nodes. For constructing a sparse graph, the degree of each nodes of the extracted sparse graph is uniformly set for all the experiments. For instance, in kNN approach, the value of k is set as 10, and accordingly $b = 10$ for b-matching methods. For fair comparison, all the experiments are conducted on the same platform with identical hardware and software environment.

For the LBP method, we use the default parameters in the software package [1], except setting the maximum iteration as 5000. Table 2 shows the comparison of the running time for the synthetic data. It is obvious that kNN provides the most efficient graph construction method due to its fast greedy search process. However, comparing LBP and PAA, both of which tend to generate balanced graphs, PAA improves the speed of graph construction in the magnitude of 10^2. In addition, we found that the propose method has more stable running performance for different random data, while both LBP method relies more on the properties of the synthetic graph data.

[1] http://www.cs.umd.edu/~bert/code/bmatching/

4.2 Performance Comparison

Graph representation has been widely used for different machine learning algorithms, including clustering and classification. In this subsection, we present the performance comparison study of clustering and classification through using different graph construction approaches. For clustering analysis, spectral clustering has been developed and shown effective in practice [16]. In a standard spectral clustering process, there are three key steps. The first step is to construct a data graph that connects sample points. Then one has to perform eigen-decomposition over the graph Laplacian. Finally, K-means algorithm is applied to derive the final clusters. In this study, we use the toy data [12] to construct different graphs, namely kNN graph, b-matching graph, and the graph built by the parallel auction algorithm (PAA-graph). Figure 4(a)-1(d) show the original data and different graphs. Then we feed these graph into standard spectral clustering algorithm to compare the results.

Figure 2 shows the error rate for the clustering results. The number of connected edges for graph nodes

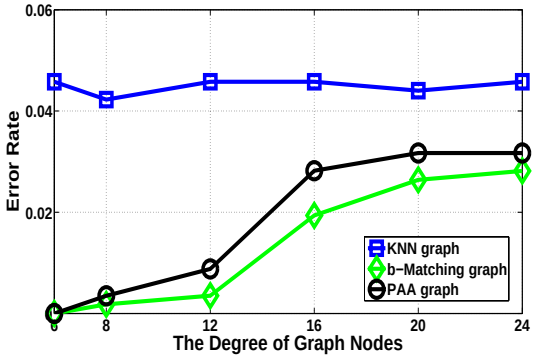
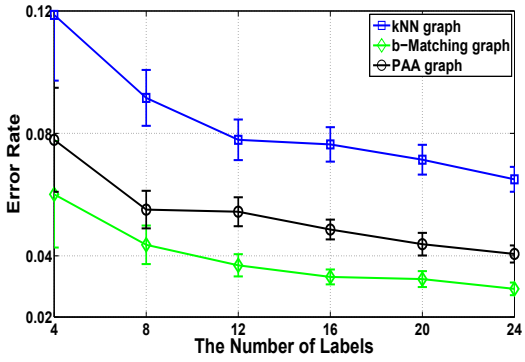


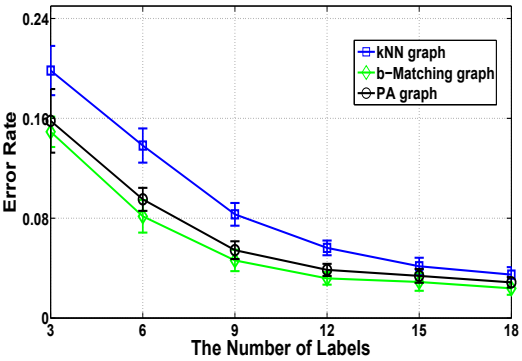
Figure 2: The performance comparison of spectral clustering using different graphs. The horizontal axis indicates the number of connected edges for graph nodes and the vertical axis shows the error rate of clustering results.

varies from 6 to 24. For all the tests, we use the same Gaussian kernel based weighting scheme with the kernel bandwidth informally set as 0.1. Clearly, kNN graph provides the worst performance and b-matching graph has the best performance. The proposed parallel auction based graph construction method has slightly worse performance than b-matching. However, it significantly increases the by the magnitude of 10^2 in practice. Figure 1(e)-1(g) shows the examples of constructed sparse graphs using different approaches. Due to the greedy search process, kNN method generates highly irregular graph and breaks the data structure by creating more cross-manifold edges. In contrast, b-matching and the proposed PAA method create more balanced graphs and have much less edges across the two clusters. When the node degree increases, it is easy to generate more cross-cluster edges. Hence, all the clustering performance of all three graph construction methods accordingly decreases.

In addition, we also compare the performance of graph based semi-supervised learning for classification tasks, i.e. classifying USPS handwritten digits [5] and face recognition [6]. In particular, we apply the method developed by Zhou et.al in [23] as the testbed for different graph construction methods. In the tests, we fix the number of node degree as 7 and use the same weighting scheme, similar to the settings used in the above clustering experiments. The number of initially given labels is varied from 4 to 20 for digits data and 3 to 18 for the face data. For each setting, we randomly run 50 independent trials and compute the average error rates of the classification results. Figure 3(a) and 3(b) show the performance curves with error bars for digits classification and face recognition, respectively. Again, b-matching approach builds the most balanced graph and therein achieves the best performance. The



(a)



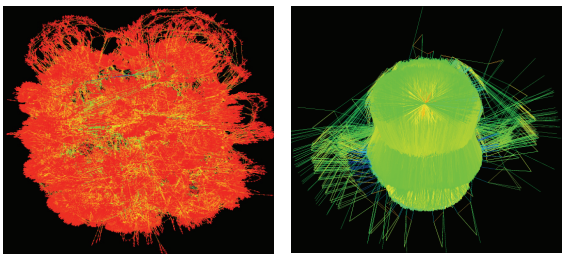
(b)

Figure 3: The performance comparison of graph based semi-supervised learning method using different graph construction methods: a) digits classification; b) face recognition. The horizontal axis indicates the number of initially given labels and the vertical axis shows the error rate of classification results.

proposed PAA method has slightly worse performance but almost as fast as kNN method, which tends to create the highly imbalanced graphs.

4.3 Evaluation of Parallel Efficiency

Since all the above experiments involve the graphs with relatively small scales, there is no extra benefit to parallelize the algorithm. However, when handling large scale data, it is very important to perform parallel process for fast graph construction. Note that the time complexity of the PAA based graph construction is linear with respect to the number of edges. This property makes the approach very suitable to the applications with large scale sparse data graphs. In order to validate the parallelization efficiency of the proposed method for such applications, here we used two real benchmark datasets about circuit simulation problems from the sparse matrix collection [7]. The first graph data "ASIC_680ks" contains 682, 712



(a) (b)

Figure 4: The visualization of the two large graph data: a) ASIC_680ks, and b) ASIC_680k.

nodes and $2,329,176$ edges (after reverse process) [2], and the second data "ASIC_680k" has $682,862$ nodes and $3,871,773$ edges (after reverse process) [3]. Figure 4 presents the visualization of these two large graphs[4]. From this figure, it is easy to see that both graphs are heavily irregular and the boundary nodes are with extremely sparse connectivity. Here we apply the proposed PAA method to recover sparse and balanced graphs from the original graph data.

Our experiments are performed using a computer cluster on which each compute node contains two 2.33 GHz Intel Xeon E5410 quadcore processors. The eight cores share 32 GB DDR2 RAM running at 800 MHz. The OS is Ubuntu 9.10 with GCC 4.10 installed. All codes were compiled with -O3 level option. We spawn parallel processes using MPICH2 to evaluate the parallel performance of the proposed PAA based graph construction method. In addition, we utilize MPICH2 so that our proposed technique can scale up to multiple compute nodes when the scale of the input graph is large enough. The graphs are partitioned evenly and launched to each process with equal number of nodes along with all the neighbor information.

For both graph data, graph construction is performed for two scenarios: $b = 2$ and $b = 4$. For those nodes with less than b edges in the original graph, all the connected edges will be preserved. We vary the number of used processes from 1 to 8, where each process is hosted by a separate core. The computational cost includes the time for generating parallel process, partitioning the graph, assigning tasks to each process, and performing the auction process (The time for loading data is not considered). Table 3 provides the running time in seconds for each tested case. From this table, it is easy to see that the proposed PAA-based graph construction method achieves lower execution time when multiple processes were used. However, the speedup is *sublinear* instead of linear because the extra time used for parallelization and communication. In general, when performing the graph construction in 8 parallel processes, the time cost will be reduced from one quarter to more than half. Especially for more complex problems (e.g., $b = 4$), the speed up is usually more significant.

Table 3: Evaluation of the parallel efficiency of the proposed PAA-based graph construction method. The numbers give the computational cost in seconds using different number of processes.

Data	**ASIC_680ks**		**ASIC_680k**	
# of processes	$b = 2$	$b = 4$	$b = 2$	$b = 4$
1	3.09	18.92	1.76	11.62
2	2.74	18.73	1.49	10.83
4	2.70	11.60	1.33	7.29
8	2.47	8.92	1.26	5.03

5 CONCLUSION

Data graph has been widely used for different machine learning and data mining algorithms. It is critical to build high quality graphs with affordable cost. Commonly, one aims in extracting sparse yet regular graph from data for both theoretic and practical advantages. Popular methods for constructing sparse graphs include nearest neighbor method and b-matching algorithm. The former is very efficient since it essentially perform greedy search to pick up the significant graph edges. However, it often generates highly imbalanced graphs with unsatisfactory performance. The later can provide fairly balanced regular graphs but suffers from extremely high computational cost.

In order to remain computational efficiency, while achieving certain properties of the extracted graph, such as sparsity and regularity, in this paper, we proposed a novel graph construction method through exploring the auction algorithm. In particular, the proposed method is almost as fast as kNN method, while maintain stratificatory regularity property. Experimental results show that the parallel auction algorithm based graph construction provides comparable performance to the b-matching algorithm, but improves the speed in the magnitude of 10^2. Finally, we also provide the empirical study of the parallelization efficiency over large data sets with up to 680K nodes. One of our future direction is to extend the proposed graph construction to semi-supervised and supervised scenarios, where partial label information are given to help improve accuracy of the auction process.

[2]http://www.cise.ufl.edu/research/sparse/matrices/Sandia/ASIC_680ks.html

[3]http://www.cise.ufl.edu/research/sparse/matrices/Sandia/ASIC_680k.html

[4]The visualization examples are available from the website: http://www.cise.ufl.edu/research/sparse/matrices/

Acknowledgements

This research was partially supported by the CRA/CCC Computing Innovation Fellow (CIFellow) Award. We would like to thank Dr. Anshul Gupta in the IBM T.J. Watson Research Center for the discussion on parallelization of the auction algorithm.

References

[1] A. Azran. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In Z. Ghahramani, editor, *Proceedings of the International Conference on Machine learning*, pages 49–56, Corvalis, Oregon, USA, June 2007.

[2] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[3] D. Bertsekas and D. Castañon. A forward/reverse auction algorithm for asymmetric assignment problems. *Computational Optimization and Applications*, 1:277–297, 1993.

[4] D. Bertsekas and D. Castanon. Parallel synchronous and asynchronous implementations of the auction algorithm*. *Parallel Computing*, 17(6-7):707–732, 1991.

[5] M. Breitenbach and G. Grudic. Clustering through ranking on manifolds. In *Proceedings of the 22nd international conference on Machine learning*, pages 73–80, 2005.

[6] M. Breitenbach and G. Grudic. Clustering through ranking on manifolds. In *Proceedings of the 22nd international conference on Machine learning*, pages 73–80, 2005.

[7] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1):1:1–1:25, 2011.

[8] H. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 448–456, 1983.

[9] B. Huang and T. Jebara. Loopy belief propagation for bipartite maximum weight b-matching. In *Int. Workshop on Artificial Intelligence and Statistics*, 2007.

[10] B. Huang and T. Jebara. Fast b-matching via sufficient selection belief propagation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[11] T. Jebara and V. Shchogolev. B-matching for spectral clustering. In *Proceedings of the European Conference on Machine Learning*, pages 679–686, 2006.

[12] T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 441–448, Montreal, Quebec, Canada, June 2009. ACM.

[13] D. Lazer, A. Pentland, L. Adamic, S. Aral, A.-L. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, G. King, M. Macy, D. Roy, and M. Van Alstyne. Computational social science. *Science*, 323(5915):721–723, 6 February 2009.

[14] W. Liu, J. He, and S. Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning*, pages 679–686, 2010.

[15] M. Maier, U. von Luxburg, and M. Hein. Influence of graph construction on graph-based clustering measures. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in neural information processing systems*, volume 22, pages 1025–1032. MIT Press, 2009.

[16] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[18] C. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Pubns, 1998.

[19] E. J. Riedy and J. Demmel. Sparse data structures for weighted bipartite matching. In *SIAM Parallel Processing for Scientific Computing*, pages 1–2, 2004.

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[21] X. Shi, J. Leskovec, and D. McFarland. Citing for high impact. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 49–58, 2010.

[22] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems 17*, volume 14, pages 945–952. MIT Press, Cambridge, MA, 2002.

[23] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, pages 321–328. MIT Press, Cambridge, MA, 2004.

[24] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. *Advances in neural information processing systems*, 16:169–176, 2004.

[25] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of International Conference on Machine Learning*, pages 912–919, 2003.

A Cluster-Cumulant Expansion at the Fixed Points of Belief Propagation

Max Welling
Dept. of Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA

Andrew E. Gelfand
Dept. of Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA

Alexander Ihler
Dept. of Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA

Abstract

We introduce a new cluster-cumulant expansion (CCE) based on the fixed points of iterative belief propagation (IBP). This expansion is similar in spirit to the loop-series (LS) recently introduced in [1]. However, in contrast to the latter, the CCE enjoys the following important qualities: 1) it is defined for arbitrary state spaces 2) it is easily extended to fixed points of generalized belief propagation (GBP), 3) disconnected groups of variables will not contribute to the CCE and 4) the accuracy of the expansion empirically improves upon that of the LS. The CCE is based on the same Möbius transform as the Kikuchi approximation, but unlike GBP does not require storing the beliefs of the GBP-clusters nor does it suffer from convergence issues during belief updating.

1 Introduction

Graphical models play a central role in knowledge representation and reasoning across a broad spectrum of scientific disciplines, in which probabilistic queries, or *inference* must be performed. A classic goal is to calculate the *partition function*, or sum over all possible configurations of the model; this task is central to many problems, such as computing the probability of observed data ("probability of evidence"), learning from data, and model comparisons.

Exact inference in these systems is typically NP-hard, motivating the need for efficient and accurate approximations. One of the most successful algorithms is *iterative belief propagation* (IBP) [7], which approximates the marginal probabilities of the distribution via an iterative, message-passing procedure. IBP can also be interpreted as a variational optimization, in which the messages and their approximate marginals, or beliefs, minimize a particular free energy approximation called the Bethe approximation [11, 3]. IBP is approximate on graphs with cycles, but is often empirically very accurate.

The wide success of IBP has also led to several techniques for improving its estimate quality. Generalized belief propagation (GBP) performs IBP-like updates on a set of *regions*, consisting of collections of variables. However, region selection is often difficult, and can even lead to convergence problems and degraded estimates [10]. An alternative is to use "series corrections" to the partition function such as the loop series and its generalizations, which compute modifications to the estimate provided by a given fixed point of IBP [1]. Such series expansions can be used to provide "any-time" algorithms that initialize to the IBP estimate and eventually converge to the exact solution.

In this paper we propose an alternative series correction to IBP estimates based on a cluster cumulant expansion (CCE). We show that the CCE has several benefits over the loop series representation, including more accurate partial estimates and more efficient aggregation of terms. We also show that our CCE representation is closely related to GBP, and that it can naturally be extended to GBP fixed points. We show the effectiveness of our approach empirically on several classes of graphical models.

2 Background

Let $X = \{X_i\}$ be a collection of random variables, each of which takes on values in a finite alphabet, $X_i = x_i \in \mathcal{X}$. For convenience, in the sequel we will not distinguish between the random variable X_i and its instantiation x_i. Suppose that the distribution on X factors into a product of real-valued, positive functions $\{\psi_f : f \in F\}$, each defined over a subset f of the variables:

$$p(\mathbf{x}) = \frac{1}{Z_\psi}\psi(\mathbf{x}) = \frac{1}{Z_\psi}\prod_{f\in F}\psi_f(x_f)$$

where $x_f = \{x_i \,|\, i \in f\}$ represents the arguments of factor ψ_f, and Z_ψ is the partition function, $Z_\psi = \sum_x \psi(\mathbf{x})$, which serves to normalize the distribution. We denote by F_i the set of factors that have x_i in their argument. A *factor graph* represents this factorization using a bipartite graph,

in which each factor (represented by a square) is connected to the variables (circles) in its argument. See Figure 1-Left.

2.1 Iterative Belief Propagation

With each factor or variable node in the factor graph (FG) we will associate a belief, denoted with $b_f(x_f)$ and $b_i(x_i)$ respectively. Iterative (or Loopy) Belief Propagation (IBP) is a message passing algorithm on the factor graph that attempts to make these beliefs consistent with each other. In particular, at a fixed point of IBP we require

$$\sum_{x_f \backslash x_i} b_f(x_f) = b_i(x_i) \tag{1}$$

for every pair of factor and variables nodes that are connected in the factor graph. IBP expresses these beliefs in terms of a set of messages m_{fi}:

$$\begin{aligned} b_i(x_i) &\propto \prod_{f \in F_i} m_{fi}(x_i) \\ b_f(x_f) &\propto \psi_f(x_f) \prod_{i \in f} \prod_{g \in F_i \backslash f} m_{gi}(x_i) \end{aligned} \tag{2}$$

and updates the messages iterative to achieve Eqn. (1):

$$m_{fi}^{\text{new}}(x_i) \leftarrow \delta(x_i) m_{fi}^{\text{old}}(x_i), \quad \delta(x_i) \doteq \frac{\sum_{x_f \backslash x_i} b_f(x_f)}{b_i(x_i)}$$

If the factor graph contains no cycles, at convergence the beliefs exactly equal the marginal probabilities of $p(\mathbf{x})$.

[11] showed that IBP fixed points correspond to stationary points of the Bethe variational free energy $\mathcal{F}$, giving an approximation to the log-partition function:

$$\begin{aligned} \log \hat{Z}_{BP} &= -\mathcal{F}(\{\psi_f\}, \{b_f, b_i\}) \\ &= \sum_{f \in F} \mathbb{E}_{b_f}[\log \psi_f] + \sum_{f \in F} \mathbb{H}(b_f) + \sum_{i \in V} (1 - |F_i|) \mathbb{H}(b_i) \end{aligned} \tag{3}$$

where V is the set of variables, $\mathbb{E}_{b_f}$ denotes the expectation under b_f and $\mathbb{H}(b_f) = -\sum_x b(x_f) \log b(x_f)$ is its entropy.

We may view the IBP updates as a *reparameterization* of the original distribution [9]. In particular, if we apply the message update from factor f to variable i, according to Eqn. (2) we change $b_i(x_i) \leftarrow b_i(x_i)\delta(x_i)$ and also a total of $|F_i|-1$ factor beliefs $b_g(x_g) \leftarrow b_g(x_g)\delta(x_i)$ with $g \in F_i \backslash f$ (see Figure 1-Left). As a result, the expression

$$\frac{1}{Z_b} b(\mathbf{x}) = \frac{1}{Z_b} \prod_{f \in F} b_f(x_f) \prod_{i \in V} b_i(x_i)^{1-|F_i|} \tag{4}$$

remains invariant under message updating, where Z_b is the partition function of the reparameterization $b(\mathbf{x})$. Moreover, if we initialize all messages to 1, we immediately see that Eqn. (4) is also equal to $\frac{1}{Z_\psi}\psi(\mathbf{x})$.

Since $\psi(\mathbf{x})$ and $b(\mathbf{x})$ correspond to the same normalized distribution $p(\mathbf{x})$, they differ by a constant multiplicative factor. Using (3) and noting that $\mathcal{F}(\{b\}, \{b_f, b_i\}) = 0$ shows that this factor is exactly $\hat{Z}_{BP}$, and so

$$Z_\psi = \hat{Z}_{BP} \cdot Z_b \tag{5}$$

This shows that the true partition function Z_ψ can be computed from IBP's estimate $\hat{Z}_{BP}$ and the reparameterization's normalization constant, Z_b.

2.2 Generalized Belief Propagation

Generalized BP, or GBP [11], generalizes (3) to include higher-order interactions than the original factors of the model. In GBP, one identifies a set of *regions* $\mathcal{R}$, in which each region $\alpha \in \mathcal{R}$ is defined to be a subset of the factors, $\alpha \subseteq F$. Each region is also given a "counting number" c_α, and messages are passed between regions; a data structure called a *region graph* is used to organize the message passing process. The GBP estimate of the partition function is

$$\log \hat{Z}_{GBP} = \sum_{f \in F} \mathbb{E}_b[\log \psi_f] + \sum_{\alpha \in \mathcal{R}} c_\alpha \mathbb{H}(b_\alpha). \tag{6}$$

BP on the factor graph constitutes a special case of GBP, in which the regions $\mathcal{R} = F \cup V$ and counting numbers $c_f = 1$ and $c_i = 1 - |F_i|$. Regions form a partial ordering defined by their set inclusion; the *ancestors* of region α are $an(\alpha) = \{\gamma \in \mathcal{R} | x_\alpha \subset x_\gamma\}$ and its *descendants* are $de(\alpha) = \{\beta \in \mathcal{R} | x_\alpha \supset x_\beta\}$, where x_α is the set of variables in region α.

The counting numbers should satisfy some basic properties; in [11], a RG is considered *valid* or *1-balanced* if

$$\sum_{\alpha \in R(f)} c_\alpha = 1 \; \forall \, f \qquad \sum_{\alpha \in R(i)} c_\alpha = 1 \; \forall \, i$$

where $R(f)$, $R(i)$ are those regions that contain factor f and variable i, respectively.

Validity can be ensured by assigning each factor f to a single outer region, setting $c_\alpha = 1$ for outer regions (i.e. regions with no parents) and recursively for inner regions as:

$$c_\alpha = 1 - \sum_{\beta \in an(\alpha)} c_\beta. \tag{7}$$

The GBP message updates can again be interpreted as reparameterization updates on the region graph. Any two regions that are connected through a directed edge will exchange messages from parent to child region. We write the beliefs in terms of the potentials and messages to α and its descendants, $\Delta_\alpha = \alpha \cup de(\alpha)$,

$$b_\alpha(x_\alpha) = \frac{1}{Z_\alpha} \prod_{f \in \alpha} \psi_f(x_f) \prod_{\substack{\gamma \in \{an(\Delta_\alpha) \backslash \Delta_\alpha\} \\ \beta \in \Delta_\alpha}} m_{\gamma\beta}(x_\beta)$$

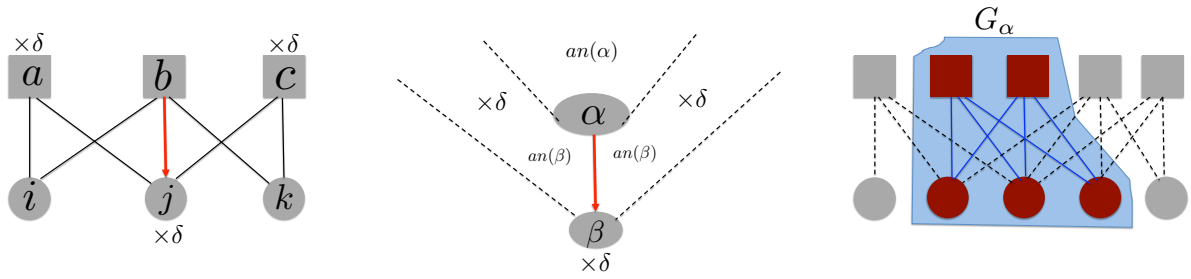


Figure 1: **Left:** A reparameterization update on a factor graph. **Middle:** A region graph GBP reparameterization update. **Right:** Factor subgraph G_α corresponding to a cluster-cumulant.

and write a fixed-point message update as,

$$m^{\text{new}}_{\alpha\beta}(x_\beta) \leftarrow \delta(x_\beta) m^{\text{old}}_{\alpha\beta}(x_\beta) = \frac{\sum_{x_\alpha \setminus x_\beta} b_\alpha(x_\alpha)}{b_\beta(x_\beta)} m^{\text{old}}_{\alpha\beta}(x_\beta)$$

In words, to compute the belief at region α we collect messages from all regions that are either ancestors or ancestors of descendants of that region, and that flow into α or its descendants. Those messages that originate in α or its descendants are not used.

It is now not difficult to see that these updates constitute a reparameterization. Namely, if we update a message $m_{\alpha\beta}$, the update factor δ is multiplied into the expressions for b_β and all its ancestors, except b_α and all its ancestors. Using the expression for the counting numbers (7) we thus see that we accumulate a total number of terms,

$$c_{\text{total}} = [c_\beta + \sum_{\gamma \in an(\beta)} c_\gamma] - [c_\alpha + \sum_{\delta \in an(\alpha)} c_\delta] = 0$$

where we used that $c_\beta = 1 - \sum_{\gamma' \in an(\beta)} c_{\gamma'}$ and similarly for c_α. This is illustrated in Figure 1-Middle.

As a result, the following expression will stay invariant under the GBP updates:

$$\frac{1}{Z_\psi} \prod_f \psi_f(x_f) = \frac{1}{Z_b} \prod_\alpha b_\alpha(x_\alpha)^{c_\alpha} \tag{8}$$

We can now use similar arguments as in Section 2.1 to show that $Z_\psi = \hat{Z}_{GBP} \cdot Z_b$ again holds.

2.3 The Loop Series Expansion

The loop series [1, 8, 2] expresses Z_b as a finite series expansion of terms related to the generalized loops of the graph. Starting from Eqn. (4) we rewrite the terms:

$$\frac{b_f(x_f)}{\prod_{i \in f} b_i(x_i)} = 1 + \frac{b_f(x_f) - \prod_{i \in f} b_i(x_i)}{\prod_{i \in f} b_i(x_i)} = 1 + U_f(x_f)$$

where U_f is like a covariance in that it measures dependence among the variables $x_i, i \in f$. We can then write

$$Z_b = \mathbb{E}_{\tilde{b}}\Big[\prod_{f \in F} (1 + U_f)\Big] \quad \text{where} \quad \tilde{b}(\mathbf{x}) = \prod_i b_i(x_i).$$

Expanding the product gives one term for each element in the power set of F. If we expect the individual terms U_f to be small, it makes sense to organize this series (for example) in order of increasing number of U_f terms, and truncate it once some computational limit is reached. Each term corresponds to a subset $\alpha \subseteq F$, and thus to a subgraph of the factor graph. One can show that if this subgraph contains any factor f that is *not* part of a loop (a.k.a. part of a dangling tree), then the corresponding term will be zero.

To make the connection with the formulation of [1], we must assume a binary alphabet $x_i \in (0, 1)$. Following [8] we can re-express,

$$\frac{b_f(x_f) - \prod_{i \in V_f} b_i(x_i)}{\prod_{i \in V_f} b_i(x_i)} = \sum_{v \subseteq V_f, |v| \geq 2} \beta_v \prod_{i \in v} (x_i - \mathbb{E}[x_i])$$

$$\beta_v = \frac{\mathbb{E}[\prod_{i \in v}(x_i - \mathbb{E}[x_i])}{\prod_{j \in v} \mathbb{V}(x_j)} \tag{9}$$

where $\mathbb{V}(x)$ denotes the variance of x. In the process we have created even more terms than the powerset of all factors because each factor itself is now a sum over all its subsets of variables with cardinality larger than one. The total set of terms is in one-to-one correspondence with all the "generalized loops" of the factor graph (graphs where every variable and factor node have degree at least two). The advantage for the binary case is that the final expansion can now be written in terms of expectations of the form $\mathbb{E}[(x_i - \mathbb{E}[x_i])^d]$ which admit closed form expressions in terms of the beliefs b_i obtained from IBP at its fixed point. We refer to [1, 8] for further details.

3 The Cluster-Cumulant Expansion

We now develop an alternative expansion of the log-partition function which we call the cluster-cumulant expansion (CCE). As with the loop series, we begin analysis at a fixed point of IBP; we will later relate our expansion to GBP and extend its definition to include GBP fixed points.

The CCE is defined over an arbitrary collection of subsets of factors which we will denote with α. We define

a partial ordering on this set through subset inclusion, i.e. $\alpha \leq \beta$ iff $\alpha \subseteq \beta$ and $\alpha < \beta$ iff $\alpha \subset \beta$. We will denote this poset with Ω. We also define a factor subgraph, $G_\alpha = \{i, f | i \in V_\alpha, f \in \alpha, \alpha \in \Omega\}$ where V_α corresponds to the set for variables that occur in the arguments of the factors $f \in \alpha$. An example of such a factor subgraph is provided in Figure 1-Right. Finally we will define the partial probability distribution over this factor subgraph as P_α and the corresponding partial log partition function, $\log Z_\alpha$ as its log normalization constant:

$$P_\alpha(x_\alpha) = \frac{1}{Z_\alpha} \prod_{i \in V_\alpha} b_i(x_i) \prod_{f \in \alpha} \frac{b_f(x_f)}{\prod_{i \in V_f} b_i(x_i)} \tag{10}$$

$$\log Z_\alpha = \log \sum_{x_\alpha} \prod_{i \in V_\alpha} b_i(x_i) \prod_{f \in \alpha} \frac{b_f(x_f)}{\prod_{i \in V_f} b_i(x_i)} \tag{11}$$

We first observe that if we choose $\alpha = F$ (the set of all factors) then $\log Z_F$ is the exact log partition function $\log Z_b$. We now show that $\log Z_\alpha = 0$ when the corresponding factor subgraph G_α is a tree:

THEOREM **1.** *If the factor subgraph G_α is* singly connected*, then* $\log Z_\alpha = 0$.

Proof. When G_α is a tree, the expression (10) represents the exact joint probability distribution expressed in terms of its marginals with $Z_\alpha = 1$. Thus the result follows. □

As a corollary we observe that $\log Z_\alpha$ vanishes if α consists of a single factor f. The theorem also allows us to remove all "dangling trees" from any factor subgraph by marginalizing out the variables that correspond to the dangling tree. We will thus define the "core" of a factor subgraph as the part of the factor subgraph that remains after all dangling trees have been removed (see also [2]).

We are now ready to define the cluster-cumulants (CCs). The idea is to decompose each partial log partition function as a sum of cluster-cumulant contributions from clusters in the same or lower levels of the partially ordered set $\alpha \in \Omega$. In this manner, we expect that the lowest order CCs generate the largest contribution and that higher order CCs subsequently represent increasingly small corrections. The definition of the CCs is provided by the expression [5, 4]

$$\log Z_\alpha = \sum_{\beta \leq \alpha} C_\beta \tag{12}$$

This relation can be inverted using a Möbius transform,

$$C_\alpha = \sum_{\beta \leq \alpha} \mu_{\beta,\alpha} \log Z_\beta \tag{13}$$

where we define the Möbius numbers as,

$$\mu_{\alpha,\alpha} = 1 \quad \text{and} \quad \mu_{\gamma,\alpha} = -\sum_{\beta : \gamma < \beta \leq \alpha} \mu_{\beta,\alpha} \quad \text{if } \gamma < \alpha.$$

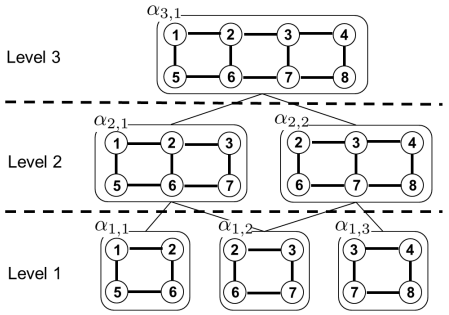


Figure 2: A poset for a pairwise MRF on a 1×3 grid. Truncating this poset at level 1 would include clusters over the faces of the grid. Truncating at level 2 would add clusters $\alpha_{2,1}$ and $\alpha_{2,2}$.

To approximate $\log Z \approx \log Z_\ell$ we truncate the cumulant series by only including clusters $\alpha \in \Omega_\ell$ up to some level ℓ in the poset:

$$\log Z_\ell = \sum_{\alpha \in \Omega_\ell} C_\alpha = \sum_{\alpha \in \Omega_\ell} \sum_{\beta \leq \alpha} \mu_{\beta,\alpha} \log Z_\beta \tag{14}$$

$$= \sum_{\beta \in \Omega_\ell} \sum_{\alpha \in \Omega_\ell} \mu_{\beta,\alpha} \mathbb{I}[\beta \leq \alpha] \log Z_\beta \doteq \sum_{\beta \in \Omega_\ell} \kappa_\beta \log Z_\beta$$

where $\mathbb{I}[\cdot]$ is the indicator function. Moreover, $\kappa_\beta \doteq \sum_{\alpha \in \Omega_\ell} \mu_{\beta,\alpha} \mathbb{I}[\beta \leq \alpha]$ can be computed recursively as

$$\kappa_\beta = 1 - \sum_{\alpha \in an(\beta)} \kappa_\alpha \tag{15}$$

with $an(\beta)$ the ancestors of cluster β in the Hasse diagram corresponding to the poset and with $\kappa_\alpha = 1$ for the clusters at the highest level of Ω_ℓ (i.e. the clusters with no parents) [6]. It is important to realize that this truncation is *not* equivalent to simply including all partial log partitions from the the highest level in the poset, i.e. $\log Z_\ell = \sum_{\alpha \in \Omega_\ell} C_\alpha \neq \sum_{\alpha \in \Omega_\ell} \log Z_\alpha$.

Figure 2 illustrates the computation of $\log Z_\ell$ on a simple pairwise MRF. On this 1×3 grid, the approximation truncated at level 2 is $\log Z_2 = \log Z_{\alpha_{2,1}} + \log Z_{\alpha_{2,2}} - \log Z_{\alpha_{1,2}}$. The computation of each partial log partition function requires exact inference on the partial factor subgraph. Thus, the time complexity for an approximation including c clusters is $O(c|V| \exp(w))$, where w is the largest induced width of the c clusters. However, the space complexity of the truncated approximation is $O(|V| \exp(w))$ since we need only retain $\log Z_\alpha$ for each cluster.

This expansion is similar to the expansion of the free energy (or the entropy) in the cluster variation method of GBP (see Section 2.2). However, there clusters serve as regions between which messages are exchanged during the execution of GBP. In contrast, in this paper CCs are calculated after IBP has converged in order to compute corrections.

The cumulant definition in Eqn. (13) immediately shows that cumulants of singly connected subgraphs vanish:

THEOREM **2.** *If the factor subgraph G_α is* singly connected*, then the cluster-cumulant satisfies $C_\alpha = 0$.*

Proof. From Eqn. (13) we see that a cumulant can be written as a linear combination of $\log Z_\alpha$ and $\log Z_\beta,\ \beta < \alpha$. Using Theorem 1 together with the facts that every subgraph of a tree is a smaller tree (or a forest of trees) and that at the lowest level of the poset we have $C_\alpha = \log Z_\alpha$, the result follows by induction. □

Unlike the terms in the loop series [1] (see Section 2.3), cluster-cumulants corresponding to *disconnected* subgraphs also vanish, even if the disconnected components are not singly connected.

THEOREM **3.** *If the factor subgraph G_α is* disconnected*, then the cluster-cumulant satisfies $C_\alpha = 0$.*

Proof. If the graph G_α is disconnected, its probability P_α factorizes: $P_\alpha = P_{\alpha_A} P_{\alpha_B}$. Then, the partial partition function decomposes as $\log Z_\alpha = \log Z_{\alpha_A} + \log Z_{\alpha_B}$ which by Eqn. (12) we rewrite as $\log Z_\alpha = \sum_{\beta_A \le \alpha_A} C_{\beta_A} + \sum_{\beta_B \le \alpha_B} C_{\beta_B}$. Again by Eqn. (12) we have

$$\begin{aligned} C_\alpha &= \log Z_\alpha - \sum_{\beta<\alpha} C_\beta \\ &= \sum_{\beta_A \le \alpha_A} C_{\beta_A} + \sum_{\beta_B \le \alpha_B} C_{\beta_B} - \sum_{\beta<\alpha} C_\beta. \end{aligned}$$

Clusters $\beta < \alpha$ in the poset fall into one of three categories: either $C_\beta \le C_{\alpha_A}$, or $C_\beta \le C_{\alpha_B}$, or $[(C_\beta > C_{\alpha_A}) \wedge (C_\beta > C_{\alpha_B}) \wedge (C_\beta < C_\alpha)]$. We now proceed by induction. Cumulants in the first two categories will cancel in the expression for C_α. Sufficiently small clusters will have no cumulants in the third category, and so must be zero. For larger clusters, cumulants in the third category correspond to smaller clusters that must also be disconnected, which by the inductive argument must vanish. The result follows. □

This property is special to the cumulant expansion and holds even though the partial log-partition function $\log Z_\alpha$ for the disconnected region does *not* vanish. Apart from reducing the number of terms that must be considered, this property also suggests (by continuity of the cumulants as a function of their factor parameters) that cumulants with nearly-independent components must also be small. Thus, we expect significant contributions from tight, highly correlated sets of variables and factors, but small contributions from clusters with components that are almost independent.

To emphasize this point, consider a cluster-cumulant corresponding to a subgraph G_α that has one uniform factor $\psi_f(x_f) = 1$. Moreover, assume that the cluster $\tilde{\alpha}$ given by removing this factor from α is also in the poset. This creates a situation where cluster α is ranked higher in the poset than $\tilde{\alpha}$, i.e., $\alpha > \tilde{\alpha}$, yet their partial partition functions must be the same: $\log Z_\alpha = \log Z_{\tilde{\alpha}}$. In this case, $C_\alpha = 0$:

THEOREM **4.** *Consider a poset Ω that includes a factor f with unit potential $\psi_f(x_f) = 1$, and a cluster α that contains factor f. Assume that there is also a cluster $\tilde{\alpha} < \alpha$ containing the same factors as α except f, i.e. $\tilde{\alpha} = \alpha \backslash f$. Then, $C_\alpha = 0$.*

Proof. For both clusters $\{\alpha, \tilde{\alpha} | \alpha > \tilde{\alpha}\}$ we must have the same partial partition function, $\log Z_\alpha = \log Z_{\tilde{\alpha}}$. Hence, from Eqn. (12) we have $\sum_{\beta \le \alpha} C_\beta = \sum_{\tilde{\beta} \le \tilde{\alpha}} C_{\tilde{\beta}}$. Since $\alpha > \tilde{\alpha}$ it follows that $\sum_{\beta \le \alpha} C_\beta = \sum_{\tilde{\beta} \le \tilde{\alpha}} C_{\tilde{\beta}} + \sum_{\gamma > \tilde{\alpha}, \gamma \le \alpha} C_\gamma$. However, since $\tilde{\alpha}$ is defined to have exactly one factor fewer than α we have that $\sum_{\gamma > \tilde{\alpha}, \gamma \le \alpha} C_\gamma = C_\alpha$. Combining these expressions we have $C_\alpha = 0$. □

The significance of Theorem 4 is illustrated by imagining a situation where $\psi_f(x_f) \approx 1$. By the continuity argument, we expect $C_\alpha \approx 0$ if the cluster $\tilde{\alpha}$ with factor f removed is in the poset. Hence, a cluster-cumulant will only significantly differ from zero if it introduces new dependencies that were not already captured by lower order cumulants. This result strengthens the interpretation of CCE as an expansion in terms of orders of statistical dependency.

These considerations suggest a way to build posets that are likely to deliver good truncated series approximations of the log-partition function. We first choose a collection of clusters which may be overlapping, but none are a subset of any other cluster. These clusters should be chosen so that 1) they are not disconnected and 2) they have no dangling trees. Moreover, tightly coupled clusters with strong interdependencies are preferred over ones with weak dependence. We then generate all intersections, intersections of intersections, etc., to construct our poset. Finally we compute their partial log partition functions and combine them using Eqn. (14).

4 CCE for Region Graphs

We now extend the cluster-cumulant expansion to region graphs; see Section 2.2 and [11] for background on region graphs. The CCE will be build on top of an existing region graph. We will call the existing region graph "calibrated" (abbreviated CRG) if the GBP fixed point equations have converged.[1] We will add new "uncalibrated" regions to the region graph during the expansion (see Figure 3-Left). Any new region will become a parent of all the existing regions that it contains. We then define the region subgraph $G_\alpha = G_{\Delta_\alpha}$ to be the collection of *calibrated* regions and parent-child relations restricted to α and its descendants. Note that G_α includes only regions in the calibrated part of the region subgraph, i.e., below the dashed line in Figure 3-Left.

[1] We only consider CCEs on calibrated region graphs; although an expansion can be established for uncalibrated RGs, Theorem 2 will not hold (increasing the number of non-zero cumulants) and terms may not fall off quickly, leading to poor approximations.

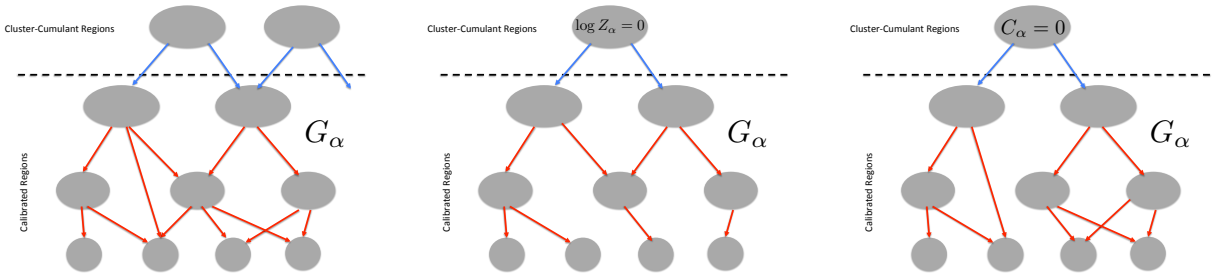


Figure 3: **Left:** Cluster-cumulant extension of the calibrated region graph. **Middle:** A tree-structured RG for which $\log Z_\alpha = 0$. **Right:** Illustration of a disconnected RG for which $C_\alpha = 0$.

We define counting numbers c_β^α for regions β in G_α by

$$c_\beta^\alpha = 1 - \sum_{\gamma \in (G_\alpha \cap an(\beta))} c_\gamma^\alpha$$

Note that a counting number c_β^α can vanish for some regions in G_α even though their counting number c_β in the original region graph did not vanish. Regions with $c_\beta^\alpha = 0$ will not contribute to $\log Z_\alpha$ and can be excluded from G_α.

The partial log-partition function for this region is,

$$\log Z_\alpha = \log \sum_{x_\alpha} \prod_{\beta \subseteq \alpha} b_\beta(x_\beta)^{c_\beta^\alpha} \tag{16}$$

To generate cumulants we again use Eqn. (13). The truncated cumulant series is the sum of these cumulants up to some level in the poset. More convieniently, we can instead use Eqn. (14) to express the series in terms of partial log partition functions. Note that this will require counting numbers κ_α that should not be confused with the counting numbers c_α above. In particular, the c_α are computed strictly in terms of the calibrated part of the region graph (e.g., the regions right below the dashed line in Figure 3-Left have $c_\alpha = 1$), while the κ_α are computed strictly in terms of the cluster-cumulants (e.g., the regions at the highest level in the extended region graph have $\kappa_\alpha = 1$).

As with the IBP expansion, $\log Z_\alpha$ vanishes for a region subgraph G_α that is singly connected (after only regions with $c_\beta^\alpha \neq 0$ have been retained):

THEOREM **5.** *If a region subgraph G_α consisting of regions in the CRG that are descendants of α and that have counting numbers $c_\beta^\alpha \neq 0$ is* singly connected*, then the partial log partition function and the cumulant satisfy* $\log Z_\alpha = 0$, $C_\alpha = 0$.

Proof. Since the marginals on G_α are calibrated (due to the reparameterization property of GBP) and because G_α is singly connected, we have $Z_\alpha = 1$. Since according to Eqn. (13) cumulants are linear combinations of log-partition functions over subsets and since all subsets must also be singly connected (or disconnected), the cumulant for a singly connected region must also vanish. □

This idea is illustrated in Figure 3-Middle. More generally, for any region subgraph we can remove "dangling trees" for the purpose of computing the CCE. (Note that this removal may change the counting numbers of regions connecting the core to the dangling trees.) To see this, follow an elimination order from the tree's leaf regions to the core. Since it is just message passing on the (hyper) tree and since the tree was already at a fixed point of GBP, the marginalization result will be equivalent to removing those regions.

Analogous to the factor graph case, cluster-cumulants defined on a region subgraph with two or more disconnected components vanish. The proof is identical to the one given for Theorem 3, and so we simply state the result:

THEOREM **6.** *If a region subgraph G_α consisting of regions in the CRG that are descendants of α and that have counting numbers $c_\beta^\alpha \neq 0$ is* disconnected*, then the cluster-cumulant satisfies $C_\alpha = 0$.*

This is illustrated in Figure 3-Right. Note again that the partial log partition function may not vanish; rather, it will equal the sum of two (or more) terms whose contributions are already included in the lower order cumulants.

In building our CCE, it is clear that we should avoid CCs that correspond to either disconnected or singly connected region subgraphs. (However, if we happen to include them, no harm is done to the approximation; their contribution will be zero). More generally, Theorem 4 remains valid for region graphs, suggesting to define CCs over tightly connected groups of regions that are expected to exhibit strong dependencies beyond what are already modeled by the calibrated region graph. This still leaves considerable freedom to design a good CCE. The experiments in Section 5 will provide further guidance in his respect.

We end this section with a different perspective of what is accomplished with the CCE. Starting with Eqn. (5) we can write

$$\begin{aligned} \log Z_\psi &= \log \hat{Z}_{BP} + \log Z_b \\ &= -\sum_{\alpha \in \mathcal{R}_{\text{calibrated}}} c_\alpha \mathcal{F}_\alpha - \sum_{\alpha' \in \mathcal{R}_{\text{uncalibrated}}} \kappa_{\alpha'} F_{\alpha'} \end{aligned} \tag{17}$$

where $\mathcal{R}_{\text{calibrated}}$ and $\mathcal{R}_{\text{uncalibrated}}$ are calibrated and uncalibrated regions and $F_\alpha = -\log Z_\alpha$. The first term is the standard decomposition of the variational free energy for region graphs (see e.g. [11]) and the second term is the cluster cumulant expansion. This expression highlights the fact that CCE can be considered a correction to the variational free energy. Moreover, it suggests a procedure where increasingly many regions are moved from the uncalibrated part of the region graph to the calibrated part of the region graph. We leave exploration of these ideas for the future.

5 Experiments

We conducted a variety of experiments to study the CCE. Since the CCE is defined on a collection of clusters, we first describe the cluster choices used in our experiments. This description primarily refers to clusters that are sets of factors, i.e., for CCE on a factor graph. CCE on a region graph considers clusters that are sets of calibrated regions. In the sequel a k-cluster is a collection of k factors (or regions).

We considered two different collections of clusters in our experiments. We first considered the poset Ω_{all} containing all pairs of factors, all triplets of factors, and so forth. Enumerating factors in this manner quickly becomes unmanageable, so the expansion must be truncated. We will use Ω^l_{all} to denote truncating the poset at level l, where for example Ω^4_{all} denotes the series truncated after all quintuplets of factors have been included.

The second collection of factors considered come from the Truncated Loop Series (TLS) algorithm of [2]. The TLS algorithm finds a subset of all generalized loops in a factor graph. It does so by first finding a set of S simple loops (i.e., cycles in the factor graph with degree 2) and then *merges* these simple loops to create a set of generalized loops (i.e., cycles with degree ≥ 2). In the TLS algorithm, a generalized loop is formed from two simple loops l and l' by finding a path in the factor graph from some factor or variable in l to some factor or variable in l'. Since many paths may connect two simple loops, the set of generalized loops is restricted to paths of at most length M.

The authors of [2] provide code that enumerates a set of generalized loops l_1,...,l_N and computes the LS approximation to $\log Z$ after every loop. In [2], the set of loops are placed in descending order by the magnitude of their contributions, $|U_l|$ (see 2.3 for details). To make the TLS algorithm an "anytime" algorithm, we do not post-process the set of loops and instead report the LS approximation on the order in which the loops are discovered. A sequence α_1,...,α_N of clusters can be constructed from a sequence of loops $l_1, .., l_N$, where α_i is the set of factors in generalized loop l_i [2]. In the experiments that follow 'LS (TLS)' denotes the Loop Series approximation on the (unsorted) sequence of loops from the TLS algorithm and 'CCE (TLS)' is the CC approximation on the unsorted sequence of loops.

When adding clusters to the CCE, it is important to note that the collection of clusters may become imbalanced in that the approximation in Eqn. (14) does not include all intersections (and intersections of intersections etc.) of all clusters, which leads to over-counting in the CCE. While adding clusters bottom up along the poset Ω_{all} guarantees a balanced CCE, this is not always true for the TLS sequence leading to suboptimal results. We thus emphasize that we include CCE (TLS) results for the sake of comparison, but that it is not the expansion that we recommend for the CCE.

We ran experiments on synthetic Markov Network (MN) instances as well as benchmark instances from the UAI-2008 solver competition. In all experiments, we take the absolute difference of true and approximate log-partition functions $|\log Z - \log \hat{Z}|$ as our error measure. We stop BP and GBP when $L_\infty(b_p(\mathbf{x}_c), b_c(\mathbf{x}_c)) < 1e^{-8}$, where $b_c(\mathbf{x}_c)$ is the belief at a child region c, $b_p(\mathbf{x}_c)$ is the marginal belief of parent region p on variables $\mathbf{x}_c$ and L_∞ is the maximum absolute difference between the two beliefs.

5.1 Grids

We first tested pairwise MNs defined on 10×10 grids. Each grid instance has unary potentials of the form $f_i(x_i) = [\exp(h_i);\ \exp(-h_i)]$ and pairwise potentials of the form:

$$f_{ij}(x_i, x_j) \quad = \quad \begin{bmatrix} \exp(w_{ij}) & \exp(-w_{ij}) \\ \exp(-w_{ij}) & \exp(w_{ij}) \end{bmatrix}$$

The values of h_i and w_{ij} were drawn from $\mathcal{N}(0, \sigma_i^2)$ and $\mathcal{N}(0, \sigma_{ij}^2)$ distributions, respectively. To study the behavior of the CCE at various interaction strengths, we fixed $\sigma_i = 0.1$ and varied σ_{ij} from 0.1 to 1. Reported errors are averages across 25 instances at each setting, and error bars indicate the standard error.

Figure 4 compares the different series approximations on grids. Following [2], the TLS algorithm was run on the 10×10 grid instances with $S = 1000$ and $M = 10$.

'CCE (BP)' is the CC approximation given a sequence of terms efficiently enumerated in Ω^{15}_{all}. All pairs and triplets of factors on a pairwise grid have zero contribution and can be ignored. The only 4-clusters with non-zero contribution are the 81 faces of the 10×10 grid. To fill out the remaining levels of Ω^{15}_{all}, we enumerate pairs, triplets and quadruplets of connected faces (sharing a vertex or edge), since the CC for two disconnected faces in the grid is zero.

'CCE (GBP)' is the CCE on a region graph. GBP was run on a region graph with outer regions equal to the faces of the grid. In this case, all pairs and triplets of faces give zero contributions and can be ignored. Each 3×3 subgraph in the grid is comprised of 4 faces forming a cycle. We take all

[2] Since many generalized loops are defined over the same set of factors, we consider only the unique sets of factors.

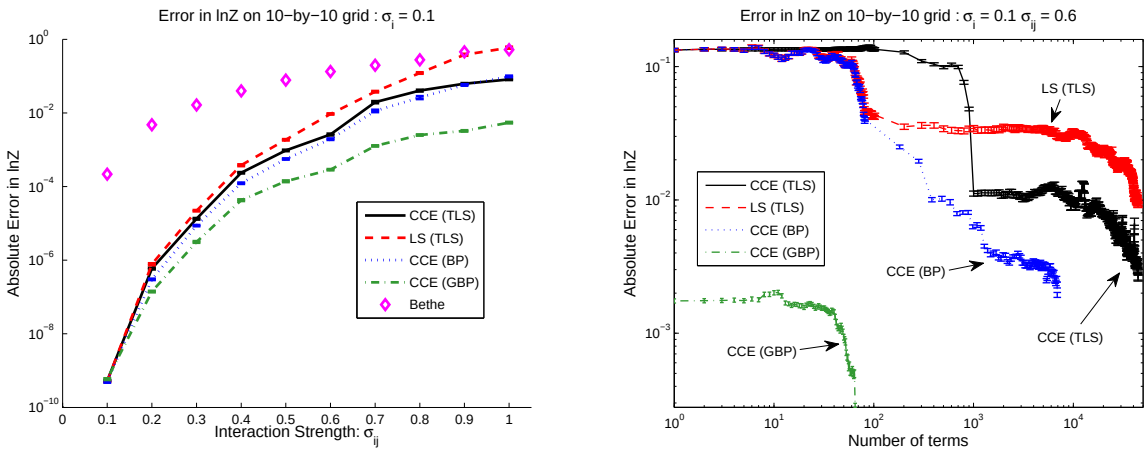


Figure 4: **Left:** Comparison of the 4 series approximations on 10×10 grids at a variety of interaction strengths. **Right:** $Error_Z$ versus the number of terms in each approximation, for $\sigma_{ij} = 0.6$.

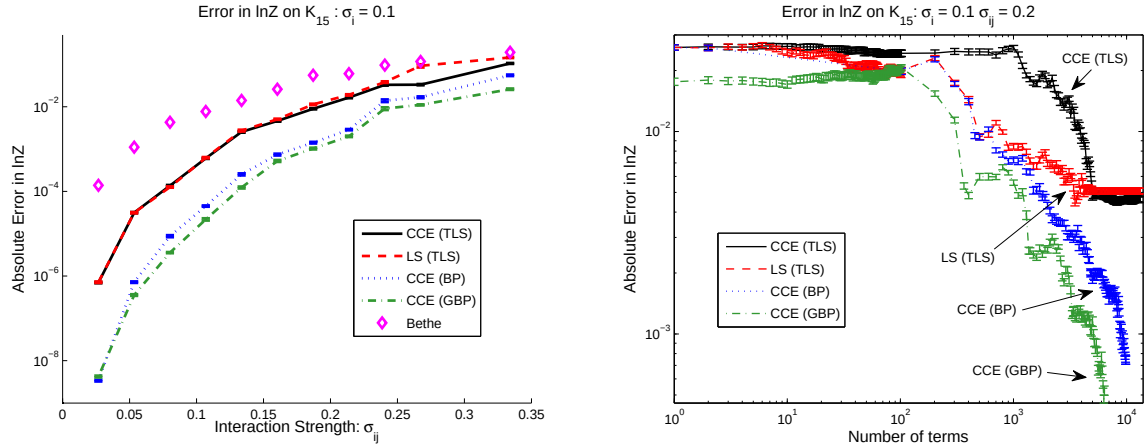


Figure 5: **Left:** Comparison of the 4 series approximations on complete MNs of 15 variables at different coupling levels. **Right:** $Error_Z$ as a function of the number of terms in each series.

64 of these to be the clusters in the 'CCE (GBP)' approximation. Importantly, the intersection of any two such clusters will be a cluster on either a disconnected or acyclic subgraph (having a zero contribution) implying that the whole collection of clusters remains balanced.

Figure 4-Left compares the series approximations at a variety of coupling levels. The error reported is w.r.t. the final $\log Z$ approximation of each series. All the approximations offer a substantial error reduction over the Bethe approximation for instances with weak interactions. As interaction strength is increased, the relative improvement of each series declines. At the strongest coupling level, the error in the LS approaches the Bethe approximation error, while CCE remains an order of magnitude better.

Figure 4-Right shows a trace of the error as a function of the number of terms included. Each LS term can be computed very efficiently for binary MNs, while each CCE term requires inference on a set of factors (or regions). Thus, comparing the series on the the number of terms may seem to unfairly favor the CC approximations. However, the LS requires enumerating generalized loops in the MN which is more expensive than enumerating clusters.

5.2 Complete Graphs

We also experimented on pairwise MNs over a complete graph of 15 variables. Here 'CCE (BP)' is the CC approximation given an efficient enumeration of terms in Ω^{14}_{all}. Since all pairs of factors are acyclic (and have zero contribution), we begin by considering all cycles of length 3, which is equivalent to all embedded K_3 subgraphs. We then proceed with adding all K_4, all K_5 and finally all K_6 subgraphs to the CCE. Note that this enumeration is efficient in that it skips certain contributions, such as cycles over 4 variables because their contribution will be included when we add the corresponding K_4 subgraph.

'CCE (GBP)' enumerates in a similar fashion. GBP is run on a region graph formed using the "star" construction – outer regions equal to all cycles of length 3 passing through vertex 0. We then enumerate all complete graphs K_4, K_5 and K_6 containing vertex 0. The set of K_4 containing ver-

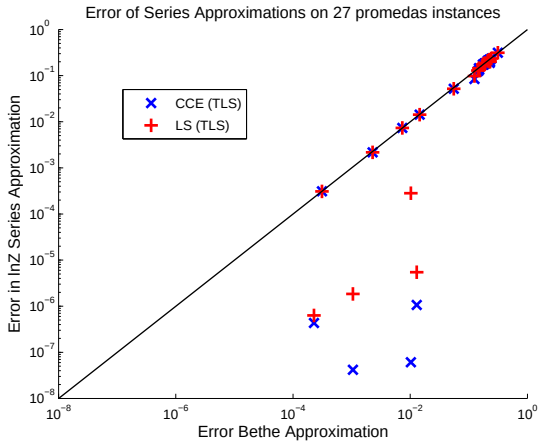


Figure 6: Scatter plot of $Error_Z$ for the LS and CC approximations versus error in the Bethe approximation. Points below the line are improvements upon IBP.

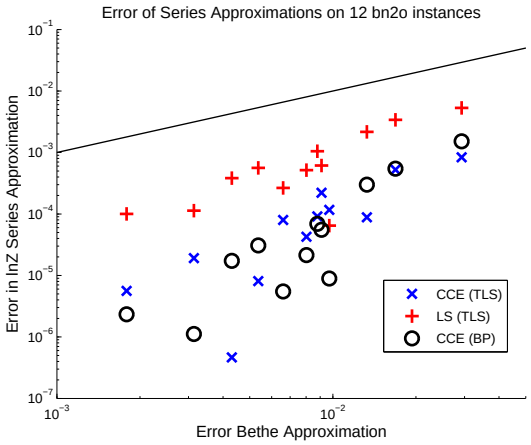


Figure 7: Scatter plot of $Error_Z$ for each series approximation versus error in the Bethe approximation. Points below the line are improvements upon IBP.

tex 0 is only a subset of all the K_4's embedded in the MN. However, enumerating embedded complete graphs in this fashion ensures that the collection of clusters remains balanced.

Figure 5 compares the four series approximations across a variety of interaction strengths. In these experiments the TLS algorithm was run with $S = 5000$ and $M = 10$. On complete graphs, the CC and LS approximations behave similarly when using terms from the TLS algorithm. The CCE as described above is much more accurate. This is because while the TLS algorithm enumerates > 10K loops, the loops contain at most 5 factors while the CCE up to K_6 covers 15 factors.

5.3 UAI-2008 Benchmarks

In addition to the synthetic instances, we evaluated each of the series approximations on instances from the 2008-UAI solver competition. We selected 12 bn2o instances and 27 promedas instances that were solvable by our Junction Tree implementation. The promedas instances have a very sparse graph structure and contain between 400-1000 factors. As a result, enumerating all clusters in Ω^l_{all} is costly and yields few non-zero terms even for small l. Thus, for the promedas instances we only compare the LS and CC approximations on the loops found by the TLS algorithm. Figure 6 shows the error of the CC and LS approximations versus the Bethe approximation. Points below the diagonal line indicate improvement over the Bethe approximation. The TLS algorithm was run on all 27 instances with $S = 100$ and $M = 10$. These results are consistent with [2] in that if the TLS algorithm finds most of the generalized loops in an instance, the error is reduced by several orders of magnitude, while if a small set of all generalized loops are found, the series approximations offer no improvement.

Unlike the promedas instances, the bn2o instances have densely connected graph structures and contain far fewer factors. These instance are thus better suited to enumeration along Ω_{all}. Figure 7 compares the error of the 4 series approximations to the error of the Bethe approximation. The TLS algorithm was run with $S = 10$K and $M = 10$, producing more than 10K generalized loops. 'CCE (BP)' was run on the set of clusters in Ω^4_{all}.

6 Conclusion

We have introduced a new cluster-cumulant expansion based on the fixed points of either BP or GBP. The expansion was inspired by the LS of [1] but has certain advantages over the latter. First, terms corresponding to disconnected clusters vanish. More generally, only tightly coupled groups of variables are expected to make significant contributions, which can be used to significantly cut down on the number clusters that need to be considered. Second, while the LS was only developed for binary variables, the CCE is defined on arbitrary alphabets. Third, the CCE has a natural extension to GBP on region graphs. Finally, the accuracy of the CCE expansion improves upon the LS.

The CCE represents a very natural extension of the Kikuchi approximation as it is based on the same type of expansion. But unlike GBP on a region graph, the CCE does not suffer from convergence issues and does not require storing beliefs during message passing. This makes it a useful "anytime" tool to improve results obtained form GBP. It also suggests new algorithms that move regions from the CCE to regions used in GBP. The question of which regions should be included in the GBP + CCE approximation and whether a region should be included in GBP or be handled by the CCE are left for future investigation.

Acknowledgements

MW acknowledges support by NSF Grants No. 0914783, 0928427, 1018433. AI acknowledges support by NSF IIS grant No. 1065618.

References

[1] M. Chertkov and V. Y. Chernyak. Loop series for discrete statistical models on graphs. *Journal of Statistical Mechanics: Theory and Experiment*, 2006.

[2] V. Gómez, J. M. Mooij, and H. J. Kappen. Truncating the loop series expansion for belief propagation. *Journal of Machine Learning Research*, 8:1987–2016, 2007.

[3] T. Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy. In *Neural Information Processing Systems*, volume 15, Vancouver, CA, 2003.

[4] N.C. Van Kampen. A simplified cluster expansion for the classical real gas. *Physica*, 27:783–792, 1961.

[5] R. Kubo. Generalized cumulant expansion method. *Journal of the Physical Society of Japan*, 17(7):1100–1120, 1962.

[6] T. Morita. Formal structure of the cluster variation method. *Progress of Theoretical Physics supplement*, 115:27–39, 1994.

[7] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, California, 1988.

[8] E. B. Sudderth, M. J. Wainwright, and A. S. Willsky. Loop series and bethe variational bounds in attractive graphical models. In *Advances in Neural Information Processing Systems 20*, pages 1425–1432, 2007.

[9] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation via pseudo-moment matching. In *AISTATS*, 2003.

[10] M. Welling. On the choice of regions for generalized belief propagation. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 585–592, 2004.

[11] J.S. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical report, MERL, 2002. Technical Report TR-2002-35.

Self-Confirming Price Prediction Strategies for Simultaneous One-Shot Auctions

Michael P. Wellman
Computer Science & Engineering
University of Michigan

Eric Sodomka
Computer Science
Brown University

Amy Greenwald
Computer Science
Brown University

Abstract

Bidding in simultaneous auctions is challenging because an agent's value for a good in one auction may depend on the uncertain outcome of other auctions: the so-called *exposure problem*. Given the gap in understanding of general simultaneous auction games, previous works have tackled this problem with heuristic strategies that employ probabilistic price predictions. We define a concept of *self-confirming prices*, and show that within an independent private value model, Bayes-Nash equilibrium can be fully characterized as a profile of optimal price-prediction strategies with self-confirming predictions. We exhibit practical procedures to compute approximately optimal bids given a probabilistic price prediction, and near self-confirming price predictions given a price-prediction strategy. An extensive empirical game-theoretic analysis demonstrates that self-confirming price-prediction strategies are effective in simultaneous auction games with both complementary and substitutable preference structures.

1 Introduction

One of the most attractive features of automated trading is the ability to monitor and participate in many markets simultaneously. Compared to human traders, software agents can take in data from multiple sources at very high throughput rates. In principle, software agents can also process massive quantities of information relevant to trading decisions in short time spans. In practice, however, dealing with multiple markets poses one of the greatest strategic challenges for automated trading. When markets interact, a strategy for bidding in one market must consider the implications of and ramifications for what happens in others.

Markets are *interdependent* when an agent's preference for the outcome in one depends on results of others. For instance, when value for one good is increased by obtaining another, the goods are *complements*. Dealing with multiple markets under complementary preferences presents an agent with the classic *exposure problem*: before it can obtain a valuable bundle, the agent must risk getting stuck with a strict subset of the goods, which it may not have wanted at the prevailing prices. Exposure is a potential issue for *substitute* goods as well, as the agent risks obtaining goods it does not want given that it obtains others.

The pitfall of exposure is a primary motivation for combinatorial auctions [Cramton et al., 2005], where the mechanism takes responsibility for allocating goods respecting agents' expressed interdependencies. Combinatorial auctions are often infeasible, however, due to nonexistence of an entity with the authority and capability to coordinate markets of independent origin. Consequently, interdependent markets are inevitable. Nonetheless, there is at present very little fundamental understanding of agent bidding strategies for these markets. Specifically, how should an agent's bidding strategy address the exposure problem?

We address this question in what is arguably the most basic form of interdependent markets: simultaneous one-shot sealed-bid (SimOSSB) auctions. Despite the simplicity of this mechanism and the practical importance of the exposure issue, there is little available guidance in the auction theory literature on the strategic problem of how to bid in SimOSSB auctions. We aim to fill this gap by providing computationally feasible methods for constructing bidding strategies for the SimOSSB-auction environment, which we justify with both theory and evidence from simulation-based analysis. Specifically, we (i) characterize Bayes-Nash equilibria of SimOSSB auctions as best responses to price predictions (§3); (ii) provide bounds on approximate Bayes-Nash equilibria in terms of the accuracy of price predictions and the degree of optimality of responses (§4); (iii) introduce methods to construct bidding strategies that respect the equilibrium form (§5,6); and (iv) demonstrate through a comprehensive empirical game-theoretic analysis the efficacy of these strategies compared to a wide variety of heuristics from the literature (§7).

2 Previous Work

Theoretical results about general simultaneous auction games are few and far between. The leading auction theory textbook [Krishna, 2010] treats sequential but not simultaneous auctions, and most of the literature that addresses simultaneity does so only in the context of ascending [Peters and Severinov, 2006] or multi-unit auctions.

In the first work to derive an equilibrium of a simultaneous-auction game, Engelbrecht-Wiggans and Weber [1979] tackle an example with perfect substitutes, where each agent is restricted to bid on at most two items. Their analysis was performed in the large-limit of auctions and agents, and exhibited a mixed equilibrium where the agents diversify their bids even though the items are indistinguishable. Krishna and Rosenthal [1996] studied a second-price setup with two categories of bidders: local bidders who have value for a single item, and global bidders who have superadditive values for multiple items. The authors characterize an equilibrium that is symmetric with respect to the global bidders, and show, somewhat surprisingly, that an increase in the number of bidders often leads to less aggressive bidding. Rosenthal and Wang [1996] tackled a first-price setup, assuming synergies and common values. Szentes and Rosenthal [2003] studied two-bidder auctions with three identical objects and complete information.

Recently, Rabinovich et al. [2011] generalized fictitious play to incomplete information games with finite actions and applied their technique to a class of simultaneous second-price auctions. They computed approximate equilibria in environments with utilities expressible as linear functions over a one-dimensional type space.

Complementing these theoretical treatments, researchers have designed trading strategies applicable to simultaneous auctions, which address the exposure problem through heuristic means. In the Trading Agent Competition (TAC) Travel game [Wellman et al., 2007], agents face an exposure problem for hotels—they must obtain a room each night for the client or the whole trip is infeasible. Experience from TAC and other domains has demonstrated the importance of *price prediction* for bidding in interdependent markets [Wellman et al., 2004]. Given probabilistic predictions of prices across markets, agents can manage exposure risk, choosing bids that trade off the profits and losses of the possible bundles of goods they stand to win.

Greenwald and Boyan [2004] framed the problem of bidding across interdependent markets given probabilistic price predictions. Follow-on work [Greenwald et al., 2009, Wellman et al., 2007] formalized this bidding problem in decision-theoretic terms and established properties of optimal bidding assuming that bids do not affect other-agent behaviors. Further experimental comparison was performed by Greenwald et al. [2010]. These works introduced a taxonomy of heuristic bidding strategies [Wellman, 2011], which we employ here.

Self-confirming price-prediction (SCPP) bidding strategies were first explored in the context of simultaneous ascending auctions (SimAAs) [Cramton, 2005]. For the SimAA environment, SCPP strategies were found to be highly effective at tackling the exposure problem [Wellman et al., 2008]. To our knowledge, no other general price-prediction methods have been proposed for SimOSSB auctions, other than learning from historical observations.

3 Price-Prediction Strategies and Equilibrium

We consider a market with m goods, $\mathcal{X} = \{1, \ldots, m\}$, and n agents. Agent i's value for a bundle $X \in 2^{\mathcal{X}}$ is given by $v_i(X)$, where $v_i(X) \in [0, \bar{V}]$. We assume free disposal: if $X \subseteq X'$, then $v_i(X) \leq v_i(X')$. The m goods are allocated to the agents via SimOSSB auctions, one per good. That the mechanism is simultaneous means that each agent i submits a bid vector $\boldsymbol{b}_i = (b_i^1, \ldots, b_i^m) \in \mathbb{R}_+^m$ before a specified closing hour. That the auctions are one-shot means that all auctions compute and report their results upon the closing hour. That the bids are sealed means that agents have no information about the bids of other auction participants until the outcome is revealed.

The *second-price sealed-bid* (SPSB) auction is an OSSB auction in which the winning bidder pays the second-highest bid rather than its own (highest) bid. The environments studied in our empirical game-theoretic analysis below employ the SPSB mechanism. For simplicity of description we focus on SPSB throughout, although our theoretical results hold as well for first-price sealed-bid auctions, or indeed any auction mechanism where the outcome to agent i (whether it gets the good and the price it pays) is a function of i's own bid and the highest other-agent bid.

Our investigation employs the familiar *independent private values* (IPV) model (see, for example, [Krishna, 2010]), where each agent i's values are drawn independently from a probability distribution that is common knowledge. Under IPV, it is a dominant strategy for an agent to bid its true value in a single SPSB auction. This result does not generalize to simultaneous SPSB auctions, however, unless the agent's value over bundles happens to be additive. When agents' values for different goods interdepend (e.g., through complementarity or substitutability), bidding truthfully in simultaneous auctions is not even an option, as the value for an individual good is not well-defined.

To deal effectively with interdependent markets, an agent's bid in each auction must reflect its beliefs about the outcomes of others. We consider beliefs in the form of *predictions* about the prices at which the agent might obtain goods in the respective auctions. Bidding strategies that are

explicitly cast as functions of some input price prediction are termed *price-prediction (PP) strategies*.

We denote by $p_j \in \mathbb{R}_+$ a *price* for good j. The vector $\boldsymbol{p} = \langle p_1, \ldots, p_m \rangle$ associates a price with each good. We represent price predictions as probability distributions over the joint price space. We use the symbol Π for such predictions in the form of cumulative probability distributions,

$$\Pi_{\boldsymbol{p}}(\boldsymbol{q}) = \Pr(\boldsymbol{p} \leq \boldsymbol{q}), \tag{1}$$

where $\boldsymbol{p} \leq \boldsymbol{q}$ holds iff $p_j \leq q_j$ for all j. We generally omit the subscript $\boldsymbol{p}$ as understood.

We have previously argued [Wellman et al., 2007] that price prediction is a key element of agent architecture for complex trading environments. Here, we support a stronger claim for the case of SimOSSB auctions: given IPV, *PP strategies are necessary and sufficient for optimal bidding*.

Let $w(\boldsymbol{b}, \boldsymbol{q}) = \{j \mid b^j > q_j\}$ denote the set of goods an agent would win by bidding $\boldsymbol{b}$ when the highest other-agent bids are $\boldsymbol{q}$.[1] Agent i's utility for a bid given others' bids can thus be written as

$$u_i(\boldsymbol{b}, \boldsymbol{q}) = v_i(w(\boldsymbol{b}, \boldsymbol{q})) - \sum_{j \in w(\boldsymbol{b}, \boldsymbol{q})} q_j. \tag{2}$$

Definition 1 (Optimal PP Bidders)**.** An *optimal PP bidding strategy* $s^*(\Pi)$ submits bids that maximize expected utility given a price prediction Π,

$$s^*(\Pi) \in \arg\max_{\boldsymbol{b}} \mathbb{E}_{\boldsymbol{q} \sim \Pi}[u_i(\boldsymbol{b}, \boldsymbol{q})]. \tag{3}$$

In games of incomplete information, agent i's strategy produces actions (here, bids) as a function of i's type. Under IPV, knowing other agents' values tells agent i nothing about its own value. The distribution of outcomes given i's bid $\boldsymbol{b}_i$ depends only on $\boldsymbol{b}_i$ and the marginal distribution of other-agent bids. Agent i's expected utility is thus conditionally independent of other-agent valuations given their bids [Wellman et al., 2011]. In particular, i's best response to a profile of other-agent strategies depends only on the distribution of their bids [Rabinovich et al., 2011].

Let b^{k*}_{-i} denote the highest bid submitted for good k by an agent other than i. Since agent i's utility depends only on what it wins and what it pays, the distribution of *highest* other-agent bids (i.e., the distribution of b^{k*}_{-i}) is a sufficient statistic for the other-agent bid distributions. This distribution can be expressed in the form of a price prediction (1).

Therefore, a best response to other-agent bidding strategies takes the form of an optimal PP bidding strategy (3), where the input PP is the distribution of highest other-agent bids induced by those other-agent strategies. Since a Bayes-Nash equilibrium (BNE) is a profile of mutual best-response strategies, any profile of optimal PP strategies, where the PP for each equals the distribution of highest bids induced by the other agents' optimal PP strategies, constitutes a BNE. Moreover, any BNE can be characterized as a profile of optimal PP strategies, or mixtures thereof.

Theorem 1. *Suppose a SimOSSB auction game, with independent private values, where the outcome of each auction to agent i depends only on i's bid and the highest other-agent bid in that auction. Then the strategy profile $\boldsymbol{s} = (s_i, s_{-i})$ is a Bayes-Nash equilibrium if and only if, for all i, s_i is equivalent to an optimal PP bidding strategy with input $\Pi_i(\boldsymbol{q}) = \Pr((b^{1*}_{-i}, \ldots, b^{m*}_{-i}) \leq \boldsymbol{q} \mid s_{-i})$, or equivalent to a mixture of such optimal PP strategies.*

Price predictions that support strategic equilibrium are themselves in a form of equilibrium. The bidding strategies employ price predictions that are actually borne out as correct (i.e., the distributions generated by the strategies are as predicted) assuming that everyone follows the given strategies. This situation can be viewed as a form of *rationalizable conjectural equilibrium* (RCE) [Rubinstein and Wolinsky, 1994], where each agent's conjecture is about the distribution of highest other-agent bids. In this instance, the RCE is also a BNE, since the conjecture provides sufficient information to determine a best response.

An auction game with IPV is *symmetric* if all agents have the same probability distribution over valuations. In earlier work on simultaneous ascending auctions [Wellman et al., 2008], we considered the symmetric IPV case and referred to price predictions in this kind of equilibrium relationship as *self-confirming*. Let Γ be an instance of a symmetric IPV SimOSSB auction game, and s a bidding strategy that employs price predictions (whether optimally or not).

Definition 2 (Self-Confirming Price Prediction (SCPP))**.** The prediction Π is *self-confirming* for PP strategy s in Γ iff Π is equal to the distribution of the highest other-agent prices $(b^{1*}_{-i}, \ldots, b^{m*}_{-i})$ when all agents play $s(\Pi)$.

The following theorem specializes Theorem 1 for the symmetric case, employing the language of SCPPs.

Theorem 2. *In symmetric IPV SimOSSB auctions, a symmetric pure BNE comprises optimal PP bidders employing self-confirming price predictions. Hence, existence of pure symmetric BNE in such games entails existence of self-confirming price predictions.*

In summary, for SimOSSB auctions under IPV, we can restrict attention to optimal PP strategies employing price predictions that are in equilibrium with one another (which for the symmetric case, means SCPPs). In the remainder of this paper, we demonstrate that PP strategies are amenable to effective approximation, are a convenient abstraction on

[1] In the case of ties, the winner is chosen uniformly from among the high bidders. Our analysis ignores ties, which are rare given our setup of real-valued bids and rich valuation functions. Our theoretical results require that the highest other-agent bid is a sufficient statistic for outcome, which is true as long as one's bid does not tie for best with *two or more* other bidders.

which to design and implement trading strategies, and exhibit a high degree of robustness across environments.

4 Approximate Price Prediction

We have shown that an optimal PP strategy is a best response to the strategies of other agents if price predictions Π exactly reflect the other-agent bid distributions. Since it is unrealistic to expect perfect price prediction, we examine the consequences of employing PP strategies that use inaccurate price predictions Π'.

We quantify the inaccuracy of Π' in two ways. The first is a multivariate form of the *Kolmogorov-Smirnov (KS) statistic*: $KS(\Pi, \Pi') \equiv \sup_{\boldsymbol{q}} |\Pi(\boldsymbol{q}) - \Pi'(\boldsymbol{q})|$. Second, we define the *bundle probability distance*, $BP(\Pi, \Pi', \boldsymbol{b})$, with respect to a bid $\boldsymbol{b}$: $\sum_{X \subseteq \mathcal{X}} |\Pr_{\boldsymbol{q} \sim \Pi}(w(\boldsymbol{b}, \boldsymbol{q}) = X) - \Pr_{\boldsymbol{q} \sim \Pi'}(w(\boldsymbol{b}, \boldsymbol{q}) = X)|/2$.

We first bound the difference between perceived and actual expected utility when incorrectly using Π' instead of Π. Let us denote the payment for $\boldsymbol{b}$ given highest other-agent bids $\boldsymbol{q}$ by $\psi(\boldsymbol{b}, \boldsymbol{q})$, so that for SPSB we have $\psi(\boldsymbol{b}, \boldsymbol{q}) = \sum_{j \in w(\boldsymbol{b}, \boldsymbol{q})} q_j$. Overall expected utility is the difference between expected value of winnings and payment:

$$\mathbb{E}_{\boldsymbol{q}}[u_i(\boldsymbol{b}, \boldsymbol{q})] = \mathbb{E}_{\boldsymbol{q}}[v_i(w(\boldsymbol{b}, \boldsymbol{q}))] - \mathbb{E}_{\boldsymbol{q}}[\psi(\boldsymbol{b}, \boldsymbol{q})]. \quad (4)$$

To bound the difference between perceived and actual expected utility, we separately consider winnings and payment. The following bounds the amount a bidder can underestimate its expected payment by using Π'.

Lemma 3. *Let $\delta_{KS} = KS(\Pi, \Pi')$ and $\|\boldsymbol{b}\|_1 \equiv \sum_{j=1}^{m} b^j$. Then for all $\boldsymbol{b}$,*

$$\mathbb{E}_{\boldsymbol{q} \sim \Pi}[\psi(\boldsymbol{b}, \boldsymbol{q})] \leq \mathbb{E}_{\boldsymbol{q} \sim \Pi'}[\psi(\boldsymbol{b}, \boldsymbol{q})] + 2\delta_{KS}(\|\boldsymbol{b}\|_1), \quad (5)$$

Proof. See Appendix A.1 in the online supplement. □

We similarly bound the amount a bidder can overestimate its expected value of winnings by using Π'. A variant distribution Π can degrade expected value of winnings only by decreasing the probability of winning valuable bundles. By constraining BP distance, we can ensure, for any set of bundles, that the total probability of winning a bundle from that set at $\boldsymbol{b}$ decreases by at most δ_{BP}. This means that the expected value of winnings can suffer by at most $\delta_{BP}\bar{V}$.

Lemma 4. *Let $\delta_{BP} = BP(\Pi, \Pi', \boldsymbol{b})$. Then*

$$\mathbb{E}_{\boldsymbol{q} \sim \Pi}[v_i(w(\boldsymbol{b}, \boldsymbol{q}))] \geq \mathbb{E}_{\boldsymbol{q} \sim \Pi'}[v_i(w(\boldsymbol{b}, \boldsymbol{q}))] - \delta_{BP}\bar{V}. \quad (6)$$

Combining the lemmas, we have the following bound.

Theorem 5. *Let $\delta_{KS} = KS(\Pi, \Pi')$ and $\delta_{BP} = BP(\Pi, \Pi', \boldsymbol{b})$. Then for all i,*

$$\mathbb{E}_{\boldsymbol{q} \sim \Pi}[u_i(\boldsymbol{b}, \boldsymbol{q})] \geq \mathbb{E}_{\boldsymbol{q} \sim \Pi'}[u_i(\boldsymbol{b}, \boldsymbol{q})] - \delta_{BP}\bar{V} - 2\delta_{KS}\|\boldsymbol{b}\|_1.$$

We can use these bounds to limit how far an optimal PP strategy with inaccurate price predictions Π' can be from equilibrium. Let us denote by $\bar{b}$ the maximum payment, that is, the L_1-norm of the greatest possible bid vector. The value of $\bar{b}$ is bounded above by $m\bar{V}$ for any rational bidding strategy under any valuation distribution, but typically it will be far less than that.

Theorem 6. *Suppose that for all agents i, strategy $\hat{s}_i$ is a best response to other-agent highest-bid distribution $\hat{\Pi}_{-i}$, and that $\Pi_{\hat{s}_{-i}}$ is the other-agent bid distribution actually induced by $\hat{s}_{-i}$. If for all i, $KS(\hat{\Pi}_{-i}, \Pi_{\hat{s}_{-i}}) \leq \delta_{KS}$, and for all $\boldsymbol{b}$, $BP(\hat{\Pi}_{-i}, \Pi_{\hat{s}_{-i}}, \boldsymbol{b}) \leq \delta_{BP}$, then $\hat{\boldsymbol{s}}$ constitutes an ϵ-Bayes-Nash equilibrium, for $\epsilon = 2\delta_{BP}\bar{V} + 4\delta_{KS}\bar{b}$.*

Proof. See online Appendix A.2. □

5 Heuristic PP Bidding Strategies

Having shown that optimal PP bidding strategies are theoretically ideal in that they comprise a BNE, we turn our attention to practical PP bidding strategies. Building on prior work [Wellman et al., 2007, Chapter 5], we explore a broad range of heuristic bidding strategies. The most salient of these are described here.

5.1 Marginal Values and Optimal Bundles

Interdependence dictates that the value of any individual good must be assessed relative to a bundle of goods. This idea is captured by the notion of *marginal value*.

Definition 3 (Marginal Value)**.** Agent i's marginal value, $\mu_i(x, X)$, for good x with respect to a fixed bundle of other goods X is given by: $\mu_i(x, X) \equiv v_i(X \cup \{x\}) - v_i(X)$.

Given a fixed vector of prices, $\boldsymbol{p} = \langle p_1, \ldots, p_m \rangle$, let $\sigma_i(X, \boldsymbol{p})$ denote agent i's surplus from obtaining the set of goods X at those prices:

$$\sigma_i(X, \boldsymbol{p}) \equiv v_i(X) - \sum_{j | x_j \in X} p_j. \quad (7)$$

Definition 4 (Acquisition [Boyan and Greenwald, 2001])**.** Given price vector $\boldsymbol{p}$, the *acquisition problem* selects an optimal bundle of goods to acquire: $X^* = \text{ACQ}_i(\boldsymbol{p}) \equiv \arg\max_{X \subseteq \mathcal{X}} \sigma_i(X, \boldsymbol{p})$.

Faced with perfect point price predictions, an optimal bidding strategy would be to compute $X^* = \text{ACQ}_i(\boldsymbol{p})$ and then to buy precisely those goods in X^*. By definition, this strategy yields the optimal surplus at these prices: $\sigma_i^*(\boldsymbol{p}) \equiv \sigma_i(\text{ACQ}_i(\boldsymbol{p}), \boldsymbol{p})$.

To assess goods with respect to (typically imperfect) point price predictions, we extend the concept of marginal value. Let $\boldsymbol{p}[p_j \leftarrow q]$ be a version of the price vector $\boldsymbol{p}$ with

the jth element revised as indicated: $\boldsymbol{p}[p_j \leftarrow q] = \langle p_1, \ldots, p_{j-1}, q, p_{j+1}, \ldots, p_m \rangle$.

Definition 5 (Marginal Value at Prices)**.** Agent i's *marginal value* $\mu_i(x_j, \boldsymbol{p})$ for good x_j at prices $\boldsymbol{p}$ is given by: $\mu_i(x_j, \boldsymbol{p}) \equiv \sigma_i^*(\boldsymbol{p}[p_j \leftarrow 0]) - \sigma_i^*(\boldsymbol{p}[p_j \leftarrow \infty])$.

Here, $\sigma_i^*(\boldsymbol{p}[p_j \leftarrow 0])$ represents the optimal surplus at the given prices, assuming good x_j is free. Similarly, $\sigma_i^*(\boldsymbol{p}[p_j \leftarrow \infty])$ represents the optimal surplus at the given prices, if x_j were unavailable. The difference is precisely the marginal value of good x_j with respect to the prices of other goods. Note that Definition 5 generalizes Definition 3, under the interpretation that goods in X have zero price, and all other goods have infinite price.

StraightMV The heuristic strategy StraightMV employs this concept directly, and simply bids marginal value for all goods: $\mathsf{StraightMV}_j = \mu(x_j, \boldsymbol{p})$.

5.2 Bidding with Price Distributions

A point price estimate fails to convey the uncertainty inherent in future prices. Probability distributions over prices provide a more general representation, expressing degrees of belief over the possible prices that might obtain.

The *expected value method* [Birge and Louveaux, 1997] approximates a stochastic optimization by collapsing probability distributions into point estimates through expectation. Let $\hat{\boldsymbol{p}}_\Pi = \langle \hat{p}_1, \ldots, \hat{p}_m \rangle$, where $\hat{p}_j = \mathbb{E}_{\boldsymbol{p} \sim \Pi}[p_j]$ is the expectation of p_j under given price prediction Π.

Any bidding strategy defined for point price predictions can be adapted to take as input distribution price predictions through this expected value method, simply by using $\hat{\boldsymbol{p}}_\Pi$ for the point price prediction. For example, we define StraightMU as the strategy that takes as input a distribution prediction, and bids as StraightMV at the mean prices. "MU" stands for "marginal utility", but its usage here is simply to distinguish the name from its corresponding point strategy ending with "MV".

We implemented two approaches to calculating $\hat{\boldsymbol{p}}_\Pi$. The first samples from Π, and the second computes the exact expectation of a piecewise approximation of Π. For the sampling method, accuracy depends on the number of samples k drawn; thus we indicate a version of the strategy by appending "k" to the name. For example, StraightMU8 takes the mean of eight samples from Π, and employs StraightMV with that average price vector as input.

AverageMU Whereas StraightMU bids the marginal value of the expected price, the PP strategy AverageMU bids the expected marginal value: $\mathsf{AverageMU}_j = \mathbb{E}_{\boldsymbol{p} \sim \Pi}[\mu(x_j, \boldsymbol{p})]$. Our implementation samples from the price distribution, calculates marginal values for each sample, and averages the results.

5.3 Explicit Optimization

Finally, we consider strategies that explicitly attempt to optimize bids given a distribution price prediction, as do the optimal PP bidders introduced above (Definition 1).

BidEval One heuristic optimization approach is to generate candidate bid vectors, and evaluate them according to the price prediction. The BidEval strategy uses other bidding heuristics to propose candidates, and estimates each candidate's performance by computing its expected utility given the price prediction. It then selects the candidate bid vector with the greatest expected utility. There are many variations of BidEval, defined by:

- *the method used to generate candidates.* When this method is a named bidding strategy, we indicate this in parentheses; for instance, BidEval(SMU8) generates candidates using StraightMU8. Strategy BidEvalMix employs a mix of previously described bidding strategies to generate candidates.
- *number of candidates generated.* Methods based on sampling naturally produce a diverse set of candidates. For example, each invocation of StraightMU8 employs a new draw of eight samples from Π to estimate $\hat{\boldsymbol{p}}_\Pi$, thus we generally obtain different bids.
- whether the candidates are evaluated by exact computation on a piecewise version of Π, or by sampling. If by sampling, then how many samples are used.

LocalBid The LocalBid strategy (see Alg. 1) employs a local search method in pursuit of optimal bids. Starting with an initial bid vector proposed by another heuristic strategy, LocalBid makes incremental improvements to that bid vector for a configurable number of iterations. Those incremental improvements are made good-by-good, treating all other goods' bids as fixed. Assuming bids for all goods except j are fixed, the agent effectively faces a single auction for good j, with the winnings for other goods determined probabilistically. For a single SPSB auction, it is a dominant strategy to bid one's expected marginal value for good j, which is given by $\mathbb{E}_{\boldsymbol{p} \sim \Pi}[v(w(\boldsymbol{b}, \boldsymbol{p}) \cup \{j\}) - v(w(\boldsymbol{b}, \boldsymbol{p}) \setminus \{j\})]$, under these circumstances.

Algorithm 1 LocalBid

Input: prediction Π, heuristic strategy s, #iterations K
Output: bid vector $\boldsymbol{b}$

```
Initialize b ← s(Π)
for k = 1 to K do
  for j = 1 to m do
    b^j ← E_{p∼Π}[v(w(b, p) ∪ {j}) − v(w(b, p) \ {j})]
return b
```

LocalBid is an iterative improvement algorithm: the expected utility of $\boldsymbol{b}$ is nondecreasing with each update. Further, if LocalBid converges, it returns a bid vector that is

consistent (unlike AverageMU), in the sense that each element of the vector is the average marginal value for its corresponding good given the rest of the bid.

In an empirical comparison of bid optimization algorithms, we found that the LocalBid method was highly effective. In one representative environment, LocalBid achieved 98.8% optimality (see Figure 1), whereas a version of BidEval produced bids with expected value 94.5% of optimal. In that same environment, the bids generated by AverageMU64 were 66.1% as profitable as the optimal bid.

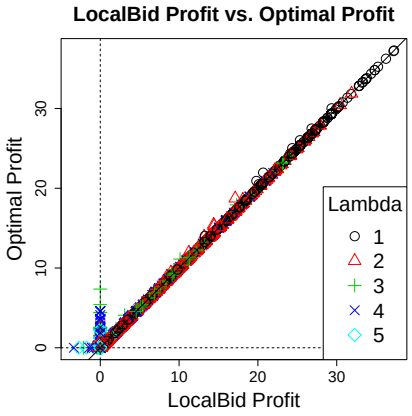


Figure 1: Expected profit for LocalBid versus optimal, for 5000 sample valuations in SimSPSB environment U[5,5]. Points above the diagonal are suboptimal instances.

6 Self-Confirming Price Predictions

Now that we have a suite of strategies that employ price predictions, we turn to the question of how to generate such predictions. We propose here to employ *self-confirming price predictions* (SCPPs) (Definition 2), originally introduced and evaluated in the context of simultaneous ascending auctions [Osepayshvili et al., 2005]. A high-level description of our iterative procedure for computing self-confirming price predictions is shown in Alg. 2.

Algorithm 2 Self-Confirming Price Search

Input: PP strategy PP, parameters F^0, L, G, κ_t, τ
Output: price prediction F
  Initialize $F \leftarrow F^0$
  for $t = 1$ to L **do**
    $F' \leftarrow$ outcome of G runs with all playing $\mathsf{PP}(F)$
    if $KS_{marg}(F, F') < \tau$ **then**
      return F
    $F \leftarrow \kappa_t F' + (1 - \kappa_t)F$
  return F

We extend the flexibility of the existing SCPP-derivation procedure by considering two interpretations of the "outcome" prices resulting from each instance of the game. In one, the resulting price is the actual transaction price of the good (price), as in the earlier SimAA study. In the second, we take the highest other-agent bid (HB) as the result.

To measure the difference between distributions at successive iterations, we adopt the Kolmogorov-Smirnov statistic. Since we maintain our prediction in terms of marginal distributions, our comparison takes the maximum of the KS statistic separately for each good: $KS_{marg}(F, F') \equiv \max_j KS(F_j, F'_j)$.

Our initial prediction F^0 considers all integer prices in the feasible range equally likely. At each iteration, we run G instances of game Γ, with all agents playing the distribution-prediction strategy $\mathsf{PP}(F)$. We tally the prices resulting from each instance, and update the price prediction as a weighted average of this tally F' and the previous prediction F. When $KS_{marg}(F, F')$ falls below threshold, we halt and return F. Or, if this distance never falls below threshold, then the procedure terminates and returns the result after L iterations.

Although not guaranteed to converge, we found that for the heuristics of the previous section the iterative procedure generally produced approximately marginally self-confirming distribution predictions. For example, in one environment with a range of strategies we ran the procedure with $G = 10^6$ and $L = 100$, achieving an accuracy $KS_{marg} < 0.01$ in all cases. Due to space constraints, we relegate a full description and evaluation of this procedure to the extended version of this paper.

7 Empirical Game-Theoretic Analysis

The heuristic strategies introduced in §5 represent plausible but not generally optimal approaches to bidding in simultaneous auctions. Even the strategies based on explicit optimization (§5.3) fall short of ideal due to inaccuracy in price prediction and non-exhaustive search of bid candidates. To evaluate the performance of these strategies, we conducted an extensive computational study, simulating thousands of strategy profiles—millions of times each—in five different simultaneous SPSB environments. Analysis of the game model induced from simulation data provides evidence for the efficacy and robustness of approximately optimal PP strategies across these environments.

7.1 Approach

The methodology of applying game-theoretic reasoning to simulation-induced game models is called *empirical game-theoretic analysis* (EGTA) [Wellman, 2006]. In EGTA, we simulate profiles of an enumerated strategy set playing a game, and estimate a normal-form game from the observed payoffs. The result is a simulation-induced game model, called the *empirical game*. By applying standard game-theoretic solution concepts to the empirical game,

we can draw conclusions about the strategic properties of the strategies and profiles evaluated. Although the strategy space in the empirical game is necessarily a severely restricted subset of the original, by including a broad set of strategies representing leading ideas from the literature, we can produce relevant evidence bearing on the relative quality of heuristic strategies in the simulated environments.

Our EGTA study of SimSPSB followed these steps.

1. Define an environment: numbers of goods and agents, and valuation distributions.
2. Specify a set of heuristic strategies. For PP strategies, this includes deriving self-confirming distribution price predictions to be input to these strategies, based on the environment defined in Step 1. The full set of strategies included in our EGTA study is described in online Appendix B.
3. Simulate select profiles among these strategies, sampling from the valuation distributions for each simulation instance (at least one million per profile, most profiles two million or more). Calculate mean payoffs for each strategy in each profile.
4. Analyze the empirical game defined by these mean payoffs to identify Nash equilibria, dominance relationships, regret values, and other analytic constructs.

In actuality, Steps 2–4 were applied in an iterative and interleaved manner, with intermediate analysis results informing the selection of strategies to explore and profiles to sample. The exploration and sample selection were guided manually, generally driven by the objective of confirming or refuting equilibrium candidates among the profiles already evaluated. The process for each environment was terminated when all of the following conditions were met: (1) a broadly representative set of heuristic strategies were covered, (2) all symmetric mixed profiles evaluated were either confirmed or refuted as equilibria, and (3) all strategies showing relative success in at least one environment were evaluated against the equilibria in all other environments. Overall, the analysis commanded some tens of CPU-years over a roughly six-month period.

7.2 Environments

We evaluated five simultaneous SPSB environments, involving 3–8 agents bidding on 5 or 6 goods. The environments span two qualitatively different valuation distributions, from highly complementary to highly substitutable. Both of these assume IPV and symmetry, so that each agent receives a private valuation drawn independently from the same distribution.

7.2.1 Scheduling Valuations

The first valuation distribution we employ in this study is based on a model of *market-based scheduling* [Reeves et al., 2005]. Goods represent time slots of availability for some resource: for example, a machine, a meeting room, a vehicle, or a skilled laborer. Agents have tasks, which require this resource for some duration of time to complete.

Specifically, the goods $\mathcal{X} = \{x_1, \ldots, x_m\}$ comprise a set of m time slots available to be scheduled. Agent i's task requires λ_i time slots to accomplish, and the agent values a set of time slots according to when they enable completion of the task. If agent i acquires λ_i time slots by time t, it obtains value V_i^t. Completion value with respect to time is a nonincreasing function: for all i, if $t < t'$ then $V_i^t \geq V_i^{t'}$. If it fails to obtain λ_i slots, the agent accrues no value ($V_i^\infty = 0$). Let $X \subseteq \mathcal{X}$ denote a set of slots. The expression $|\{x_j \in X \mid j \leq t\}|$ represents the number of these that are for time t or earlier. The overall valuation function for agent i is $v_i(X) = V_i^{T(X,\lambda_i)}$, where

$$T(X, \lambda) = \min\left(\{t \text{ s.t. } |\{x_j \in X \mid j \leq t\}| \geq \lambda\} \cup \{\infty\}\right)$$

is the earliest time by which X contains at least λ slots.

For each agent, a task length λ_i is drawn uniformly over the integers $\{1, \ldots, m\}$. Values associated with task completion times are drawn uniformly over $\{1, \ldots, 50\}$, then pruned to impose monotonicity [Reeves et al., 2005]. The valuations induced by this scheduling scenario exhibit strong complementarity among goods. When $\lambda > 1$, the agent gets no value at all for goods in a bundle of fewer than λ. On the other hand, there is some degree of substitutability across goods when there may be multiple ways of acquiring a bundle of the required size.

We denote environments using this valuation by $U[m, n]$, with m and n the numbers of goods and agents, and "U" indicating the uniform distribution over task lengths.

7.2.2 Homogeneous-Good Valuations

The second valuation distribution expresses the polar opposite of complementarity: goods are perfect substitutes, in that agents cannot distinguish one from another. Agents' marginal values for units of this good are weakly decreasing. Specifically, valuation is a function of the number of goods obtained, constructed as follows. Agent i's value for obtaining exactly one good, $v_i(\{1\})$, is drawn uniformly over $\{0, \ldots, 127\}$. Its value for obtaining two, $v_i(\{1, 2\})$, is then drawn from $\{v_i(\{1\}), v_i(\{1\}) + 1, \ldots, 2v_i(\{1\})\}$. In other words, its marginal value for the second good is uniform over $\{0, \ldots, v_i(\{1\})\}$. Subsequent marginal values are similarly constrained not to increase. Its marginal value for the kth good is uniform over $\{0, \ldots, v_i(\{1, \ldots, k-1\}) - v_i(\{1, \ldots, k-2\})\}$.

We denote environments using this valuation by $H[m, n]$,

with "H" an indicator for homogeneity.

7.3 Regret

We evaluate the stability of a strategy profile by *regret*, the maximal gain a player could achieve by deviating from the profile. Formally, let $\Gamma = \{n, S, u(\cdot)\}$ be a symmetric normal-form game with n players, strategy space S (the same for each player, since the game is symmetric), and payoff function $u : S \times S^{n-1} \to \mathbb{R}$. The expression $u(s_i, s_{-i})$ represents the payoff to playing strategy s_i in a profile where the other players play strategies $s_{-i} \in S^{n-1}$.

Definition 6 (Regret)**.** The *regret* $\epsilon(\boldsymbol{s})$ of a strategy profile $\boldsymbol{s} = (s_1, \ldots, s_n)$ is given by

$$\epsilon(\boldsymbol{s}) = \max_i \max_{s'_i \in S} \left(u(s'_i, s_{-i}) - u(s_i, s_{-i})\right).$$

A Nash equilibrium profile has zero regret, and more generally regret provides a measure of approximation to Nash equilibrium. Using this regret definition, profile $\boldsymbol{s}$ is an $\epsilon(\boldsymbol{s})$-Nash equilibrium.

Regret is a property of profiles. Evaluation of a particular strategy is inherently relative to a context of strategies played by other agents. Jordan et al. [2007] proposed ranking strategies according to their performance when other agents are playing an equilibrium.

Definition 7 (NE regret [Jordan, 2010])**.** Let $\boldsymbol{s}^{NE}$ be a Nash equilibrium of game Γ. The regret of strategy $s_i \in S$ relative to $\boldsymbol{s}^{NE}$, $u(s_i^{NE}, s_{-i}^{NE}) - u(s_i, s_{-i}^{NE})$, is an *NE regret* of s_i in Γ.

NE regret represents the loss experienced by an agent for deviating to a specified strategy from a Nash equilibrium of a game. The rationale for this measure comes from the judgment that all else equal, Nash equilibria provide a compelling strategic context for evaluating a given strategy. For games with multiple NE, a given strategy may have multiple NE-regret values.

7.4 Results

Table 1 summarizes the extent of simulation coverage of the five SPSB environments investigated.[2] The empirical games comprise 600–14,000 profiles, over 29–34 strategies. Evaluated profiles constitute a small fraction (as little as 0.03%) of the entire profile space over these strategies. Nevertheless, these are sufficient to confirm symmetric Nash equilibria for each game. Whereas it is impossible to rule out additional equilibria without exhaustive evaluation, we have either confirmed or refuted every evaluated symmetric mixed profile (i.e., every subset of strategies for which all profiles are evaluated). Since this includes every strategy in conjunction with those most effective in other contexts, we doubt that there are any other small-support symmetric equilibria, and expect few if any alternative symmetric equilibria among the explored strategies.

Table 1: Strategies and profiles simulated for the environments addressed in our EGTA study.

Environment	# Strategies	# Profiles	% Profiles
$U[6,4]$	34	1165	1.76
$U[5,5]$	30	5219	1.88
$U[5,8]$	29	9096	0.03
$H[5,3]$	32	608	10.16
$H[5,5]$	34	13197	2.62

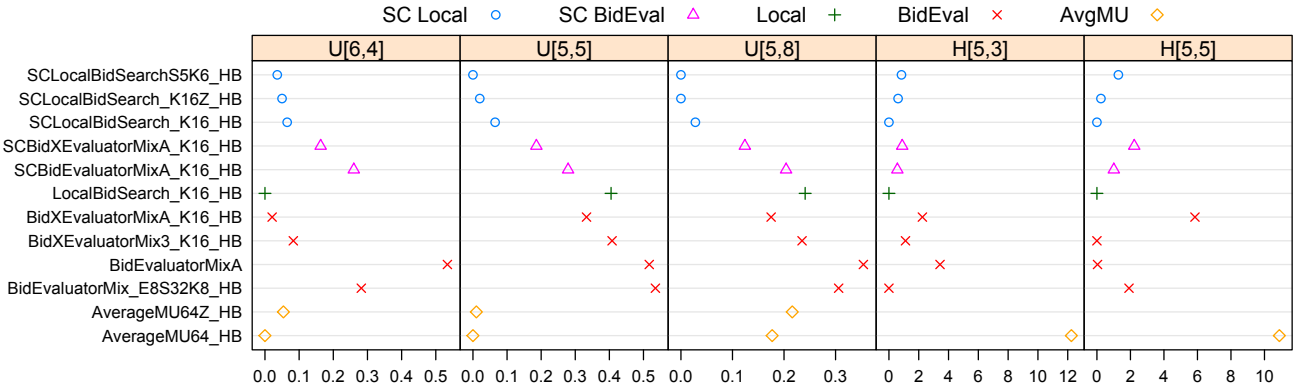
As it happens, our process identified exactly one symmetric mixed-strategy NE in each game. Of the 44 distinct strategies explored across environments, only seven were supported in equilibrium in any environment. Variants of SCLocalBid, a strategy that explicitly optimizes with respect to self-confirming prices, predominate in equilibrium in four out of five environments. LocalBid and the BidEvalMix strategies also explicitly optimize, but with respect to price predictions that are self-confirming for different strategies (see Appendix B). AverageMU64_HB performs remarkably well in the "U" environments, and is the sole non-optimizing heuristic to appear in equilibria.

Whether a strategy is in equilibrium or not is a crude binary classification of merit. We measure relative degrees of effectiveness by NE regret (Definition 7), as reported in Table 2. The table depicts the NE-regret values for 12 top strategies: all those ranked fourth or better in at least one environment. The values are indicated on a horizontal scale for each game environment, ranging from zero (indicating the strategy is in the support of equilibrium) to the highest NE regret value among the listed strategies. We also verified via a bootstrap technique that the results are statistically robust: for each identified equilibrium the one-tailed 90% confidence bound on regret is at the far left end (1% of the range) of the NE regret scale shown.

These results provide solid support for the SCLocalBid strategy. For the one environment it fails to participate in equilibrium, its NE regret is still quite low, thus we find it to be a strong all-around strategy. This situation contrasts starkly with prior findings for bidding in simultaneous ascending auctions [Wellman et al., 2008], where the best strategies for complementary (scheduling valuation) environments were awful in substitutes (homogeneous good)

[2]The extended version includes an online data supplement with full payoff data for the empirical games. The results we report here subsume those of a preliminary study [Yoon and Wellman, 2011]. That study set the groundwork for the current investigation by developing the infrastructure for simulation and derivation of self-confirming price predictions, and tuning parameters (e.g., number of evaluating samples) for several of the strategies. However, the preliminary results reflected sparser coverage of relevant profiles and a weaker overall set of strategy candidates.

Table 2: NE regret for top strategies across five environments.



environments, and vice versa. The fact that SCLocalBid performs so well aligns with our key theoretical finding, in support of optimal PP bidders with self-confirming price predictions. As the LocalBid search method is most effective in optimizing bids, it is consistent with our theory to find that both examples of strategies in this class are leaders among explicit optimizing strategies.

All the remaining top strategies are in the BidEval class (also explicit optimizers), except for AverageMU64. In contrast to the others, however, AverageMU64's quality is limited to one of the valuation distributions—the strategy performs poorly in homogeneous-good environments. This observation is consistent with the results reported by Boyan and Greenwald [2001], where an example environment with perfect substitutes was contrived to demonstrate the shortcomings of marginal-utility-based bidding. Given such examples, it is perhaps unsurprising that the heuristic strategies based on marginal value have a difficult time competing with explicit optimizers. If anything it is the observed success of AverageMU64 that is striking, but this outcome is consistent with past experimental results in an environment that exhibits substantial complementarity [Stone et al., 2003]. Still, compared to optimal PP bidders, AverageMU lacks robustness across valuation classes.

Finally, all the top strategies but one employ the highest-bid (HB) statistic in deriving self-confirming price distributions, as opposed to the actual transaction price. This, too, is aligned with what the theory would dictate. The lone exception in Table 2 is BidEvaluatorMixA, which performed impressively in one of the homogenous-good environments but not so well in the rest.

8 Conclusion

Our theoretical and experimental findings point to two key ingredients for developing effective bidding strategies for SimOSSB auctions with independent private values. The first is an algorithm for computing approximately optimal bid vectors given a probabilistic price prediction. We have found that a simple local search approach (LocalBid) achieves a high fraction of optimality, and that this translates into superior performance in strategic simulations (§7). The second ingredient is a method for computing self-confirming price predictions for a given price-prediction bidding strategy and a specification of a simultaneous-auction environment. We have found a simple iterative estimation procedure (§6) to be effective at finding price distributions that are approximately marginally self-confirming for a range of strategies and environments.

Our theoretical results say that if these ingredients can achieve their tasks perfectly, we have a solution (i.e., a Bayes-Nash equilibrium) to the corresponding simultaneous auction game. Approximations to the ideal in these ingredients yield approximate game solutions. Our computational experiments indicate that following this approach produces results that are as good or better than any other general method proposed for bidding in SimOSSB auctions. The evidence takes the form of a comprehensive empirical game-theoretic analysis, covering both complementary and substitutable valuation classes and a broad swath of heuristic strategies from the literature.

There is still room for improvement in our proposed methods, as well as opportunity to subject our conclusions to further empirical scrutiny. More sophisticated stochastic search techniques may improve upon our best bid optimization algorithms, and allow them to scale to larger environments with more complex valuations. Similarly, we do not consider our simple iterative method to be a last word on finding self-confirming price distributions. In particular, we expect that substantial improvement could be obtained by accounting for some joint dependencies in price predictions. Finally, further testing against alternative proposals or in alternative environments would go some way to bolstering or refuting our positive conclusions.

References

John Birge and Francois Louveaux. *Introduction to Stochastic Programming*. Springer, 1997.

Justin Boyan and Amy Greenwald. Bid determination in simultaneous auctions: An agent architecture. In *Third ACM Conference on Electronic Commerce*, pages 210–212, Tampa, 2001.

Peter Cramton. Simultaneous ascending auctions. In Cramton et al. [2005].

Peter Cramton, Yoav Shoham, and Richard Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2005.

Richard Engelbrecht-Wiggans and Robert J. Weber. An example of a multi-object auction game. *Management Science*, 25:1272–1277, 1979.

Amy Greenwald and Justin Boyan. Bidding under uncertainty: Theory and experiments. In *Twentieth Conference on Uncertainty in Artificial Intelligence*, pages 209–216, Banff, 2004.

Amy Greenwald, Seong Jae Lee, and Victor Naroditskiy. RoxyBot-06: Stochastic prediction and optimization in TAC travel. *Journal of Artificial Intelligence Research*, 36:513–546, 2009.

Amy Greenwald, Victor Naroditskiy, and Seong Jae Lee. Bidding heuristics for simultaneous auctions: Lessons from TAC travel. In Wolfgang Ketter, Han La Poutré, Norman Sadeh, Onn Shehory, and William Walsh, editors, *Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis*, volume 44 of *Lecture Notes in Business Information Processing*, pages 131–146. Springer-Verlag, 2010.

Patrick R. Jordan. *Practical Strategic Reasoning with Applications in Market Games*. PhD thesis, University of Michigan, 2010.

Patrick R. Jordan, Christopher Kiekintveld, and Michael P. Wellman. Empirical game-theoretic analysis of the TAC supply chain game. In *Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1188–1195, Honolulu, 2007.

Vijay Krishna. *Auction Theory*. Academic Press, second edition, 2010.

Vijay Krishna and Robert W. Rosenthal. Simultaneous auctions with synergies. *Games and Economic Behavior*, 17:1–31, 1996.

Anna Osepayshvili, Michael P. Wellman, Daniel M. Reeves, and Jeffrey K. MacKie-Mason. Self-confirming price prediction for bidding in simultaneous ascending auctions. In *Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 441–449, Edinburgh, 2005.

Michael Peters and Sergei Severinov. Internet auctions with many traders. *Journal of Economic Theory*, 130:220–245, 2006.

Zinovi Rabinovich, Victor Naroditskiy, Enrico H. Gerding, and Nicholas R. Jennings. Computing pure Bayesian Nash equilibria in games with finite actions and continuous types. Technical report, University of Southampton, 2011.

Daniel M. Reeves, Michael P. Wellman, Jeffrey K. MacKie-Mason, and Anna Osepayshvili. Exploring bidding strategies for market-based scheduling. *Decision Support Systems*, 39:67–85, 2005.

Robert W. Rosenthal and Ruqu Wang. Simultaneous auctions with synergies and common values. *Games and Economic Behavior*, 17:32–55, 1996.

Ariel Rubinstein and Asher Wolinsky. Rationalizable conjectural equilibrium: Between Nash and rationalizability. *Games and Economic Behavior*, 6:299–311, 1994.

Peter Stone, Robert E. Schapire, Michael L. Littman, János A. Csirik, and David McAllester. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. *Journal of Artificial Intelligence Research*, 19:209–242, 2003.

Balázs Szentes and Robert W. Rosenthal. Three-object two-bidder simultaneous auctions: Chopsticks and tetrahedra. *Games and Economic Behavior*, 44:114–143, 2003.

Michael P. Wellman. Methods for empirical game-theoretic analysis (extended abstract). In *Twenty-First National Conference on Artificial Intelligence*, pages 1552–1555, Boston, 2006.

Michael P. Wellman. *Trading Agents*. Morgan and Claypool, 2011.

Michael P. Wellman, Daniel M. Reeves, Kevin M. Lochner, and Yevgeniy Vorobeychik. Price prediction in a trading agent competition. *Journal of Artificial Intelligence Research*, 21:19–36, 2004.

Michael P. Wellman, Amy Greenwald, and Peter Stone. *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition*. MIT Press, 2007.

Michael P. Wellman, Anna Osepayshvili, Jeffrey K. MacKie-Mason, and Daniel M. Reeves. Bidding strategies for simultaneous ascending auctions. *B. E. Journal of Theoretical Economics (Topics)*, 8(1), 2008.

Michael P. Wellman, Lu Hong, and Scott E. Page. The structure of signals: Causal interdependence models for games of incomplete information. In *Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 727–735, Barcelona, 2011.

Dong Young Yoon and Michael P. Wellman. Self-confirming price prediction for bidding in simultaneous second-price sealed-bid auctions. In *IJCAI-11 Workshop on Trading Agent Design and Analysis*, Barcelona, 2011.

Latent Structured Ranking

Jason Weston
Google, New York, USA.
jweston@google.com

John Blitzer
Google, Mountain View, USA.
blitzer@google.com

Abstract

Many latent (factorized) models have been proposed for recommendation tasks like collaborative filtering and for ranking tasks like document or image retrieval and annotation. Common to all those methods is that during inference the items are scored independently by their similarity to the query in the latent embedding space. The structure of the ranked list (i.e. considering the set of items returned as a whole) is not taken into account. This can be a problem because the set of top predictions can be either too diverse (contain results that contradict each other) or are not diverse enough. In this paper we introduce a method for learning latent structured rankings that improves over existing methods by providing the right blend of predictions at the top of the ranked list. Particular emphasis is put on making this method scalable. Empirical results on large scale image annotation and music recommendation tasks show improvements over existing approaches.

1 INTRODUCTION

Traditional latent ranking models score the ith item $d_i \in \mathbb{R}^D$ given a query $q \in \mathbb{R}^D$ using the following scoring function:

$$f(q, d_i) = q^\top W d_i = q^\top U^\top V d_i, \quad (1)$$

where $W = U^\top V$ has a low rank parameterization, and hence $q^\top U^\top$ can be thought of as the latent representation of the query and Vd_i is equivalently the latent representation for the item. The latent space is n-dimensional, where $n \ll D$, hence U and V are $n \times D$ dimensional matrices. This formulation covers a battery of different algorithms and applications.

For example, in the task of collaborative filtering, one is required to rank items according to their similarity to the user, and methods which learn latent representations of both users and items have proven very effective. In particular, Singular Value Decomposition (SVD) (Billsus and Pazzani, 1998; Bell *et al.*, 2009) and Non-negative Matrix Factorization (NMF) (Lee and Seung, 2001) are two standard methods that at inference time use equation (1), although the methods to learn the actual parameters U and V themselves are different. In the task of document retrieval, on the other hand, one is required to rank text documents given a text query. The classical method Latent Semantic Indexing (LSI) (Deerwester *et al.*, 1990) is an unsupervised approach that learns from documents only, but still has the form of equation (1) at test time. More recently, supervised methods have been proposed that learn the latent representation from (query, document) relevance pairs, e.g. the method Polynomial Semantic Indexing (SSI) (Bai *et al.*, 2009). Finally, for multiclass classification tasks, particularly when involving thousands of possible labels, latent models have also proven to be very useful, e.g. the WSABIE model achieves state-of-the-art results on large-scale image (Weston *et al.*, 2011) and music (Weston *et al.*, 2012) annotation tasks. Moreover, all these models not only perform well but are also efficient in terms of computation time and memory usage.

Scoring a single item as in equation (1) is not the end goal of the tasks described above. Typically for recomendation and retrieval tasks we are interested in ranking the items. This is achieved by, after scoring each individual item using $f(q, d_i)$, sorting the scores, largest first, to produce a ranked list. Further, typically only the top few results are then presented to the user, it is thus critical that the method used performs well for those items. However, one potential flaw in the models described above is that scoring items individually as in eq. (1) does not fully take into account the joint set of items at the top of the list (even when optimizing top-of-the-ranked-list type loss functions).

The central hypothesis of this paper is that latent ranking methods could be improved if one were to take into account the structure of the ranked list during inference. In particular this would allow the model to make sure there is the right amount of consistency and diversity in the predictions.

Let us suppose for a given query that some of the predictions at the top of the ranked list are accurate and some are inaccurate. A model that improves the consistency of the predictions might improve overall accuracy. A structured ranking model that predicts items dependent on both the query *and* other items at the top of the ranked list can achieve such a goal. To give a concrete example, in a music recommendation task you might not want to recommend both "heavy metal" and "60s folk" in the same top k list. In that case, a structured model which encodes item-item similarities as well as query-item similarities could learn this by representing those two items with very different latent embedding vectors such that their pairwise item-item contribution is a large negative value, penalizing both items appearing in the top k. Note that a structured ranking model can do this despite the possibility that both items are a good match to the query, so an unstructured model would find this difficult to achieve.

Conversely, if improved results are gained from encouraging the top ranked items to be a rather diverse set for a particular query, then a structured model can learn to predict that instead. For example in the task of document retrieval, for ambiguous queries like "jaguar", which may refer either to a *Panthera* or to the car manufacturer, diversity should be encouraged. The goal of a structured ranker is to learn the optimal tradeoff between consistency and diversity on a case-by-case (per query) basis. As latent parameters are being learnt for each query type this is indeed possible.

In this work we propose a latent modeling algorithm that attempts to do exactly what we describe above. Our model learns to predict a ranked list that takes into account the structure of the top ranked items by learning query-item and item-item components. Inference then tries to find the maximally scoring set of documents. It should be noted that while there has been strong interest in building structured ranking models recently (Bakir *et al.*, 2007), to our knowledge this is the first approach of this type to do so for latent models. Further, the design of our algorithm is also particularly tuned to work on large scale datasets which are the common case for latent models, e.g. in collaborative filtering and large scale annotation and ranking tasks. We provide empirical results on two such large scale datasets, on a music recommendation task, and an image annotation task, that show our structured method brings accuracy improvements over the same method without structure as well as other standard baselines. We also provide some analysis of why we think this is happening.

The rest of the paper is as follows. Section 2 describes our method, Latent Structured Ranking (LaSR). Section 3 discusses previous work and connects them to our method. Section 4 describes our empirical results and finally Section 5 concludes.

2 METHOD

Given a query $q \in \mathcal{Q}$ our task is to rank a set of documents or items $\mathcal{D}$. That is, we are interested in outputting (and scoring) a permutation $\bar{d}$ of the set $\mathcal{D}$, where $\bar{d}_j$ is the jth item in the predicted ranked list. Our ultimate goal will be to design models which take into account not just individual document scores but the (learned) relative similarities of documents in different positions as well.

2.1 SCORING PERMUTATIONS BY SCORING INDIVIDUAL ITEMS

Let us begin by proposing methods for using the standard latent model of eq. (1) to score permutations. We need a method for transforming the scores for single documents into scores for permutations. Such transformations have been studied in several previous works, notably (Le and Smola, 2007). They show that finding maximally scoring permutations from single documents can be cast as a linear assignment problem, solvable in polynomial time with the Hungarian algorithm.

For the vanilla model we propose here, however, we can use a simple parameterization which allows for inference by sorting. For any given permutation we assign a score as follows:

$$f_{vanilla}(q, \bar{d}) = \sum_{i=1}^{|\bar{d}|} w_i (q^\top U^\top V \bar{d}_i), \qquad (2)$$

where for each position i in the permutation, we associate a weight w_i, where w_i can be any weights such that $w_1 > w_2 > \cdots > w_{|\bar{d}|} \geq 0$. For example, one can just set $w_i = \frac{1}{i}$. Inference using this model is then performed by calculating:

$$F_{vanilla}(q) = \text{argmax}_{\bar{d}'}[f_{vanilla}(q, \bar{d}')].$$

In this case, computing the best-scoring assignment is simply a matter of sorting documents by their scores from eq. (1). To see this note that the score of any unsorted pair can be increased by sorting, since the positional weights w_i are fixed and decreasing.

2.2 LATENT STRUCTURED RANKING

The fundamental hypothesis of this paper is that including knowledge about the structure of the rankings at inference time will improve the overall set of ranked items. That is, we want to define a model where the score of a document $\bar{d}_i$ does not only depend on the query q but also on the other items and their respective positions as well. What is more, we would prefer a model that places more weight on the top items in a permutation (indeed, this is reflected by common ranking losses like MAP and precision@k).

This leads us to propose the following class of Latent Structured Ranking (LaSR) models:

$$f_{lsr}(q, \bar{d}) = \sum_{i=1}^{|\bar{d}|} w_i (q^\top U^\top V \bar{d}_i) + \sum_{i,j=1}^{|\bar{d}|} w_i w_j (\bar{d}_i^\top S^\top S \bar{d}_j) \ . \tag{3}$$

In addition to the parameters of eq. (2), we now introduce the additional parameter S. S takes into account the structure of the predicted ranked list. $S^\top S$ is a low rank matrix of item-item similarities where S is a $n \times D$ matrix, just like U and V, and must also be learnt by the model using training data.

2.3 CHOICE OF THE w_i PARAMETERS

The weights w are crucial to the usefulness of the matrix in the second term of eq. (3). If $w_i = 1$ for all i then the entire second term would always be the same no matter what choice of ranking $\bar{d}$ one chooses. If the position weights w_i are decreasing, however, then the structural term S is particularly meaningful at the top of the list.

As suggested before in Section 2.1 we could choose $w_i = \frac{1}{i}$. In that case the items that are at the top of the predicted ranked list dominate the overall score from the second term. In particular, the pairwise item-item similarities between items in the top-ranked positions play a role in the overall choice of the entire ranked list $\bar{d}$. Our model can hence learn the consistency vs. diversity tradeoff within the top k we are interested in.

However, if one knows in advance the number of items one wishes to show to the user (i.e. the top k) then one could choose directly to only take into account those predictions:

$$w_i = 1/i, \quad \text{if } i \leq k, \quad \text{and 0 otherwise.} \tag{4}$$

As we will see this also has some computational advantages due to its sparsity, and will in fact be our method of choice in the algorithm we propose.

2.4 MODEL INFERENCE

At test time for a given query we need to compute:

$$F_{lsr}(q) = \text{argmax}_{\bar{d}'}[f_{lsr}(q, \bar{d}')]. \tag{5}$$

Just as inference in the vanilla model can be cast as a linear assignment problem, inference in the LaSR model can be cast as a quadratic assignment problem (Lacoste-Julien *et al.*, 2006). This is known to be NP hard, so we must approximate it. In this section, we briefly discuss several alternatives.

- Linear programming relaxation: Since we know we can cast our problem as quadratic assignment, we could consider directly using the linear programming relaxation suggested by (Lacoste-Julien *et al.*, 2006). In our experiments, however, we have tens of thousands of labels. Solving even this relaxed LP per query is computationally infeasible for problems of this size. We note that WSABIE's (Weston *et al.*, 2011) sampling-based technique is an attempt to overcome even linear time inference for this problem.

- Greedy structured search: we could also consider a greedy approach to approximately optimizing eq. (5) as follows: (i) pick the document $\bar{d}_1 \in \mathcal{D}$ that maximizes:

$$f_{greedy}(q, \bar{d}_1) = w_1 (q U^\top V \bar{d}_1) + (w_1)^2 (\bar{d}_1^\top S^\top S \bar{d}_1) \tag{6}$$

  and then fix that document as the top ranked prediction. (ii) Find the second best document dependent on the first, by maximizing (for $N = 2$):

$$f_{greedy}(q, \bar{d}_N) = w_N (q U^\top V \bar{d}_N) q + \sum_{i=1}^{N} w_i w_N (\bar{d}_i^\top S^\top S \bar{d}_N).$$

  Finally, (iii) repeat the above, greedily adding one more document each iteration by considering the above equation for $N = 3, \ldots, k$ up to the number of desired items to be presented to the user.

  This method has complexity $O(k^2 |\mathcal{D}|)$. Its biggest drawback is that the highest-scoring document is chosen using the vanilla model. Even if we could improve our score by choosing a different document, taking into account the pairwise scores with other permutation elements, this algorithm will not take advantage of it. Another way to look at this is, precision@1 would be no better than using the vanilla model of eq. (1).

  The greedy procedure also permits beam search variants. Using a beam of M candidates this gives

a complexity of $O(Mk^2|\mathcal{D}|)$. This is tractable at test time, but the problem is that during (online) learning one would have to run this algorithm per query, which we believe is still too slow for the cases we consider here.

- Iterative search: Motivated by the defects in greedy search and LP relaxation, we propose one last, iterative method. This method is analogous to inference by iterated conditional modes in graphical models (Besag, 1986). (i) On iteration $t = 0$ predict with an unstructured model (i.e. do not use the second term involving S):

$$f_{iter:t=0}(q, \bar{d}) = \sum_{i=1}^{|\bar{d}|} w_i (qU^\top V \bar{d}_i). \quad (7)$$

As mentioned before, computing the best ranking $\bar{d}$ just involves sorting the scores $qU^\top V d_i$ and ordering the documents, largest first. Utilizing the sparse choice of $w_i = 1/i$, if $i \leq k$, and 0 otherwise described in Section 2.3 we do not have to sort the entire set, but are only required to find the top k which can be done in $O(|\mathcal{D}| \log k)$ time using a heap. Let us denote the predicted ranked list as $\bar{d}^0$ and in general on each iteration t we are going to make predictions $\bar{d}^t$. (ii) On subsequent iterations, we maximize the following scoring function:

$$f_{iter:t>0}(q, \bar{d}) = \sum_{i=1}^{|\bar{d}|} w_i (qU^\top V \bar{d}_i) + \sum_{i,j=1}^{|\bar{d}|} w_i w_j (\bar{d}_i^\top S^\top S \bar{d}_j^{t-1}). \quad (8)$$

As $\bar{d}^{t-1}$ is now fixed on iteration t, the per-document d_i scores

$$(qU^\top V \bar{d}_i) + \sum_{j=1}^{|\bar{d}|} w_j (\bar{d}_i^\top S^\top S \bar{d}_j^{t-1}) \quad (9)$$

are now independent of each other. Hence, they can be calculated individually and, as before, can be sorted or the top k can be found, dependent on the choice of w. If we use the sparse w of eq. (4) (which we recommend) then the per-document scores are also faster to compute as we only require:

$$(qU^\top V \bar{d}_i) + \sum_{j=1}^{k} w_j (\bar{d}_i^\top S^\top S \bar{d}_j^{t-1}).$$

Overall this procedure then has complexity $O(Tk|\mathcal{D}|)$ when running for T steps. While this does not look at first glance to be any faster than the greedy or beam search methods at testing time, it has important advantages at training time as we will see in the next section.

2.5 LEARNING

We are interested in learning a *ranking* function where the top k retrieved items are of particular interest as they will be presented to the user. We wish to optimize all the parameters of our model jointly for that goal. As the datasets we intend to target are large scale, stochastic gradient descent (SGD) training seems a viable option. However, during training we cannot afford to perform full inference during each update step as otherwise training will be too slow. A standard loss function that already addresses that issue for the unstructured case which is often used for retrieval is the margin ranking criterion (Herbrich *et al.*, 2000; Joachims, 2002). In particular, it was also used for learning factorized document retrieval models in Bai *et al.* (2009). The loss can be written as:

$$err_{AUC} = \sum_{i=1}^{m} \sum_{d^- \neq d_i} \max(0, 1 - f(q_i, d_i) + f(q_i, d^-)). \quad (10)$$

For each training example $i = 1, \ldots, m$, the positive item d_i is compared to all possible negative items $d^- \neq d_i$, and one assigns to each pair a cost if the negative item is larger or within a "margin" of 1 from the positive item. These costs are called *pairwise violations*. Note that all pairwise violations are considered equally if they have the same margin violation, independent of their position in the list. For this reason the margin ranking loss might not optimize the top k very accurately as it cares about the average rank.

For the standard (unstructured) latent model case, the problem of optimizing the top of the rank list has also recently been addressed using sampling techniques (Weston *et al.*, 2011) in the so-called WARP (Weighted Approximately Ranked Pairwise) loss. Let us first write the predictions of our model for all items in the database as a vector $\bar{f}(q)$ where the i^{th} element is $\bar{f}_i(q) = f(q, d_i)$. One then considers a class of ranking error functions:

$$err_{WARP} = \sum_{i=1}^{m} L(rank_{d_i}(\bar{f}(q_i))) \quad (11)$$

where $rank_{d_i}(\bar{f}(q_i))$ is the margin-based rank of the labeled item given in the i^{th} training example:

$$rank_{d_i}(\bar{f}(q)) = \sum_{j \neq i} \theta(1 + \bar{f}_j(q) \geq \bar{f}_i(q)) \quad (12)$$

where θ is the indicator function, and $L(\cdot)$ transforms this rank into a loss:

$$L(r) = \sum_{i=1}^{r} \alpha_i, \text{ with } \alpha_1 \geq \alpha_2 \geq \cdots \geq 0. \quad (13)$$

The main idea here is to *weight* the pairwise violations depending on their position in the ranked list. Different choices of α define different weights (importance) of the relative position of the positive examples in the ranked list. In particular it was shown that by choosing $\alpha_i = 1/i$ a smooth weighting over positions is given, where most weight is given to the top position, with rapidly decaying weight for lower positions. This is useful when one wants to optimize precision at k for a variety of different values of k at once Usunier *et al.* (2009). (Note that choosing $\alpha_i = 1$ for all i we have the same AUC optimization as equation (10)).

We can optimize this function by SGD following the authors of Weston *et al.* (2011), that is samples are drawn at random, and a gradient step is made for each draw. Due to the cost of computing the exact rank in (11) it is approximated by sampling. That is, for a given positive label, one draws negative labels until a violating pair is found, and then approximates the rank with

$$rank_d(\bar{f}(q)) \approx \left\lfloor \frac{|\mathcal{D}| - 1}{N} \right\rfloor$$

where $\lfloor . \rfloor$ is the floor function, $|\mathcal{D}|$ is the number of items in the database and N is the number of trials in the sampling step. Intuitively, if we need to sample more negative items before we find a violator then the rank of the true item is likely to be small (it is likely to be at the top of the list, as few negatives are above it).

This procedure for optimizing the top of the ranked list is very efficient, but it has a disadvantage with respect to structured learning: we cannot simply sample and score items any longer as we need to somehow score entire permutations. In particular, it is not directly applicable to several of the structured prediction approaches like LP, greedy or beam search. That is because we cannot compute the score of $\bar{f}_i$ independently because they depend on the ranking of all documents, which then makes the sampling scheme invalid. However, for (a variant of) the iterative algorithm which we described in the previous section the WARP (or AUC) technique can still be used.

The method is as follows. In the first iteration the model scores in eq. (7) are independent and so we can train using the WARP (or AUC) loss. We then have to compute $\bar{d}^0$ (the ranking of items) for each training example for use in the next iteration. Note that using the sparse w of eq. (4) this is $O(\mathcal{D} \log k)$ to compute, and storage is also only a $|\mathcal{D}| \times k$ matrix of top items. After computing $\bar{d}^0$, in the second iteration we are again left with independent scoring functions $\bar{f}_i$ as long as we make one final modification, instead

Algorithm 1 LASR training algorithm

Input: Training pairs $\{(q_i, d_i)\}_{i=1,\ldots,l}$.
Initialize model parameters U_t, V_t and S_t (we use mean 0, standard deviation $\frac{1}{\sqrt{d}}$) for each t.
for $t = 0, \ldots, T$ **do**
  repeat
    if $t = 0$ **then**
      $f(q, d) = qU_0^\top V_0 d$.
    else
      $f(q, d) = qU_t^\top V_t d + \sum_{j=1}^{k} w_j d^\top S_t^\top S_t \bar{d}_j^{t-1}$
    end if
    Pick a random training pair (q, d^+).
    Compute $f(q, d^+)$.
    Set $N = 0$.
    repeat
      Pick a random document $d^- \in \mathcal{D}, d \neq d_i$.
      Compute $f(q, d^-)$.
      $N = N + 1$.
    until $f(q^+, d^+) < f(q^+, d^-) + 1$ or $N \geq |\mathcal{D}| - 1$
    if $f(q^+, d^+) < f(q^+, d^-) + 1$ **then**
      Make a gradient step to minimize:
      $L(\left\lfloor \frac{|\mathcal{D}|-1}{N} \right\rfloor) \max(1 - f(q^+, d^+) + f(q^+, d^-), 0)$.
      Project weights to enforce constraints,
      i.e. if $||U_{ti}|| > C$ then $U_{ti} \leftarrow (CU_{ti})/||U_{ti}||$,
        $i = 1, \ldots, D$ (and likewise for V_t and S_t).
    end if
  until validation error does not improve.
  For each training example, compute the top k ranking documents $\bar{d}_i^t, \quad i = 1, \ldots, k$ for iteration t using $f(q, d)$ defined above.
end for

of using eq. (8) we instead use:

$$\begin{aligned} f_{iter:t>0}(q, \bar{d}) &= \sum_{i=1}^{|\bar{d}|} w_i (qU_t^\top V_t \bar{d}_i) \\ &+ \sum_{i,j=1}^{|\bar{d}|} w_i w_j (\bar{d}_i^\top S_t^\top S_t \bar{d}_j^{t-1}). \quad (14) \end{aligned}$$

on iteration t, where U_t, V_t and S_t are separate matrices for each iteration. This decouples the learning at each iteration. Essentially, we are using a cascade-like architecture of t models trained one after the other. Note that if a global optimum is reached for each t then the solution should always be the same or improve over step $t - 1$, as one could pick the weights that give exactly the same solution as for step $t - 1$.

So far, the one thing we have failed to mention is regularization during learning. One can regularize the parameters by preferring smaller weights. We constrain them using $||S_{ti}|| \leq C$, $||U_{ti}|| \leq C$, $||V_{ti}|| \leq C$, $i = 1, \ldots, |\mathcal{D}|$. During SGD one projects the parameters back on to the constraints at each step, following

the same procedure used in several other works, e.g. Weston *et al.* (2011); Bai *et al.* (2009). We can optimize hyperparameters of the model such as C and the learning rate for SGD using a validation set.

Overall, our preferred version of Latent Structured Ranking that combines all these design decisions is given in Algorithm 1.

3 PRIOR WORK

In the introduction we already mentioned several latent ranking methods: SVD (Billsus and Pazzani, 1998; Bell *et al.*, 2009), NMF (Lee and Seung, 2001), LSI (Deerwester *et al.*, 1990), PSI (Bai *et al.*, 2009) and WSABIE (Weston *et al.*, 2011). We should mention that many other methods exist as well, in particular probabilistic methods like pLSA (Hofmann, 1999) and LDA (Blei *et al.*, 2003). None of those methods, whether they are supervised or unsupervised, take into the structure of the ranked list as we do in this work, and we will use several of them as baselines in our experiments.

There has been a great deal of recent work on structured output learning (Bakir *et al.*, 2007), particularly for linear or kernel SVMs (which are not latent embedding methods). In methods like Conditional Random Fields (Lafferty *et al.*, 2001), SVM-struct (Tsochantaridis *et al.*, 2004) LaSO (Daumé III and Marcu, 2005) and SEARN (Daumé *et al.*, 2009) one learns to predict an output which has structure, e.g. for sequence labeling, parse tree prediction and so on. Predicting ranked lists can also be seen in this framework. In particular LaSO (Daumé III and Marcu, 2005) is a general approach that considers approximate inference using methods like greedy approximation or beam search that we mentioned in Section 2.4. As we said before, due to the large number of items we are ranking many of those approaches are infeasible. In our method, scalability is achieved using a cascade-like training setup, and in this regard is related to (Weiss *et al.*, 2010). However, unlike that work, we do not use it to prune the set of items considered for ranking, we use it to consider pairwise item similarities.

The problem of scoring entire permutations for ranking is well-known and has been investigated by many authors (Yue *et al.*, 2007a,b; Le and Smola, 2007; Xia *et al.*, 2008). These works have primarily focused on on using knowledge of the structure (in this case the predicted positions in the ranked list) in order to optimize the right metric, e.g. MAP or precision@k. In that sense, methods like WSABIE which uses the WARP loss already use structure in the same way. In our work we also optimize top-of-the-ranked-list metrics by using WARP, but in addition we also use the ranking structure to make predictions dependent on the query and the other predicted items during inference by encoding this in the model itself. That is, in our work we explicitly seek to use (and learn) inter-document similarity measures.

There has been work on taking into account inter-document similarities during ranking. The most famous and prominent idea is pseudo-relevance feedback via query expansion (Rocchio, 1971). Pseudo-relevance works by doing normal retrieval (e.g. using cosine similarity in a vector space model), to find an initial set of most relevant documents, and then assuming that the top k ranked documents are relevant, and performing retrieval again by adjusing the cosine similarity based on previously retrieved documents. In a sense, LaSR is also a pseudo-relevance feedback technique, but where inter-document similarities are learned to minimize ranking loss.

More recently, some authors have investigated incorporating inter-document similarity during ranking. Qin *et al.* (2008) have investigated incorporating a fixed document-document similarity feature in ranking. In their work, however, they did not score permutations. Instead, each document was associated with a relevance score and the authors treated learning as a structured regression problem. For a situation with implicit feedback, Raman *et al.* (2012) investigate an inference technique similar to our greedy algorithm. Volkovs and Zemel (2009) also explored listwise ranking using pairwise document interactions in a probabilistic setup. To the best of our knowledge, however, none of these methods investigate a learned inter-document similarity (i.e. latent parameters for that goal), which is the most powerful feature of LaSR.

4 EXPERIMENTS

We considered two large scale tasks to test our proposed method. The first is a music recommendation task with over 170,000 artists (possible queries or items) and 5 million training pairs. The second is a task of large scale image annotation with over 15,000 labels and 7 million training examples.

4.1 MUSIC RECOMMENDATION TASK

The first task we conducted experiments on is a large scale music recommendation task. Given a query (seed) artist, one has to recommend to the user other artists that go well together with this artist if one were listening to both in succession, which is the main step in playlisting and artist page recommendation on sites like `last.fm`, `music.google.com` and programs such as iTunes and `http://the.echonest.com/`.

Table 1: Recommendation Results on the music recommendation task. We report for recall at 5, 10, 30 and 50 for our method and several baselines.

Method	R@5	R@10	R@30	R@50
NMF	3.76%	6.38%	13.3%	17.8%
SVD	4.01%	6.93%	13.9%	18.5%
LaSR $(t = 0)$	5.60%	9.49%	18.9%	24.8%
LaSR $(t = 1)$	6.65%	10.73%	20.1%	26.7%
LaSR $(t = 2)$	6.93%	10.95%	20.3%	26.5%

Table 2: Changing the embedding size on the music recommendation task. We report R@5 for various dimensions n.

Method $n =$	25	50	100	200
NMF	2.82%	3.76%	3.57%	4.82%
SVD	3.61%	4.01%	4.53%	5.28%
LaSR $(t = 0)$	5.23%	5.60%	6.24%	6.42%

We used the "Last.fm Dataset - 1K users" dataset available from `http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html`. This dataset contains (user, timestamp, artist, song) tuples collected from the Last.fm (`www.lastfm.com`) API, representing the listening history (until May 5th, 2009) for 992 users and 176,948 artists. Two consecutively played artists by the same user are considered as a (query, item) pair. Hence, both q_i and d_i are $D = 176,948$ sparse vectors with one non-zero value (a one) indicating which artist they are. One in every five days (so that the data is disjoint) were left aside for testing, and the remaining data was used for training and validation. Overall this gave 5,408,975 training pairs, 500,000 validation pairs (for hyperparameter tuning) and 1,434,568 test pairs.

We compare our Latent Structured Ranking approach to the same approach *without* structure by only performing one iteration of Algorithm 1. We used $k = 20$ for eq. (4). We also compare to two standard methods of providing latent recommendations, Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF). For SVD the Matlab implementation is used, and for NMF the implementation at `http://www.csie.ntu.edu.tw/~cjlin/nmf/` is used.

Main Results We report results comparing NMF, SVD and our method, Latent Structured Ranking (LaSR) in Table 1. For every test set (query, item) pair we rank the document set $\mathcal{D}$ according to the query and record the position of the item in the ranked list. We then measure the recall at 5, 10, 30 and 50. (Note that because there is only one item in the pair, precision at k is equal to recall@k divided by k.) We then average the results over all pairs. For all methods the latent dimension $n = 50$. For LaSR, we give results for iterations $t = 0, \ldots, 2$, where $t = 0$ does not use the structure. LaSR with $t = 0$ already outperforms SVD and NMF. LaSR optimizes the top of the ranked list at training time (via the WARP loss), whereas SVD and NMF do not, which explains why it can perform better here on top-of-the-list metrics. We tested LaSR $t = 0$ using the AUC loss (10) instead of WARP (11) to check this hypothesis and we obtained a recall at 5, 10, 30 and 50 of 3.56%, 6.32%, 14.8% and 20.3% respectively which are slightly worse, than, but similar to SVD, thus confirming our hypothesis. For LaSR with $t = 1$ and $t = 2$ our method takes into account the structure of the ranked list at inference time, $t = 1$ outperforms iteration $t = 0$ that does not use the structure. Further slight gains are obtained with another iteration $(t = 2)$.

Changing the Embedding Dimension The results so far were all with latent dimension $n = 50$. It could be argued that LaSR with $t > 0$ has more capacity (more parameters) than competing methods, and those methods could have more capacity by increasing their dimension n. We therefore report results for various embedding sizes $(n = 10, 25, 50, 100)$ in Table 2. The results show that LaSR $(t = 0)$ consistently outperforms SVD and NMF for all the dimensions tried, but even with 200 dimensions, the methods that do not use structure (SVD, NMF and LaSR $t = 0$) are still outperformed by LaSR that does use structure $(t > 0)$ even with $n = 50$ dimensions.

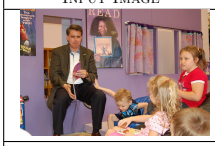
Analysis of Predictions We give two example queries and the top ranked results for LaSR with and without use of structure ($(t = 0)$ and $(t = 1)$) in Table 3. The left-hand query is a popular artist "Bob Dylan". LaSR $(t = 0)$ performs worse than $(t = 1)$ with "Wilco" in positon 1 - the pair ("Bob Dylan","Wilco") only appears 10 times in the test set, whereas ("Bob Dylan", "The Beatles") appears 40 times, and LaSR $(t = 1)$ puts the latter in the top position. In general $t = 1$ improves the top ranked items over $t = 0$, removing or demoting weak choices, and promoting some better choices. For example, "Sonic Youth" which is a poor match is demoted out of the top 20. The second query is a less popular artist "Plaid" who make electronic music. Adding structure to LaSR again improves the results in this case by boosting relatively more popular bands like "Orbital" and "$\mu - ziq$", and relatively more related bands like "Four-Tet" and "Squarepusher", whilst demoting some lesser known bands.

Table 3: Music Recommendation results for our method LaSR with ($t = 1$) and without ($t = 0$) using the structure. We show top ranked results for a popular query, "Bob Dylan" (folk rock music) and a less popular query "Plaid" (electronic music). Total numbers of train and test pairs for given artist pairs are in square brackets, and totals for all artists shown are given in the last row. Artists where the two methods differ are labeled with an asterisk. Adding structure improves the results, e.g. unrelated bands like "Sonic Youth" are demoted in the "Bob Dylan" query, and relatively more popular bands like Orbital and $\mu-$ziq and more related bands like Four-Tet and Squarepusher are boosted for the "Plaid" query.

LaSR $t = 0$ *(no structure)*	LaSR $t = 1$ *(structured ranking)*
Query: Bob Dylan	Query: Bob Dylan
Wilco [53,10]	The Beatles [179,40]
The Rolling Stones [40,9]	Radiohead [61,16]
The Beatles [179,40]	The Rolling Stones [40,9]
R.E.M. [35,12]	Johnny Cash [49,11]
Johnny Cash [49,11]	The Cure [38,10]
Beck [42,18]	David Bowie [48,12]
David Bowie [48,12]	Wilco [53,10]
Pixies [31,4]	Pink Floyd* [30,8]
Belle And Sebastian [28,4]	U2* [40,16]
The Beach Boys* [22,6]	The Smiths [25,7]
The Cure [38,10]	Sufjan Stevens [23,4]
Arcade Fire* [34,6]	R.E.M. [35,12]
Radiohead [61,16]	Belle And Sebastian [28,4]
Sonic Youth* [35,9]	Beck [42,18]
Bruce Springsteen [41,11]	The Shins* [22,13]
The Smiths [25,7]	Pixies [31,4]
The Velvet Underground* [29,11]	Ramones* [36,8]
Sufjan Stevens [23,4]	Bruce Springsteen [44,11]
Tom Waits* [19,13]	Death Cab For Cutie* [32,7]
(Train,Test) totals = [835,213]	(Train,Test) totals = [856,220]

LaSR $t = 0$ *(no structure)*	LaSR $t = 1$ *(with structured ranking)*
Query: Plaid	Query: Plaid
Boards Of Canada [27,5]	Boards Of Canada [27,5]
Autechre [13,1]	Aphex Twin [9,3]
Aphex Twin [9,3]	Autechre [13,1]
Biosphere [6,1]	Biosphere [6,1]
Wagon Christ [3,1]	Squarepusher [11,2]
Amon Tobin [6,3]	Future Sound Of London [5,2]
Arovane [5,1]	Four Tet* [6,2]
Future Sound Of London [5,2]	Massive Attack* [4,2]
The Orb [3,2]	Arovane [5,1]
Squarepusher [11,2]	Air [9]
Bola [5,2]	The Orb [3,2]
Chris Clark* [4,2]	Isan [3,1]
Kettel* [3,0]	Amon Tobin [6,3]
Ulrich Schnauss [7,1]	Bola [5,2]
Apparat* [3,0]	Orbital* [7,1]
Isan [3,1]	Murcof [4]
Air [9]	Ulrich Schnauss [7,1]
Clark* [0,0]	Wagon Christ [3,1]
Murcof [4,0]	μ-Ziq* [5,3]
(Train,Test) totals = [126,27]	(Train,Test) totals = [138,33]

Table 4: Image Annotation Results comparing Wsabie with LaSR. The top 10 labels of each method is shown, the correct label (if predicted) is shown in bold. In many cases LaSR can be seen to improve on Wsabie by demoting bad predictions e.g. war paint, soccer ball, segway, denture, rottweiler, reindeer, tv-antenna, leopard frog (one example from each of the first 8 images). In the last 3 images neither method predicts the right label (armrest, night snake and heifer) but LaSR seems slightly better (e.g. more cat, snake and cow predictions).

Input Image	Wsabie	LaSR
	workroom, life office, war paint, **day nursery**, homeroom, salon, foundling hospital, teacher, sewing room, canteen	workroom, salon, **day nursery**, schoolroom, student, homeroom, clothespress, day school, sewing room, study
	soccer ball, tent, inflated ball, fly tent, shelter tent, camping, **pack tent**, magpie, field tent, white admiral butterfly	shelter tent, tent, camping, **pack tent**, fly tent, mountain tent, tent flap, two-man tent, field tent, pop tent
	roller skating, skateboard, cross-country skiing, skating, segway, hockey stick, skateboarding, skating rink, crutch, roller skate wheel	roller skate wheel, roller skating, skateboard, in-line skate, **roller skate**, cross-country skiing, skateboarding, unicycle, skate, skating
	partial denture, round-bottom flask, **flask**, panty girdle, night-light, patchouly, organza, organdy, baby, wig	round-bottom flask, **flask**, rummer, tornado lantern, spotlight, foetus, oil lamp, sconce, infrared lamp, decanter
	rubber boot, rottweiler, pillar, **combat boot**, calf, riding boot, trouser, watchdog, shepherd dog, leg	**combat boot**, riding boot, rubber boot, leg covering, rubber, trouser, tabis boot, trouser leg, boot, lawn tool
	reaper, reindeer, cannon, steamroller, seeder, plow, tractor, combine, cannon, caribou	reaper, seeder, **gun carriage**, farm machine, combine, plow, caribou, haymaker, tractor, trench mortar,

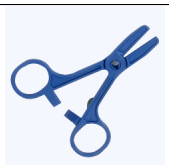
Input Image	Wsabie	LaSR
	tv-antenna, transmission line, shears, refracting telescope, scissors, electrical cable, chain wrench, astronomical telescope, wire, boat hook,	scissors, shears, tinsnips, windmill, garden rake, tv-antenna, forceps, safety harness, **medical instrument**, plyers
	leopard frog, sauceboat, sugar bowl, cream pitcher, tureen, spittoon, pitcher, earthenware, rainbow fish, slop bowl	tureen, glazed earthenware, slop bowl, sauceboat, spittoon, **cullender**, earthenware, punch bowl, sugar bowl, cream pitcher
	redpoll, frogmouth, screech owl, possum, grey parrot, finch, gypsy moth, gray squirrel, lycaenid butterfly, cairn terrier	frogmouth, soft-coated wheaten terrier, gray squirrel, cairn terrier, gypsy moth, tabby cat, chinchilla laniger, egyptian cat, burmese cat, abyssinian cat
	grass snake, garter snake, ribbon snake, common kingsnake, black rattler, western ribbon snake, european viper, pickerel frog spiny anteater, eastern ground snake	garter snake, european viper, grass snake, california whipsnake, common kingsnake, eastern ground snake, northern ribbon snake, puff adder, whipsnake, black rattler
	black bear, black vulture, labiated bear, greater gibbon, vulture, american black bear, cuckoo, black sheep, buffalo, yak,	black bear, yak, bear, american black bear, stocky horse, calf, wild boar, bull, draft horse, black angus

Table 5: **Summary of Test Set Results on Imagenet.** Recall at 1, 5, 10, Mean Average Precision and Mean Rank are given.

Algorithm	r@1	r@5	r@10	MAP	MR
One-vs-Rest	2.83%	8.48%	13.2%	0.065	667
Rank SVM	5.35%	14.1%	19.3%	0.102	804
WSABIE	8.39%	19.6%	26.3%	0.144	626
LaSR ($t = 1$)	9.45%	22.1%	29.1%	0.161	523

4.2 IMAGE ANNOTATION TASK

ImageNet (Deng *et al.*, 2009) (`http://www.image-net.org/`) is a large scale image database organized according to WordNet (Fellbaum, 1998). WordNet is a graph of linguistic terms, where each concept node consists of a word or word phrase, and the concepts are organized within a hierarchical structure. ImageNet is a growing image dataset that attaches quality-controlled human-verified images to these concepts by collecting images from web search engines and then employing annotators to verify whether the images are good matches for those concepts, and discarding them if they are not. For many nouns, hundreds or even thousands of images are labeled. We can use this dataset to test image annotation algorithms. We split the data into train and test and try to learn to predict the label (annotation) given the image. For our experiments, we downloaded the "Spring 2010" release which consists of 9 million images and 15,589 possible concepts (this is a different set to (Weston *et al.*, 2011) but our baseline results largely agree). We split the data into 80% for training, 10% for validation and 10% for testing.

Following (Weston *et al.*, 2011) we employ a feature representation of the images which is an ensemble of several representations which is known to perform better than any single representation within the set (see e.g. (Makadia *et al.*, 2008)). We thus combined multiple feature representations which are the concatenation of various spatial (Grauman and Darrell, 2007) and multiscale color and texton histograms (Leung and Malik, 1999) for a total of about 5×10^5 dimensions. The descriptors are somewhat sparse, with about 50,000 non-zero weights per image. Some of the constituent histograms are normalized and some are not. We then perform Kernel PCA (Schoelkopf *et al.*, 1999) on the combined feature representation using the intersection kernel (Barla *et al.*, 2003) to produce a 1024 dimensional input vector for training.

We compare our proposed approach to several baselines: one-versus-rest large margin classifiers (One-vs-Rest) of the form $f_i(x) = w_i^\top x$ trained online to perform classification over the 15,589 classes, or the same models trained with a ranking loss instead, which we refer to as RANK SVM, as it is an online version of the pairwise multiclass (ranking) loss of (Weston and Watkins, 1999; Crammer and Singer, 2002). Finally, we compare to WSABIE a (unstructured) latent ranking method which has yielded state-of-the-art performance on this task. For all methods, hyperparameters are chosen via the validation set.

Results The overall results are given in Table 5. One-vs-Rest performs relatively poorly (2.83% recall@1), perhaps because there are so many classes (over 15,000) that the classifiers are not well calibrated to each other (as they are trained independently). The multiclass ranking loss of Rank SVM performs much better (5.35% recall@1) but is still outperformed by WSABIE (8.39% recall@1). WSABIE uses the WARP loss to optimize the top of the ranked list and its good performance can be explained by the suitability of this loss function for measure like recall@k. LaSR with $t = 0$ is essentially identical to WSABIE in this case and so we use that model as our "base learner" for iteration 0. LaSR ($t = 1$), that does use structure, outperforms WSABIE with a recall@1 of 9.45%.

Some example annotations are given in Table 4. LaSR seems to provide more consistent results than WSABIE on several queries (with less bad predictions in the top k) which improves the overall results, whilst maintaining the right level of diversity on others.

5 CONCLUSION

In this paper we introduced a method for learning a latent variable model that takes into account the structure of the predicted ranked list of items given the query. The approach is quite general and can potentially be applied to recommendation, annotation, classification and information retrieval tasks. These problems often involve millions of examples or more, both in terms of the number of training pairs and the number of items to be ranked. Hence, many otherwise straight-forward approaches to structured prediction approaches might not be applicable in these cases. The method we proposed is scalable to these tasks.

Future work could apply latent structured ranking to more applications, for example in text document retrieval. Moreover, it would be interesting to explore using other algorithms as the "base algorithm" which we add the structured predictions to. In this work, we used the approach of (Weston *et al.*, 2011) as our base algorithm, but it might also be possible to make structured ranking versions of algorithms like Non-negative matrix factorization, Latent Semantic Indexing or Singular Value Decomposition as well.

References

Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., Cortes, C., and Mohri, M. (2009). Polynomial semantic indexing. In *Advances in Neural Information Processing Systems (NIPS 2009)*.

Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., and Vishwanathan, S. V. N. (2007). *Predicting Structured Data (Neural Information Processing)*. The MIT Press.

Barla, A., Odone, F., and Verri, A. (2003). Histogram intersection kernel for image classification. *International Conference on Image Processing (ICIP)*, **3**, III–513–16 vol.2.

Bell, R., Koren, Y., and Volinsky, C. (2009). The bellkor solution to the netflix prize.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, **48**, 259–302.

Billsus, D. and Pazzani, M. (1998). Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, volume 54, page 48.

Blei, D. M., Ng, A., and Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research*, **3**, 993–1022.

Crammer, K. and Singer, Y. (2002). On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, **2**, 265–292.

Daumé, H., Langford, J., and Marcu, D. (2009). Search-based structured prediction. *Machine learning*, **75**(3), 297–325.

Daumé III, H. and Marcu, D. (2005). Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176. ACM.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *JASIS*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Grauman, K. and Darrell, T. (2007). The Pyramid Match Kernel: Efficient Learning with Sets of Features. *Journal of Machine Learning Research*, **8**, 725–760.

Herbrich, R., Graepel, T., and Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. *Advances in large margin classifiers*, **88**(2), 115–132.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *SIGIR 1999*, pages 50–57.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD*, pages 133–142. ACM.

Lacoste-Julien, S., Taskar, B., Klein, D., and Jordan, M. (2006). Word alignment via quadratic assignment. In *North American chapter of the Association for Computational Linguistics (NAACL)*.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Le, Q. and Smola, A. (2007). Direct optimization of ranking measures. Technical report, National ICT Australia.

Lee, D. and Seung, H. (2001). Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, **13**.

Leung, T. and Malik, J. (1999). Recognizing surface using three-dimensional textons. *Proc. of 7th Int'l Conf. on Computer Vision, Corfu, Greece*.

Makadia, A., Pavlovic, V., and Kumar, S. (2008). A new baseline for image annotation. In *European conference on Computer Vision (ECCV)*.

Qin, T., Liu, T., Zhang, X., Wang, D., and Li, H. (2008). Global ranking using continuous conditional random fields. In *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS 2008)*.

Raman, K., Shivaswamy, P., and Joachims, T. (2012). Learning to diversify from implicit feedback. In *WSDM Workshop on Diversity in Document Retrieval*.

Rocchio, J. (1971). Relevance feedback in information retrieval.

Schoelkopf, B., Smola, A. J., and Müller, K. R. (1999). Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning (ICML)*, pages 104–112.

Usunier, N., Buffoni, D., and Gallinari, P. (2009). Ranking with ordered weighted pairwise classification. In L. Bottou and M. Littman, editors, *ICML*, pages 1057–1064, Montreal. Omnipress.

Volkovs, M. and Zemel, R. (2009). Boltzrank: learning to maximize expected ranking gain. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1089–1096. ACM.

Weiss, D., Sapp, B., and Taskar, B. (2010). Sidestepping intractable inference with structured ensemble cascades. *Advances in Neural Information Process-*

ing Systems, **23**.

Weston, J. and Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In *Proceedings of the seventh European symposium on artificial neural networks*, volume 4, pages 219–224.

Weston, J., Bengio, S., and Usunier, N. (2011). Large scale image annotation: Learning to rank with joint word-image embeddings. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Weston, J., Bengio, S., and Hamel, P. (2012). Large-scale music annotation and retrieval: Learning to rank in joint semantic spaces. In *Journal of New Music Research*.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., and Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*.

Yue, Y., Finley, T., Radlinski, F., and Joachims, T. (2007a). A support vector method for optimizing average precision. In *SIGIR*, pages 271–278.

Yue, Y., Finley, T., Radlinski, F., and Joachims, T. (2007b). A support vector method for optimizing average precision. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 271–278.

Non-Convex Rank Minimization via an Empirical Bayesian Approach

David Wipf
Visual Computing Group, Microsoft Research Asia
davidwipf@gmail.com

Abstract

In many applications that require matrix solutions of minimal rank, the underlying cost function is non-convex leading to an intractable, NP-hard optimization problem. Consequently, the convex nuclear norm is frequently used as a surrogate penalty term for matrix rank. The problem is that in many practical scenarios there is no longer any guarantee that we can correctly estimate generative low-rank matrices of interest, theoretical special cases notwithstanding. Consequently, this paper proposes an alternative empirical Bayesian procedure build upon a variational approximation that, unlike the nuclear norm, retains the same globally minimizing point estimate as the rank function under many useful constraints. However, locally minimizing solutions are largely smoothed away via marginalization, allowing the algorithm to succeed when standard convex relaxations completely fail. While the proposed methodology is generally applicable to a wide range of low-rank applications, we focus our attention on the robust principal component analysis problem (RPCA), which involves estimating an unknown low-rank matrix with unknown sparse corruptions. Theoretical and empirical evidence are presented to show that our method is potentially superior to related MAP-based approaches, for which the convex principle component pursuit (PCP) algorithm (Candès et al., 2011) can be viewed as a special case.

1 INTRODUCTION

Recently there has been a surge of interest in finding low-rank decompositions of matrix-valued data subject to some problem-specific constraints (Babacan et al., 2011; Candès & Recht, 2008; Candès et al., 2011; Chandrasekaran et al., 2011; Ding et al., 2011). While the methodology proposed herein is applicable to a wide range of low-rank applications, we will focus our attention on the robust principal component analysis problem (RPCA) described in Candès et al. (2011). We begin with the observation model

$$Y = X + S + E, \tag{1}$$

where $Y \in \mathbb{R}^{m\times n}$ is an observed data matrix, X is an unknown low-rank component, S is a sparse corruption matrix, and E is diffuse noise, with iid elements distributed as $N(0, \lambda)$. Without loss of generality, we will assume throughout that $n \geq m$. To estimate X and S given Y, one possibility is to solve

$$\min_{X,S} \frac{1}{\lambda}\|Y - X - S\|_{\mathcal{F}}^2 + n\text{Rank}\,[X] + \|S\|_0, \tag{2}$$

where $\|S\|_0$ denotes the matrix ℓ_0 norm of S, or a count of the nonzero elements in S. The reason for the factor of n is to ensure that the rank and sparsity terms are properly balanced, meaning that both terms range from 0 to nm, which reflects our balanced uncertainty regarding their relative contributions to Y. Unfortunately, solving (2) is problematic because the objective function is both discontinuous and non-convex. In general, the only way to guarantee that the global minimum is found is to conduct an exhaustive combinatorial search, which is intractable in all but the simplest cases.

The most common alternative, sometimes called principle component pursuit (PCP) (Candès et al., 2011), is to replace (2) with a convex surrogate such as

$$\min_{X,S} \frac{1}{\lambda}\|Y - X - S\|_{\mathcal{F}}^2 + \sqrt{n}\|X\|_* + \|S\|_1, \tag{3}$$

where $\|X\|_*$ denotes the nuclear norm of X (or the sum of its singular values). Note that the scale factor from (2) has been changed from n to $\sqrt{n}$; this is an

artifact of the relaxation mechanism balancing the nuclear and ℓ_1 norms.[1] A variety of recent theoretical results stipulate when solutions of (3), particularly in the limiting case where $\lambda \to 0$ (reflecting the assumption that $E = 0$), will produce reliable estimates of X and S (Candès et al., 2011; Chandrasekaran et al., 2011). However, in practice these results have marginal value since they are based upon strong, typically unverifiable assumptions on the support of S and the structure of X. In general, the allowable support of S may be prohibitively small and unstructured (possibly random); related assumptions are required for the rank and structure of X. Thus there potentially remains a sizable gap between what can be achieved by minimizing the 'ideal' cost function (2) and the convex relaxation (3).

In Section 2, as a motivational tool we discuss a simple non-convex scheme based on a variational majorization-minimization approach for locally minimizing (2). Then in Section 3 we reinterpret this method as *maximum a posteriori* (MAP) estimation and use this perspective to design an alternative empirical Bayesian algorithm that avoids the major shortcomings of MAP estimation. Section 4 investigates analytical properties of this empirical Bayesian alternative with respect to globally and locally minimizing solutions. Later in Section 5 we compare a state-of-the-art PCP algorithm with the proposed approach on simulated data as well as a photometric stereo problem.

2 NON-CONVEX MAJORIZATION-MINIMIZATION

One possible alternative to (3) is to replace (2) with a non-convex yet smooth approximation that can at least be locally minimized. In the sparse estimation literature, one common substitution for the ℓ_0 norm is the Gaussian entropy measure $\sum_i \log |s_i|$, which may also sometimes include a small regularizer to avoid taking the log of zero.[2] This can be justified (in part) by the fact that

$$\sum_i \log |s_i| \equiv \lim_{p \to 0} \frac{1}{p} \sum_i \left(|s_i|^p - 1 \right) \propto \|\boldsymbol{s}\|_0. \quad (4)$$

An analogous approximation suggests replacing the rank penalty with $\log |XX^T|$ as has been suggested for related rank minimization problems (Mohan & Fazel, 2010), since $\log |XX^T| = \sum_i \log \sigma_i$, where σ_i are the singular values of XX^T. This leads to the alternative cost function

$$\min_{X,S} \frac{1}{\lambda} \|Y - X - S\|_{\mathcal{F}}^2 + n \log |XX^T| + 2 \sum_{i,j} \log |s_{ij}|. \quad (5)$$

Optimization of (5) can be accomplished by a straightforward majorization-minimization approach based upon variational bounds on the non-convex penalty terms (Jordan et al., 1999). For example, because $\log |s|$ is a concave function of s^2, it can be expressed using duality theory (Boyd & Vandenberghe, 2004) as the minimum of a particular set of upper-bounding lines:

$$\log s^2 = \min_{\gamma \geq 0} \frac{s^2}{\gamma} + \log \gamma - 1. \quad (6)$$

Here γ is a non-negative variational parameter controlling the slope. Therefore, for any fixed γ we have a strict, convex upper-bound on the concave log function. Likewise, for the rank term we can use the analogous representation (Mohan & Fazel, 2010)

$$n \log \left| XX^T \right| = \min_{\Psi \succeq 0} \text{Trace} \left[XX^T \Psi^{-1} \right] + n \log |\Psi| + C, \quad (7)$$

where C is an irrelevant constant and Ψ is a positive semi-definite matrix of variational parameters.[3] Combining these bounds we obtain an equivalent optimization problem

$$\min_{X,S,\Gamma \geq 0, \Psi \succeq 0} \frac{1}{\lambda} \|Y - X - S\|_{\mathcal{F}}^2 + \sum_{ij} \left(\frac{s_{ij}^2}{\gamma_{ij}} + \log \gamma_{ij} \right) + \text{Trace} \left[XX^T \Psi^{-1} \right] + n \log |\Psi|, \quad (8)$$

where Γ is a matrix of non-negative elements composed of the variational parameters corresponding to each s_{ij}. With Γ and Ψ fixed, (8) is quadratic in X and S and can be minimized in closed form via

$$\begin{aligned} \boldsymbol{x}_j &\rightarrow \Psi \left(\Psi + \bar{\Gamma}_j \right)^{-1} \boldsymbol{y}_j, \\ \boldsymbol{s}_j &\rightarrow \bar{\Gamma}_j \left(\Psi + \bar{\Gamma}_j \right)^{-1} \boldsymbol{y}_j, \ \ \forall j \end{aligned} \quad (9)$$

where $\boldsymbol{y}_j$, $\boldsymbol{x}_j$, and $\boldsymbol{s}_j$ represent the j-th columns of Y, X, and S respectively and $\bar{\Gamma}_j$ is a diagonal matrix formed from the j-th column of Γ. Likewise, with X and S fixed, Γ and Ψ can also be obtained in closed form using the updates

$$\begin{aligned} \Psi &\rightarrow \frac{1}{n} XX^T \\ \gamma_{ij} &\rightarrow s_{ij}^2, \ \ \forall i, j. \end{aligned} \quad (10)$$

While local minimization of (5) is clear cut, finding global solutions can still be highly problematic just as

[1] Actually, different scaling factors can be adopted to reflect different assumptions about the relative contributions of low-rank and sparse terms. But we assume throughout that no such knowledge is available.

[2] Algorithms and analysis follow through much the same regardless.

[3] If X is full rank, then Ψ must be positive definite.

before. Whenever any coefficient of S goes to zero, or whenever the rank of X is reduced, we are necessarily at a local minimum with respect to this quantity such that we can never increase the rank or a zero-valued coefficient magnitude in search of the global optimum. (This point will be examined in further detail in Section 4.) Thus the algorithm may quickly converge to one of a combinatorial number of local solutions.

3 VARIATIONAL EMPIRICAL BAYESIAN ALGORITHM

From a Bayesian perspective we can formulate (5) as a MAP estimation problem based on the distributions

$$\begin{aligned} p(Y|X,S) &\propto \exp\left[-\frac{1}{2\lambda}\|Y-X-S\|_{\mathcal{F}}^2\right] \\ p(X) &\propto \frac{1}{|XX^T|^{n/2}} \\ p(S) &\propto \prod_{i,j}\frac{1}{|s_{ij}|}. \end{aligned} \tag{11}$$

It is then transparent that solving

$$\max_{X,S} p(X,S|Y) \equiv \max_{X,S} p(Y|X,S)p(X)p(S) \tag{12}$$

is equivalent to solving (5) after an inconsequential $-2\log(\cdot)$ transformation. But as implied above, this strategy is problematic because the effective posterior is characterized by numerous spurious peaks rendering MAP estimation intractable. A more desirable approach would ignore most of these peaks and focus only on regions with significant posterior mass, regions that hopefully also include the posterior mode. One way to accomplish this involves using the bounds from (6) and (7) to construct a simple approximate posterior that reflects the mass of the original $p(X,S|Y)$ sans spurious peaks. We approach this task as follows.

From (6) and (7)) we can infer that

$$p(S) \propto \max_{\Gamma\succeq 0}\hat{p}(S;\Gamma) \tag{13}$$

$$p(X) \propto \max_{\Psi\succeq 0}\hat{p}(X;\Psi) \tag{14}$$

where

$$\begin{aligned} \hat{p}(S;\Gamma) &\triangleq \exp\left[-\frac{1}{2}\sum_{ij}\left(\frac{s_{ij}^2}{\gamma_{ij}}+\log\gamma_{ij}\right)\right] \\ \hat{p}(X;\Psi) &\triangleq \exp\left[-\frac{1}{2}\text{Trace}\left[XX^T\Psi^{-1}\right]-\frac{n}{2}\log|\Psi|\right], \end{aligned} \tag{15}$$

which can be viewed as unnormalized approximate priors offering strict lower bounds on $p(S)$ and $p(X)$. We also then obtain a tractable posterior approximation given by

$$\hat{p}(X,S|Y;\Gamma,\Psi) \triangleq \frac{p(Y|S,X)\hat{p}(S;\Gamma)\hat{p}(X;\Psi)}{\int p(Y|S,X)\hat{p}(S;\Gamma)\hat{p}(X;\Psi)dSdX}. \tag{16}$$

Here $\hat{p}(X,S|Y;\Gamma,\Psi)$ is a Gaussian distribution with closed-form first and second moments, e.g., the means of S and X are actually given by the righthand sides of (9). The question remains how to choose Γ and Ψ. With the goal of reflecting the mass of the true distribution $p(Y,X,S)$, we adopt the approach from Wipf et al. (2011) and attempt to solve

$$\min_{\Psi,\Gamma}\int |p(Y,X,S)-p(Y|S,X)\hat{p}(S;\Gamma)\hat{p}(X;\Psi)|\,dXdS \tag{17}$$

$$= \min_{\Psi,\Gamma}\int p(Y|S,X)\,|p(X)p(S)-\hat{p}(S;\Gamma)\hat{p}(X;\Psi)|\,dXdS. \tag{18}$$

The basic idea here is that we only care that the approximate priors match the true ones in regions where the likelihood function $p(Y|X,S)$ is significant; in other regions the mismatch is more or less irrelevant. Moreover, by virtue of the strict lower variational bound, (18) reduces to

$$\max_{\Psi,\Gamma}\int p(Y|S,X)\hat{p}(S;\Gamma)\hat{p}(X;\Psi)dXdS \equiv \min_{\Psi,\Gamma}\mathcal{L}(\Psi,\Gamma) \tag{19}$$

where

$$\mathcal{L}(\Psi,\Gamma) \triangleq \sum_{j=1}^{n}\left[\boldsymbol{y}_j^T\Sigma_{y_j}^{-1}\boldsymbol{y}_j+\log\left|\Sigma_{y_j}\right|\right] \tag{20}$$

with

$$\Sigma_{y_j} \triangleq \Psi+\bar{\Gamma}_j+\lambda I. \tag{21}$$

This Σ_{y_j} can be viewed as the covariance of the j-th column of Y given fixed values of Ψ and Γ. To recap then, we need now minimize $\mathcal{L}(\Psi,\Gamma)$ with respect to Ψ and Γ, and then plug these estimates into (16) giving the approximate posterior. The mean of this distribution (see below) can then be used as a point estimate for X and S. This process is sometimes referred to as *empirical Bayes* because we are using the data to guide our search for an optimal prior distribution (Berger, 1985; Tipping, 2001).

3.1 UPDATE RULE DERIVATIONS

It turns out that minimization of (20) can be accomplished concurrently with computation of the posterior mean leading to simple, efficient update rules. While (20) is non-convex, we can use a majorization-minimization approach analogous to that used for MAP estimation. For this purpose, we utilize simplifying upper bounds on both terms of the cost function as

has been done for related sparse estimation problems Wipf & Nagarajan (2010).

First, the data-dependent term is concave with respect to Ψ^{-1} and Γ^{-1} and hence can be expressed as a minimization over (Ψ^{-1}, Γ^{-1})-dependent hyperplanes. With some linear algebra, it can be shown that

$$\boldsymbol{y}_j^T \Sigma_{y_j}^{-1} \boldsymbol{y}_j = \min_{\boldsymbol{x}_j, \boldsymbol{s}_j} \frac{1}{\lambda} \|\boldsymbol{y}_j - \boldsymbol{x}_j - \boldsymbol{s}_j\|_{\mathcal{F}}^2 + \boldsymbol{x}_j^T \Psi^{-1} \boldsymbol{x}_j + \sum_i \frac{s_{ij}^2}{\gamma_{ij}} \tag{22}$$

for all j. With a slight abuse of notation, we adopt $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ and $S = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n]$ as the variational parameters in (22) because they end up playing the same role as the unknown low-rank and sparse coefficients and provide a direct link to the MAP estimates. Additionally, the $\boldsymbol{x}_j$ and $\boldsymbol{s}_j$ which minimize (22) turn out to be equivalent to the posterior means of (16) given Ψ and Γ and will serve as our point estimates.

Secondly, for the log-det term, we first use the determinant identity

$$\log \left| \Psi + \bar{\Gamma}_j + \lambda I \right| = \log |\Psi| + \log \left| \bar{\Gamma}_j \right| + \log |A_j| + C, \tag{23}$$

where

$$A_j \triangleq \lambda^{-1} \begin{bmatrix} I & I \\ I & I \end{bmatrix} + \begin{bmatrix} \Psi^{-1} & \mathbf{0} \\ \mathbf{0} & \bar{\Gamma}_j^{-1} \end{bmatrix} \tag{24}$$

and C is an irrelevant constant. The term $\log |A_j|$ is jointly concave in both Ψ^{-1} and $\bar{\Gamma}_j^{-1}$ and thus can be bounded in a similar fashion as (22), although a closed-form solution is no longer available. (Other decompositions lead to different bounds and different candidate update rules.) Here we use

$$\log |A_j| = \min_{U_j, V_j \succeq 0} \text{Trace} \left[U_j^T \Psi^{-1} + V_j^T \bar{\Gamma}_j^{-1} \right] - h^*(U_j, V_j) \tag{25}$$

where $h^*(U_j, V_j)$ is the concave conjugate function of $\log |A_j|$ with respect to Ψ^{-1} and $\bar{\Gamma}_i^{-1}$. Note that while $h^*(U_j, V_j)$ has no closed-form solution, the minimizing values of U_j and V_j can be computed in closed-form via

$$U_j = \frac{\partial \log |A_j|}{\partial \Psi^{-1}}, \qquad V_j = \frac{\partial \log |A_j|}{\partial \bar{\Gamma}_j^{-1}}. \tag{26}$$

When we drop the minimizations over the variational parameters $\boldsymbol{x}_j$, $\boldsymbol{s}_j$, U_j, and V_j for all j, we arrive at a convenient family of upper bounds on the cost function $\mathcal{L}(\Psi, \Gamma)$. Given some estimate of Ψ and Γ, we can evaluate all variational parameters in closed form (see below). Likewise, given all of the variational parameters we can solve directly for Ψ and Γ because now $\mathcal{L}(\Psi, \Gamma)$ has been conveniently decoupled and we need only compute

$$\min_{\Psi \succeq 0} \sum_j \left(\boldsymbol{x}_j^T \Psi^{-1} \boldsymbol{x}_j + \text{Trace} \left[U_j^T \Psi^{-1} \right] \right) + n \log |\Psi| \tag{27}$$

and

$$\min_{\gamma_{ij} \geq 0} \frac{s_{ij}^2}{\gamma_{ij}} + \frac{[V_j]_{ii}}{\gamma_{ij}} + \log \gamma_{ij}, \quad \forall i, j. \tag{28}$$

We summarize the overall procedure next.

3.2 ALGORITHM SUMMARY

1. Compute $\kappa \triangleq \frac{1}{nm} \|Y\|_{\mathcal{F}}^2$
2. Initialize $\Psi^{(0)} \rightarrow \kappa I$, and $\bar{\Gamma}_j^{(0)} \rightarrow \kappa I$ for all j.
3. For the $(k+1)$-th iteration, compute the optimal $\boldsymbol{x}_j$ and $\boldsymbol{s}_j$ via

$$\boldsymbol{x}_j^{(k+1)} \rightarrow \Psi^{(k)} \left(\Psi^{(k)} + \bar{\Gamma}_j^{(k)} + \lambda I \right)^{-1} \boldsymbol{y}_j$$

$$\boldsymbol{s}_j^{(k+1)} \rightarrow \bar{\Gamma}^{(k)} \left(\Psi^{(k)} + \bar{\Gamma}_j^{(k)} + \lambda I \right)^{-1} \boldsymbol{y}_j \tag{29}$$

4. Likewise, compute the optimal U_j and V_j via

$$U_j^{(k+1)} \rightarrow \Psi^{(k)} - \Psi^{(k)} \left(\Psi^{(k)} + \bar{\Gamma}_j^{(k)} + \lambda I \right)^{-1} \Psi^{(k)}$$

$$V_j^{(k+1)} \rightarrow \bar{\Gamma}_j^{(k)} - \bar{\Gamma}_j^{(k)} \left(\Psi^{(k)} + \bar{\Gamma}_j^{(k)} + \lambda I \right)^{-1} \bar{\Gamma}_j^{(k)} \tag{30}$$

5. Update Ψ and Γ using the new variational parameters via

$$\Psi^{(k+1)} \rightarrow \frac{1}{n} \sum_j \left[\boldsymbol{x}_j^{(k+1)} \left(\boldsymbol{x}_j^{(k+1)} \right)^T + U_j^{(k+1)} \right]$$

$$\gamma_{ij}^{(k+1)} \rightarrow \left(s_{ij}^{(k+1)} \right)^2 + \left[V_j^{(k+1)} \right]_{ii}, \forall i, j \tag{31}$$

6. Repeat steps 3 through 5 until convergence. (Recall that $\bar{\Gamma}_j$ is a diagonal matrix formed from the j-th column of Γ.) This process is guaranteed to reduce or leave unchanged the cost function at each iteration.

Note that if we set $U_j^{(k+1)}, V_j^{(k+1)} \rightarrow 0$ for all j, then the algorithm above is guaranteed to (at least locally) minimize the MAP cost function from (5). Additionally, for matrix completion problems (Candès & Recht, 2008), where the support of the sparse errors is known a priori, we need only set each γ_{ij} corresponding to a corrupted entry to ∞. This limiting case can easily be handled with efficient reduced rank updates.

One positive aspect of this algorithm is that it is largely parameter free. We must of course choose some stopping criteria, such as a maximum number of iterations or a convergence tolerance. (For all experiments in Section 5 we simply set the maximum number of iterations at 100.) We must also choose some value for λ, which balances allowable contributions from a diffuse error matrix E, although frequently methods have some version of this parameter, including the PCP algorithm. For all of our experiments we simply choose $\lambda = 10^{-6}$ since we did not include an E component consistent with the original RPCA formulation from Candès et al. (2011).

From a complexity standpoint, each iteration of the above algorithm can be computed in $O(m^3 n)$, where $n \geq m$, so it is linear in the larger dimension of Y and cubic in the smaller dimension. For many computer vision applications (see Section 5 for one example), images are vectorized and then stacked, so Y may be $m =$number-of-images by $n =$ number-of-pixels. This is relatively efficient, since the number of images may be on the order of 100 or fewer (see Wu et al. (2010)). However, when Y is a large square matrix, the updates are more expensive to compute. In the future we plan to investigate various approximation techniques to handle this scenario.

As a final implementation-related point, when given access to a priori knowledge regarding the rank of X and/or sparsity of S, it is possible to bias the algorithm's initialization (from Step 1 above) and improve the estimation accuracy. However, we emphasize that for all of the experiments reported in Section 5 we assumed no such knowledge.

3.3 Alternative Bayesian Methods

Two other Bayesian-inspired methods have recently been proposed for solving the RPCA problem. The first from Ding et al. (2011) is a hierarchical model with conjugate prior densities on model parameters at each level such that inference can be performed using a Gibbs sampler. This method is useful in that the λ parameter balancing the contribution from diffuse errors E is estimated directly from the data. Moreover, the authors report significant improvement over PCP on example problems. A potential downside of this model is that theoretical analysis is difficult because of the underlying complexity. Additionally, a large number of MCMC steps are required to obtain good estimates leading to a significant computational cost even when Y is small. It also uses an estimate of Rank$[X]$ which can effect the convergence rate of the Gibbs sampler.

A second method from Babacan et al. (2011) similarly employs a hierarchial Bayesian model but uses a factorized mean-field variational approximation for inference (Attias, 2000). Note that this is an entirely different type of variational method than ours, relying on a posterior distribution that factorizes over X and S, meaning $p(X, S|Y) \approx q(X|Y)q(S|Y)$, where $q(X|Y)$ and $q(S|Y)$ are approximating distributions learned by minimizing a free energy-based cost function.[4] Unlike our model, this factorization implicitly decouples X and S in a manner akin to MAP estimation, and may potentially produce more locally minimizing solutions (see analysis below). Moreover, while this approach also has a mechanism for estimating λ, there is no comprehensive evidence given that it can robustly expand upon the range of corruptions and rank that can already be handled by PCP.

To summarize both of these methods then, we would argue that while they offer a compelling avenue for computing λ automatically, the underlying cost functions are substantially more complex than PCP or our method rendering more formal analyses somewhat difficult. As we shall see in Sections 4 and 5, the empirical Bayesian cost function we propose is analytically principled and advantageous, and empirically outperforms PCP by a wide margin.

4 ANALYSIS

In this section we will examine global and local minima properties of the proposed method and highlight potential advantages over MAP, of which PCP can also be interpreted as a special case. For analysis purposes and comparisons with MAP estimation, it is helpful to convert the empirical Bayes cost function (20) into (X, S)-space by first optimizing over U_j, V_j, Ψ and Γ, leaving only the unknown coefficient matrices X and S. Using this process, it is easily shown that the estimates of X and S obtained by globally (or locally) minimizing (20) will also globally (or locally) minimize

$$\min_{X,S} \|Y - X - S\|_{\mathcal{F}}^2 + \lambda g_{EB}(X, S; \lambda), \qquad (32)$$

where the penalty function is given by

$$g_{EB}(X, S; \lambda) \triangleq \qquad (33)$$

$$\min_{\Gamma \geq 0, \Psi \succeq 0} \sum_{i=1}^{n} \boldsymbol{x}_j \Psi^{-1} \boldsymbol{x}_j + \boldsymbol{s}_j^T \bar{\Gamma}_j^{-1} \boldsymbol{s}_j + \log \left| \Psi + \bar{\Gamma}_j + \lambda I \right|.$$

Note that the implicit MAP penalty from (5) is nearly identical:

$$g_{map}(X, S) \triangleq \qquad (34)$$

[4] Additional factorizations are also included in the model.

$$\min_{\Gamma\succeq 0,\Psi\succeq 0}\sum_{i=1}^{n}\boldsymbol{x}_j\Psi^{-1}\boldsymbol{x}_j+\boldsymbol{s}_j^T\bar{\Gamma}_j^{-1}\boldsymbol{s}_j+\log|\Psi|+\log\left|\bar{\Gamma}_j\right|.$$

The primary distinction is that in the MAP case the variational parameters separate whereas in empirical Bayesian case they do not. (Note that, as discussed below, we can apply a small regularizer analogous to λ to the log terms in the MAP case as well.) This implies that $g_{map}(X,S)$ can be expressed as some $g_{map}(X)+g_{map}(S)$ whereas $g_{EB}(X,S;\lambda)$ cannot. A related form of non-separability has been shown to be advantageous in the context of sparse estimation from overcomplete dictionaries (Wipf et al., 2011).

We now examine how this crucial distinction can be beneficial in producing maximally sparse, low-rank solutions that optimize (2). We first demonstrate how (32) mimics the global minima profile of (2). Later we show how the smoothing mechanism of the empirical Bayesian marginalization can mitigate spurious locally minimizing solutions.

The original RPCA development from Candès et al. (2011) assumes that $E=0$, which is somewhat easier to analyze. We consider this scenario first.

Theorem 1. Assume that there exists at least one solution to $Y=X+S$ such that $\text{Rank}[X]+\max_j\|\boldsymbol{s}_j\|_0<m$. Then in the limit as $\lambda\to 0$, any solution that globally minimizes (32) will globally minimize (2).

Proofs will be deferred to a subsequent journal publication. Note that the requirement $\text{Rank}[X]+\max_j\|\boldsymbol{s}_j\|_0<m$ is a relatively benign assumption, because without it the matrices X and S are formally unidentifiable even if we are able to globally solve (2). For $E>0$, we may still draw direct comparisons between (32) and (2) when we deviate slightly from the Bayesian development and treat $g_{EB}(X,S;\lambda)$ as an abstract, stand-alone penalty function. In this context we may consider $g_{EB}(X,S;\alpha)$, with $\alpha\neq\lambda$ as a more general candidate for estimating RPCA solutions.

Corollary 1. Assume that $X_{(\lambda)}$ and $S_{(\lambda)}$ are a unique, optimal solution to (2) and that $\text{Rank}\left[X_{(\lambda)}\right]+\max_j\|\left[\boldsymbol{s}_{(\lambda)}\right]_j\|_0<m$. Then there will always exist some λ' and α' such that the global minimum of

$$\min_{X,S}\|Y-X-S\|_{\mathcal{F}}^2+\lambda' g_{EB}(X,S;\alpha'), \tag{35}$$

denoted $X_{(\lambda',\alpha')}$ and $S_{(\lambda',\alpha')}$, satisfies the conditions $\left\|X_{(\lambda',\alpha')}-X_{(\lambda)}\right\|<\epsilon$ and $\left\|S_{(\lambda',\alpha')}-S_{(\lambda)}\right\|<\epsilon$, where ϵ can be arbitrarily small.

Of course MAP estimation can satisfy a similar property as Theorem 1 and Corollary 1 after a minor modification. Specifically, we may define

$$g_{map}(X,S;\alpha)\triangleq \tag{36}$$

$$\min_{\Gamma\succeq 0,\Psi\succeq 0}\sum_{j=1}^{n}\boldsymbol{x}_j\Psi^{-1}\boldsymbol{x}_j+\boldsymbol{s}_j^T\bar{\Gamma}_j^{-1}\boldsymbol{s}_j+\log|\Psi+\alpha|+\log\left|\bar{\Gamma}_j+\alpha\right|$$

and then achieve a comparable result to the above using $g_{map}(X,S;\alpha')$. The advantage of empirical Bayes then is not with respect to global minima, but rather with respect to local minima. The separable, additive low-rank plus sparsity penalties that emerge from MAP estimation will always suffer from the following limitation:

Theorem 2. Let $S_{ij}^{(a)}$ denote any matrix S with $s_{ij}=a$. Now consider any optimization problem of the form

$$\min_{X,S}g_1(X)+g_2(S),\quad\text{s.t. } Y=X+S, \tag{37}$$

where g_1 is an arbitrary function of the singular values of X and g_2 is an arbitrary function of the magnitudes of the elements in S. Then to ensure that a global minimum of (37) is a global minimum of (2) for all possible Y, we require that

$$\lim_{\epsilon\to 0}\frac{g_2\left[S_{ij}^{(\epsilon)}\right]-g_2\left[S_{ij}^{(0)}\right]}{\epsilon}=\infty \tag{38}$$

for all i and j and S. An analogous condition holds for the function g_1.

This result implies that whenever an element of S approaches zero, it will require increasing the associated penalty $g_2(S)$ against an arbitrarily large gradient to escape in cases where this coefficient was incorrectly pruned. Likewise, if the rank of X is prematurely reduced in the wrong subspace, there may be no chance to ever recover since this could require increasing $g_1(X)$ against an arbitrarily large gradient factor. In general, Theorem 2 stipulates that if we would like to retain the same global minimum as (2) using a MAP estimation-based cost function, then we will necessarily enter an inescapable basin of attraction whenever *either* $\text{Rank}[X]<m$ *or* $\|\boldsymbol{s}_j\|_0<m$ for some j. This is indeed a heavy price to pay.

Crucially, because of the coupling of low-rank and sparsity regularizers, the penalty function $g_{EB}(X,S;\lambda)$ does not have this limitation. In fact, we only encounter insurmountable gradient barriers when $\text{Rank}[X]+\|\boldsymbol{s}_j\|_0<m$ for some j, in which case the covariance Σ_{y_j} from (21) becomes degenerate (with λ small), a much weaker condition. To summarize (emphasize) this point then, MAP can be viewed as heavily dependent on degeneracy of the matrices Ψ and Γ

in isolation, whereas empirical Bayes is only sensitive to degeneracy of their summation.

This distinction can also be observed in how the effective penalties on X and S, as imbedded in $g_{EB}(X, S; \lambda)$, vary given fixed values of Γ or Ψ respectively. For example, when Ψ is close to being full rank and orthogonal (such as when the algorithm is initialized), then the implicit penalty on S is minimally non-convex (only slightly concave). In fact, as Ψ becomes large and orthogonal, the penalty converges to a scaled version of the ℓ_1 norm. In contrast, as Ψ becomes smaller and low-rank, the penalty approaches a scaled version of the ℓ_0 norm, implying that maximally sparse corruptions will be favored. Thus, we do not aggressively favor maximally sparse S until the rank has already been reduced and we are in the basin of attraction of a good solution. Of course no heuristic annealing strategy is necessary, the transition is handled automatically by the algorithm.

Additionally, whenever Ψ is fixed, the resulting cost function formally decouples into n separate, canonical sparse estimation problems on each $\boldsymbol{s}_j$ in isolation. With $\lambda = 0$, it not difficult to show that each of these subproblems is equivalent to solving

$$\min_{\boldsymbol{s}_j} \|\boldsymbol{y}_j - \Phi \boldsymbol{s}_j\|_2^2 + g_{EB}(\boldsymbol{s}_j) \tag{39}$$

where

$$g_{EB}(\boldsymbol{s}_j) \triangleq \min_{\boldsymbol{\gamma}_j \geq 0} \sum_{j=1}^{n} \boldsymbol{s}_j^T \bar{\Gamma}_j^{-1} \boldsymbol{s}_j + \log \left| \Phi \bar{\Gamma}_j \Phi^T + I \right| \tag{40}$$

is a concave sparsity penalty on $\boldsymbol{s}_j$ and Φ is any matrix such that $\Phi \Psi \Phi^T = I$.[5] When Φ is nearly orthogonal, this problem has no local minima and a global solution that approximates the hard thresholding of the ℓ_0 norm; however, direct minimization of the ℓ_0 norm will have 2^n local minima (Wipf et al., 2011). In contrast, when Φ is poorly conditioned (with approximately low-rank structure, it has been argued in Wipf et al. (2011) that penalties such as $g_{EB}(\boldsymbol{s}_j)$ are particularly appropriate for avoiding local minima.

Something similar occurs when Γ is now fixed and we evaluate the penalty on X. This penalty approaches something like a scaled version of the nuclear norm (less concave) when elements of Γ are set to a large constant and it behaves more like the rank function when Γ is small. At initialization, when Γ is all ones, we are relatively free to move between solutions of various rank without incurring a heavy penalty. Later as Γ becomes sparse, solutions satisfying $\text{Rank}[X] + \|\boldsymbol{s}_j\|_0 < m$ for some j become heavily favored.

[5] We have assumed here that Ψ is full rank.

As a final point, the proposed empirical Bayesian approach can be implemented with alternative variational bounds and possibly optimized with something akin to simultaneous reweighted nuclear and ℓ_1 norm minimization, a perspective that naturally suggests further performance analyses such as those applied to sparse estimation in Wipf & Nagarajan (2010).

5 EMPIRICAL RESULTS

This section provides some empirical evidence for the efficacy of our RPCA method. First, we present comparisons with PCP recovering random subspaces from corrupted measurements. Later we discuss a photometric stereo application. In all cases we used the the augmented lagrangian method (ALM) from Lin et al. (2010) to implement PCP. This algorithm has efficient, guaranteed convergence and in previous empirical tests ALM has outperformed a variety of other methods in computing minimum nuclear norm plus ℓ_1 norm solutions.

5.1 RANDOM SUBSPACE SIMULATIONS

Here we demonstrate that the empirical Bayesian algorithm from Section 3.2, which we will refer to as EB, can recovery unknown subspaces from corrupted measurements in a much broader range of operating conditions compared to the convex PCP. In particular, for a given value of $\text{Rank}[X]$, our method can handle a substantially larger fraction of corruptions as measured by $\rho = \|S\|_0/(nm)$. Likewise, for a given value of ρ, we can accurately estimate an X with much higher rank. Consistent with Candès et al. (2011), we consider the case where $E = 0$, such that all the error is modeled by S. This allows us to use the stable, convergent ALM code available online.[6]

The first experiment proceeds as follows. We generate a low-rank matrix X with dimensions reflective of many computer vision problems: *number-of-images* $\times$ *number-of-pixels*. Here we choose $m = 20$ and $n = 10^4$, the later dimension equivalent to a 100×100 pixel image. For each trial, we compute an $m \times n$ matrix with iid $\mathcal{N}(0, 1)$ entries. We then compute the SVD of this matrix and set all but the r largest singular values to zero to produce a low-rank X. S is generated with nonzero entries selected uniformly with probability $\rho = 0.2$. Nonzero values are sampled from an iid Uniform[-10,10] distribution. We then compute $Y = X + S$ and try to estimate X and S using the EB and PCP algorithms. Estimation results averaged over multiple trials as r is varied from 1 to 10 are depicted in Figure 1. We plot normalized mean-squared error

[6] `http://perception.csl.uiuc.edu/matrix-rank/`

(MSE) as computed via $\left\langle \|X - \hat{X}\|_{\mathcal{F}}^2 / \|X\|_{\mathcal{F}}^2 \right\rangle$ as well as the average angular error between the estimated and true subspaces. In both cases the average is across 10 trials.

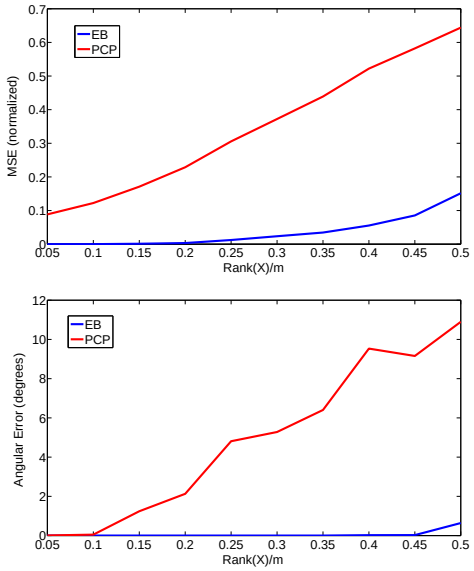


Figure 1: Estimation results where the corruption probability ρ is 0.2 and Rank$[X]/m$ is varied from 0.05 to 0.50, meaning up to 50% of full rank. *Top*: Normalized mean-squared error (MSE). *Bottom*: Average angle (in degrees) between the estimated and true subspaces.

From Figure 1 we observe that EB can accurately estimate X for substantially higher values of the rank. Interestingly, we are also still able to estimate the correct subspace spanned by columns of X perfectly even when the MSE of estimating X starts to rise (compare Figure 1(*Top*) with Figure 1(*Bottom*)). Basically, this occurs because, even if we have estimated the subspace perfectly, reducing the MSE to zero implicitly requires solving a challenging sparse estimation problem for every observation column $\boldsymbol{y}_j$. For each column, this problem requires learning $d_j \triangleq \text{Rank}[X] + \|\boldsymbol{s}_j\|_0$ nonzero entries given only $m = 20$ observations. For our experiment, we can have $d_j > 14$ with high probability for some columns when the rank is high, and thus we may expect some errors in $\hat{S}$ (not shown). However, the encouraging evidence here is that EB is able to keep these corrupting errors at a minimum and estimate the subspace accurately long after PCP has failed. Moreover, if an accurate estimate of X is needed, as opposed to just the correct spanning subspace, then a postprocessing error correction step can potentially be applied to each column individually to improve performance.

The second experiment is similar to the first only now we hold Rank$[X]$ fixed at 4, meaning Rank$[X]/m = 0.2$, and vary the fraction of corrupted entries in S from 0.1 to 0.8. Figure 2 shows that EB is again able to drastically expand the range whereby successful estimates are obtained. Notably it is able to recover the correct subspace even with 70% corrupted entries.

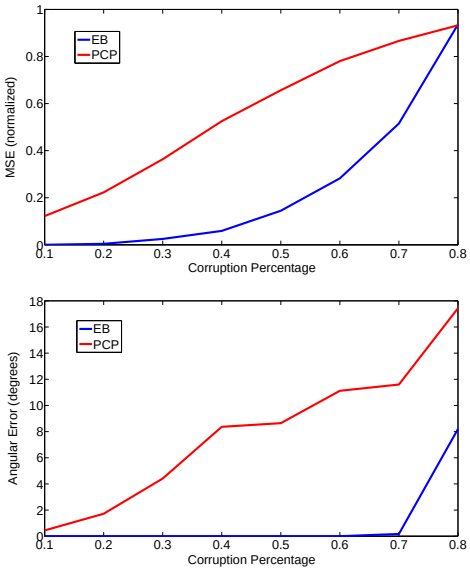


Figure 2: Estimation results with Rank$[X]/m = 0.2$ and ρ varied from 0.1 to 0.8. *Top*: Normalized mean-squared error (MSE). *Bottom*: Average angle (in degrees) between the estimated and true subspaces.

As a final comparison, we tested PCP and EB on a 400×400 observation matrix Y generated as above with a Rank$[X]/m = 0.1$ and $\rho = 0.5$. The estimation results are reported in Table 1. PCP performs poorly since the normalized MSE is above one, meaning we would have been better off simply choosing $\hat{X} = 0$ in this regard. Additionally, the angular error is very near 90 degrees, consistent with the error from a randomly chosen subspace in high dimensions. In contrast, EB provides a reasonably good estimate considering the difficulty of the problem.

Table 1: Estimation results with $m = n = 400$, Rank$[X]/m = 0.1$ and $\rho = 0.5$.

	PCP	**EB**
MSE (norm.)	1.235	0.066
Angular Error	88.50	5.01

5.2 PHOTOMETRIC STEREO

Photometric stereo is a method for estimating surface normals of an object or scene by capturing multiple images from a fixed viewpoint under different lighting conditions (Woodham, 1980). At a basic level, this methodology assumes a Lambertian object surface, point light sources at infinity, an orthographic camera view, and a linear sensor response function. Under these conditions, it has been shown that the intensities of a vectorized stack of images Y can be expressed as

$$Y = L^T N \Upsilon, \tag{41}$$

where L is a $3 \times m$ matrix of m normalized lighting directions, N is a $3 \times n$ matrix of surface normals at n pixel locations, and Υ is a diagonal matrix of diffuse albedo values (Woodham, 1980). Thus, if we were to capture at least 3 images with known, linearly independent lighting directions we can solve for N using least squares. Of course in reality many common non-Lambertian effects can disrupt this process, such as specularities, cast or attached shadows, and image noise, etc. In many cases, these effects can be modeled as an additive sparse error term S applied to (41).

As proposed in Wu et al. (2010), we can estimate the subspace containing N by solving (2) assuming $X = L^T N \Upsilon$ and $E = 0$. The resulting $\hat{X}$, combined with possibly other *a priori* information regarding the lighting directions L can lead to an estimate of N. Wu et al. (2010) propose using a modified version of PCP for this task, where a shadow mask is included to simplify the sparse error correction problem. However, in practical situations it may not always be possible to accurately locate all shadow regions in this manner so it is desirable to treat them as unknown sparse corruptions.

For this experiment we consider the synthetic Caesar image from the INRIA 3D Meshes Research Database[7] with known surface normals. Multiple 2D images with different known lighting conditions can easily be generated using the Cook-Torrance reflectance model (Cook & Torrance, 1981). These images are then stacked to produce Y. Because shadows are extremely difficult to handle in general, as a preprocessing step we remove rows of Y corresponding to pixel locations with more than 10% shadow coverage. Specular corruptions were left unfiltered. We tested our algorithm as the number of images, drawn randomly from a batch of 40 total, was varied from 10 to 40. Results averaged across 5 trials are presented in Figure 3. The error metrics have been redefined to accommodate the photometric stereo problem. We now define the normalized MSE as $\left\langle \|X - \hat{X}\|_{\mathcal{F}}^2 / \|X - Y\|_{\mathcal{F}}^2 \right\rangle$, which measures how much improvement we obtain beyond just using the observation matrix Y directly. Similarly we normalized the angular error by dividing by the angle between Y and the true X for each trial.

[7] http://www-roc.inria.fr/gamma/gamma/download/download.php

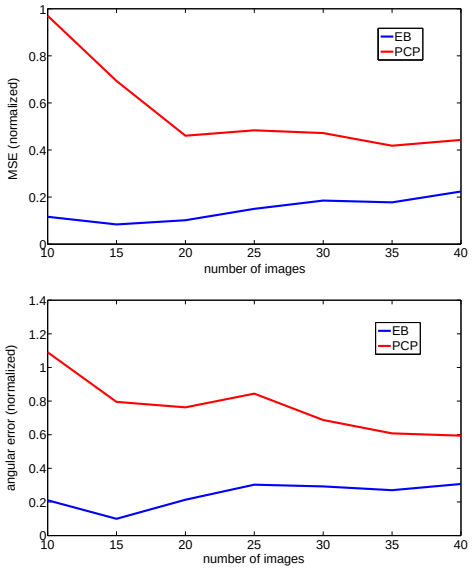


Figure 3: Estimation results using photometric stereo data as the number of images m is varied from 10 to 40. *Top*: Normalized MSE (see altered definition in the main text). *Bottom*: Normalized angle between the estimated and the true subspaces.

From Figure 3 it is clear that EB outperforms PCP in both MSE and angular error, especially when there are fewer images present. It is not entirely clear however why the MSE and angular error are relatively flat for EB as opposed to dropping lower as m increases. Of course these are errors relative to using Y directly to predict X, which could play a role in this counterintuitive effect.

6 CONCLUSIONS

In this paper we have analyzed a new empirical Bayesian approach for matrix rank minimization in the context of RPCA, where the goal is to decompose a given data matrix into low-rank and sparse components. Using a variational approximation and subsequent marginalization, we ultimately arrive at a novel regularization term that couples low-rank and sparsity penalties in such a way that locally minimizing solutions are effectively smoothed while the global optimum matches that of the ideal RPCA cost function.

References

Attias, H. A variational Bayesian framework for graphical models. *Advances in Neural Information Processing Systems 12*, 2000.

Babacan, S.D., Luessi, M., Molina, R., and Katsaggelos, A.K. Sparse Bayesian methods for low-rank matrix estimation. *IEEE Trans. Signal Processing (submitted)*, 2011.

Berger, J.O. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York, 2nd edition, 1985.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Candès, E. and Recht, B. Exact matrix completion via convex optimization. *Found. of Comput. Math.*, 9, 2008.

Candès, E., Li, X., Ma, Y., and Wright, J. Robust principal component analysis? *J. ACM*, 58(2), May 2011.

Chandrasekaran, V., Sanghavi, S., Parrilo, P.A., and Willsky, A.S. Rank-sparsity incoherence for matrix decomposition. *SIAM J. Optimization*, 21(3), June 2011.

Cook, R.L. and Torrance, K.E. A reflectance model for computer graphics. *SIGGRAPH Comput. Graph.*, 15, 1981.

Ding, X., He, L., and Carin, L. Bayesian robust principal component analysis. *IEEE Trans. Image Processing*, 20(12), Dec. 2011.

Jordan, M.I., Ghahramani, Z., Jaakkola, T., and Saul, L.K. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

Lin, Z., Chen, M., and Ma, Y. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *UIUC Technical Report UILU-ENG-09-2214*, Oct 2010.

Mohan, K. and Fazel, M. Iterative reweighted least squares for matrix rank minimization. *Proc. Allerton Conference on Communications, Control, and Computing*, Sept 2010.

Tipping, M.E. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

Wipf, D.P. and Nagarajan, S. Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions. *Journal of Selected Topics in Signal Processing (Special Issue on Compressive Sensing)*, 4(2), April 2010.

Wipf, D.P., Rao, B.D., and Nagarajan, S. Latent variable Bayesian models for promoting sparsity. *IEEE Trans. Information Theory*, 57(9), Sept. 2011.

Woodham, R.J. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1), 1980.

Wu, L., Ganesh, A., Shi, B., Matsushita, Y., Wang, Y., and Ma, Y. Robust photometric stereo via low-rank matrix completion and recovery. *Proc. Asian Conference on Computer Vision*, 2010.

An Improved Admissible Heuristic for Learning Optimal Bayesian Networks

Changhe Yuan[1,3] **and Brandon Malone**[2,3]
[1]Queens College/City University of New York, [2]Helsinki Institute for Information Technology,
[3]and Mississippi State University
changhe.yuan@qc.cuny.edu, bmmalone@gmail.com

Abstract

Recently two search algorithms, A* and breadth-first branch and bound (BFBnB), were developed based on a simple admissible heuristic for learning Bayesian network structures that optimize a scoring function. The heuristic represents a relaxation of the learning problem such that each variable chooses optimal parents independently. As a result, the heuristic may contain many directed cycles and result in a loose bound. This paper introduces an improved admissible heuristic that tries to avoid directed cycles within small groups of variables. A sparse representation is also introduced to store only the unique optimal parent choices. Empirical results show that the new techniques significantly improved the efficiency and scalability of A* and BFBnB on most of datasets tested in this paper.

1 Introduction

Bayesian networks are often used to represent relationships among variables in a domain. When the network structure is unknown, we can learn the structure directly from a given dataset. Several exact algorithms for learning optimal Bayesian networks have been developed based on dynamic programming (Koivisto and Sood 2004; Ott, Imoto, and Miyano 2004; Silander and Myllymaki 2006; Singh and Moore 2005; Parviainen and Koivisto 2009; Malone, Yuan, and Hansen 2011), branch and bound (de Campos and Ji 2011), and linear and integer programming (Cussens 2011; Jaakkola et al. 2010).

Recently, Yuan *et al.* (2011) proposed a shortest-path finding formulation for the Bayesian network structure learning problem, in which the shortest path found in an implicit search graph called order graph corresponds to an optimal network structure. An A* search algorithm was developed to solve the search problem. Malone *et al.* (2011) adopted the same formulation, but realized that the search can be performed in a layered, breadth-first order. External memory and delayed duplicate detection are used to ensure completion regardless of the amount of available RAM. This algorithm, named breadth-first branch and bound (BFBnB), was shown to have similar runtimes as A* but scale to many more variables.

A simple admissible heuristic was used in the A* and BFBnB algorithms (Yuan, Malone, and Wu 2011; Malone et al. 2011) to guide the search. Its main idea is to relax the acyclicity constraint of Bayesian networks such that each variable can freely choose optimal parents from all the other variables. The heuristic provides an optimistic estimation on how good a solution can be and, hence, is admissible. However, the simple relaxation behind the heuristic may introduce many directed cycles and result in a loose bound.

This paper introduces a much improved admissible heuristic named *k-cycle conflict heuristic* based on the additive pattern database technique (Felner, Korf, and Hanan 2004). The main idea is to avoid directed cycles within small groups of variables, called patterns, and compute heuristic values by concatenating patterns. Also, a set of exponential-size parent graphs were created by the A* and BFBnB algorithms to retrieve optimal parent choices during the search. We introduce a sparse representation for storing only unique optimal parent sets which can improve both time and space efficiency of the search. Empirical results show that the new techniques significantly improved the efficiency and scalability of A* and BFBnB on most of the benchmark datasets we tested.

The remainder of this paper is structured as follows. Section 2 provides an overview of Bayesian network structure learning and the shortest-path finding formulation of the problem. Section 3 introduces the improved heuristic. Section 4 introduces the sparse representation of optimal parent choices and discusses how to adapt A* and BFBnB algorithms to use the new techniques. Section 5 reports the empirical results on a set of benchmark datasets. Finally, Section 6 concludes the paper with some remarks.

2 Background

This section reviews the basics of score-based methods for learning Bayesian network structures.

2.1 Learning Bayesian network structures

A Bayesian network is a directed acyclic graph (DAG) in which the vertices correspond to a set of random variables $\mathbf{V} = \{X_1, ..., X_n\}$, and the arcs and lack of them represent dependence and conditional independence relations between the variables. The relations are further quantified using a set of conditional probability distributions. We consider the problem of learning a network structure from a dataset $\mathbf{D} = \{D_1, ..., D_N\}$, where D_i is an instantiation of all the variables in $\mathbf{V}$. A scoring function can be used to measure the goodness of fit of a network structure to $\mathbf{D}$. For example, the minimum description length (MDL) scoring function (Rissanen 1978) uses one term to reward structures with low entropy and another to penalize complex structures. The task is to find an optimal structure that minimizes the MDL score. MDL is decomposable (Heckerman 1998), i.e., the score for a structure is simply the sum of the scores for each variable. All algorithms we describe here assume the scoring function is decomposable. The remainder of the paper assumes the use of MDL score, but our method is equally applicable to other decomposable scoring functions, such as AIC, BIC or BDe.

2.2 The shortest-path finding formulation

Yuan *et al.* (2011) formulated the above structure learning problem as a shortest-path finding problem. Figure 1 shows the *implicit* search graph for four variables. The top-most node with the empty set is the *start* search node, and the bottom-most node with the complete set is the *goal* node. An arc from $\mathbf{U}$ to $\mathbf{U} \cup \{X\}$ in the graph represents generating a successor node by adding a new variable $\{X\}$ to the existing variables $\mathbf{U}$; the cost of the arc is equal to the cost of selecting the optimal parent set for X out of $\mathbf{U}$, and is computed by considering all subsets of $\mathbf{U}$, i.e.,

$$BestScore(X, \mathbf{U}) = \min_{PA_X \subseteq \mathbf{U}} score(X|PA_X).$$

With the search graph thus specified, each path from the start node to the goal is an ordering of the variables in the order of their appearance. That is why the search graph is also called an *order graph*. Because each variable only selects optimal parents from the preceding variables, putting together all the optimal parent choices of a particular ordering generates a valid Bayesian network that is optimal for that specific ordering. The shortest path among all possible paths corresponds to a global optimal Bayesian network.

During the search of the order graph, we need to compute the cost for each arc being visited. We use another data

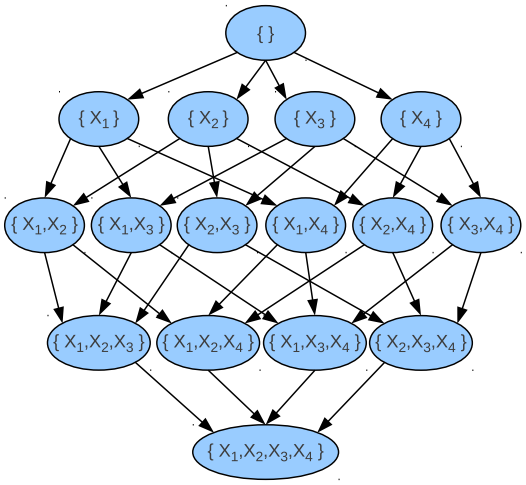


Figure 1: An order graph of four variables

structure called *parent graph* for retrieving the costs. The parent graph for variable X consists of all subsets of $\mathbf{V} \setminus \{X\}$. Figure 2 shows the parent graph for X_1. Figure 2(a) shows a parent graph containing the raw scores for using each subset as the parent set of X_1, while Figure 2(b) shows the optimal scores after propagating the best scores from top to bottom in the graph. For the arc from $\mathbf{U}$ to $\mathbf{U} \cup \{X\}$, we find its score by looking up the parent graph of variable X to find the node that contains $\mathbf{U}$.

Various search methods as well as dynamic programming have been applied to solve the shortest-path finding problem (Malone, Yuan, and Hansen 2011; Malone et al. 2011; Yuan, Malone, and Wu 2011). In (Yuan, Malone, and Wu 2011), an A* search algorithm was proposed based on the following admissible heuristic function.

Definition 1. *Let* $\mathbf{U}$ *be a node in the order graph, its heuristic value is*

$$h(\mathbf{U}) = \sum_{X \in \mathbf{V} \setminus \mathbf{U}} BestScore(X, \mathbf{V} \setminus \{X\}). \quad (1)$$

The A* algorithm is shown to be much more efficient than existing dynamic programming algorithms. However, A* requires all the search information, including parent and order graphs, to be stored in RAM during the search, which makes the algorithm easily run out of memory for large datasets. Malone et al. (2011) developed a breadth-first branch and bound (BFBnB) algorithm to search the order graph in a layered, breadth-first order. By carefully coordinating the parent and order graphs, most of the search information can be stored on disk and are only processed incrementally after being read back to RAM when necessary. The BFBnB algorithm was shown to be as efficient as the A* algorithm but was able to scale to much larger datasets. Theoretically, the scalability of the BFBnB algorithm is only limited by the amount of disk space available.

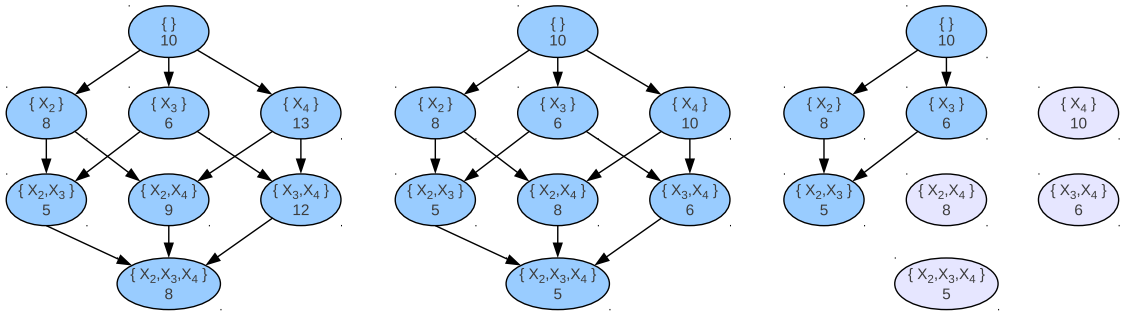


Figure 2: A sample parent graph for variable X_1. (a) The raw scores for all the parent sets. The first line in each node gives the parent set, and the second line gives the score of using all of that set as the parents for X_1. (b) The optimal scores for each candidate parent set. The second line in each node gives the optimal score using some subset of the variables in the first line as parents for X_1. (c) The unique optimal parent sets and their scores. The pruned parent sets are shown in gray. A parent set is pruned if any of its predecessors has an equal or better score.

3 An Improved Admissible Heuristic

The heuristic function defined in Equation 1 is based on a classic approach to designing admissible heuristics. Pearl (1984) pointed out that the optimal solution to a relaxed problem can be used as an admissible bound for the original problem. For structure learning, the original problem is to learn a Bayesian network that is an *acyclic* directed graph (DAG). Equation 1 relaxes the problem by completely ignoring the acyclicity constraint, so all *directed graphs* are allowed. This paper aims to improve the heuristic by enforcing partial acyclicity. We will first motivate our approach using a small example. We then describe the specifics of the new heuristic.

3.1 A motivating example

According to Equation 1, the heuristic estimate of the start node in the order graph allows each variable to choose optimal parents from all the other variables. Suppose the optimal parents for X_1, X_2, X_3, X_4 are $\{X_2, X_3, X_4\}$, $\{X_1, X_4\}$, $\{X_2\}$, $\{X_2, X_3\}$ respectively. These parent choices are shown as the directed graph in Figure 3. Since the acyclicity constraint is ignored, directed cycles are introduced, e.g., between X_1 and X_2. However, we know the final solution cannot have cycles; three scenarios are possible between X_1 and X_2: (1) X_2 is a parent of X_1 (so X_1 cannot be a parent of X_2), (2) X_1 is a parent of X_2, or (3) neither of the above is true. The third case is dominated by the other two cases because of the following theorem.

Theorem 1. *Let* $\mathbf{U}$ *and* $\mathbf{V}$ *be two candidate parent sets for* X*, and* $\mathbf{U} \subset \mathbf{V}$*, then* $BestScore(X, \mathbf{V}) \leq BestScore(X, \mathbf{U})$.

This theorem has appeared in many earlier papers, e.g. (Teyssier and Koller 2005; de Campos and Ji 2010), and simply means that an equal or better score can be obtained if a larger set of parent candidates is available to choose from. The third case cannot provide a better value than the other two cases because one of the variables must have fewer parents to choose from. Between the first two cases it is unclear which one is better, so we take the minimum of them. Consider the first case first: We have to delete the arc $X_1 \rightarrow X_2$ to rule out X_1 as a parent of X_2. Then we have to let X_2 to rechoose optimal parents from $\{X_3, X_4\}$, that is, we must check all the parent sets not including X_1; the deletion of the arc alone cannot produce the new bound. The total bound of X_1 and X_2 is computed by summing together the original bound of X_1 and the new bound of X_2. We call this total bound b_1. The second case is handled similarly; we call that total bound b_2. Because the joint heuristic for X_1 and X_2 must be optimistic, we compute it as the minimum of b_1 and b_2. Effectively we have considered all possible ways to break the cycle and obtained a new but improved heuristic value. The new heuristic is clearly admissible, as we still allow cycles among other variables.

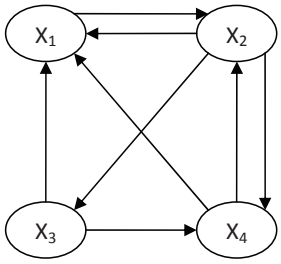


Figure 3: A directed graph representing the heuristic estimate for the start search node.

Often, the simple heuristic introduces multiple cycles. The graph in Figure 3 also has a cycle between X_2 and X_4. This cycle shares X_2 with the earlier cycle; we say they

overlap. Overlapping cycles cannot be broken independently. For example, suppose we break the cycle between X_1 and X_2 by setting the parents of X_2 to be $\{X_3\}$. This effectively breaks the cycle between X_2 and X_4 as well, but introduces a new cycle between X_2 and X_3. As described in more detail shortly, we divide the variables into non-overlapping groups and focus only on avoiding cycles within each group. So if X_2 and X_3 are in different groups, they are allowed to form a cycle.

3.2 The k-cycle conflict heuristic

The idea above can be generalized to compute the joint heuristics for all groups of variables with a size up to k by avoiding cycles within each group. We call the resulting technique the *k-cycle conflict heuristic*. Note that Equation 1 is a special case of this new heuristic, as it simply contains heuristics for the individual variables (k=1). The new heuristic is an application of the *additive pattern database* technique (Felner, Korf, and Hanan 2004). We first have to explain what *pattern database* (Culberson and Schaeffer 1998) is. Pattern database is an approach to computing an admissible heuristic for a problem by solving a relaxed problem. As an example, the 15 sliding tile puzzle can be relaxed to only contain the tiles 1-8 by removing the other tiles. Because of the relaxation, multiple states of the original problem are mapped to one state in the *abstract* state space of the relaxed problem. Each abstract state is called a *pattern*, and the exact costs for solving all the abstract states are stored in a pattern database; each cost can be retrieved as an admissible heuristic for any consistent state in the original state space. Furthermore, we can relax the problem in different ways and obtain multiple pattern databases. If the solutions to a set of relaxed problems are independent, the problems are said to have no interactions between them. Again consider 15-puzzle, we can also relax it to only contain the tiles 9-15. This relaxation can be solved independently from the previous one, as they do not share puzzle movements. The costs of their pattern databases can be *added* together to obtain an admissible heuristic, hence the name additive pattern databases.

In the learning problem, a pattern is defined as a set of variables, and its cost is the joint heuristic of the variables involved. The costs of two patterns sharing no variables can be added together to obtain an admissible heuristic because of the decomposability of the scoring function.

We do not need to explicitly break cycles to compute the k-cycle conflict heuristic. The following theorem offers a straightforward approach to computing the heuristic.

Theorem 2. *The cost of the pattern* $\mathbf{U}$ *is equal to the shortest distance from the node* $\mathbf{V} \setminus \mathbf{U}$ *to the goal node in the order graph.*

The theorem can be proven by noting that avoiding cycles between the variables in $\mathbf{U}$ is equivalent to finding an optimal ordering of the variables with the best joint score, and the different paths from $\mathbf{V} \setminus \mathbf{U}$ to the goal correspond to the different orderings of the variables, among which the shortest path thus corresponds to the optimal ordering. Again consider the example in Figure 3. The joint heuristic for the pattern $\{X_1, X_2\}$ is equal to the shortest distance from the node $\{X_3, X_4\}$ to the goal in Figure 1. Therefore, the new heuristic can be computed by finding the shortest distances from all the nodes in the last k layers of the order graph to the goal. We will describe a backward search algorithm for computing the heuristic in Section 4.2.

Furthermore, the *difference* between the cost of the pattern $\mathbf{U}$ and the simple heuristic of $\mathbf{V} \setminus \mathbf{U}$ indicates the amount of improvement brought by avoiding cycles within the pattern. The differential cost can thus be used as a quality measure for ordering the patterns and for choosing patterns that are more likely to result in a tighter heuristic. Also, we can discard any pattern that does not introduce additional improvement over any of its subset patterns. The pruning can significantly reduce the size of a pattern database and improve the efficiency of accessing the database.

3.3 Computing the heuristic for a search node

Once the k-cycle conflict heuristic is computed, we can use it to calculate the heuristic value for any node during the search. For a node $\mathbf{U}$, we need to partition $\mathbf{V} \setminus \mathbf{U}$ into a set of non-overlapping patterns, and sum their costs together as the heuristic value. There are potentially many ways to do the partition; ideally we want to find the one with the highest total cost, which represents the most accurate heuristic value. The problem of finding the optimal partition can be formulated as *maximum weighted matching* problem (Felner, Korf, and Hanan 2004). For $k = 2$, we can define a graph in which each vertex represents a variable, and each edge between two variables representing the pattern containing the same variables with an edge weight equal to the cost of the pattern. The goal is to select a set of edges from the graph so that no two edges share a vertex and the total weight of the edges is maximized. The matching problem can be solved in $O(n^3)$ time (Papadimitriou and Steiglitz 1982), where n is the number of vertices.

For $k > 2$, we have to add *hyperedges* to the matching graph for connecting up to k vertices to represent larger patterns. The goal becomes to select a set of edges and hyperedges to maximize the total weight. However, the three-dimensional or higher-order maximum weighted matching problem is NP-hard (Garey and Johnson 1979). That means we have to solve an NP-hard problem when calculating the heuristic value for a search node.

To alleviate the potential inefficiency, we elect to use a greedy method to compute the heuristic value. The method sequentially chooses patterns based on their quality. Consider the node $\mathbf{U}$; the unsearched variables are $\mathbf{V} \setminus \mathbf{U}$. We

first choose the pattern with the highest differential cost from all patterns that are subsets of $\mathbf{V} \setminus \mathbf{U}$. We repeat this process by choosing the next pattern for the remaining unsearched variables until all the variables are covered. The total cost of the chosen patterns is used as the heuristic value for the node $\mathbf{U}$.

3.4 Dynamic and static pattern databases

The version of the k-cycle conflict heuristic introduced above is an example of the *dynamically partitioned pattern database* (Felner, Korf, and Hanan 2004), as the patterns are dynamically selected during the search algorithm. We refer to it as *dynamic pattern database* for short. A potential drawback of dynamic pattern databases is that, even using the greedy method, computing a heuristic values is still more expensive than the simple heuristic in Equation 1. Consequently, the running time can be longer even though the tighter heuristic results in more pruning and fewer expanded nodes.

We can resort to another version of the k-cycle conflict heuristic based on the *statically partitioned pattern database* technique (Felner, Korf, and Hanan 2004). The idea is to statically divide all the variables into several groups, and create a separate pattern database for each group. Consider a problem with variables $\{X_1, ..., X_8\}$. We simply divide the variables into two equal-sized groups, $\{X_1, ..., X_4\}$ and $\{X_5, ..., X_8\}$. For each group, say $\{X_1, ..., X_4\}$, we create a pattern database that contains the costs of *all* subsets of $\{X_1, ..., X_4\}$ and store them as a hash table. We refer to this heuristic as the *static pattern database* for short.

It is much simpler to use static pattern databases to compute a heuristic value. Consider the node $\{X_1, X_4, X_8\}$; the unsearched variables are $\{X_2, X_3, X_5, X_6, X_7\}$. We divide these variables into two patterns $\{X_2, X_3\}$ and $\{X_5, X_6, X_7\}$ according to the static grouping. We then simply look up the costs of these two patterns in the pattern databases and sum them together as as the heuristic value for the node. Better yet, every search step only processes one variable and affects one pattern, so computing the heuristic value can be done incrementally.

4 The Search Algorithms

Both computing the k-cycle conflict heuristic and solving the shortest-path finding problem requires us to search the order graph. The searches further require the parent graphs to be calculated in advance or during the search. In this section, we first introduce a sparse representation of the parent graphs. We then discuss how to search the order graph backward to compute the k-cycle conflict heuristic, and forward to solve the shortest path-finding problem by adapting the A* and BFBnB algorithms.

$parents_{X_1}$	$\{X_2, X_3\}$	$\{X_3\}$	$\{X_2\}$	$\{\}$
$scores_{X_1}$	5	6	8	10

Table 1: Sorted scores and parent sets for X_1 after pruning parent sets which are not possibly optimal.

$parents_{X_1}$	$\{X_2, X_3\}$	$\{X_3\}$	$\{X_2\}$	$\{\}$
X_2	1	0	1	0
X_3	1	1	0	0
X_4	0	0	0	0

Table 2: The $parents_X(X_i)$ bit vectors for X_1. A "1" in line X_i indicates that the corresponding parent set includes variable X_i, while a "0" indicates otherwise. Note that, after pruning, none of the optimal parent sets include X_4.

4.1 Sparse representation of parent graphs

The parent graph for each variable X exhaustively enumerates the optimal scores for all subsets of $\mathbf{V} \setminus \{X\}$. Naively, this approach requires storing $n2^{n-1}$ scores and parent sets. Due to Theorem 1, however, the number of *unique* optimal parent sets is often far smaller. For example, Figure 2(b) shows that each score may be shared by several nodes in a parent graph. The parent graph representation will allocate space for this repetitive information, resulting in waste of space.

Instead of storing the complete parent graphs, we propose a sparse representation which *sorts* all the *unique* parent scores for each variable X in a list, and also maintain a parallel list that stores the associated optimal parent sets. We call these sorted lists $scores_X$ and $parents_X$. Table 1 shows the sorted lists for the parent graph in Figure 2(b). In essence, this allows us to store and efficiently process only scores in Figure 2(c). We do not have to create the full parent graphs before realizing some scores can be pruned (post-pruning). For example, we can use the following theorem (Tian 2000) to prune some scores before even computing them (pre-pruning).

Theorem 3. *In an optimal Bayesian network based on the MDL scoring function, each variable has at most* $\log(\frac{2N}{\log N})$ *parents, where* N *is the number of data points.*

Because of the pruning of duplicate scores, the sparse representation requires much less memory than storing all the possible parent sets and scores. As long as $\|scores(X)\| < C(n-1, \frac{n}{2})$, it also requires less memory than the BFBnB algorithm for X. In practice, $\|scores_X\|$ is almost always smaller than $C(n-1, \frac{n}{2})$ by several orders of magnitude. So this approach offers (usually substantial) memory savings compared to previous best approaches. In addition, the sparse representation is also much more efficient to create because of the pre-pruning.

The key operation in parent graphs is querying the opti-

$parents_{X_1}$	$\{X_2, X_3\}$	$\{X_3\}$	$\{X_2\}$	$\{\}$
$valid_{X_1}$	1	1	1	1
$\sim X_3$	0	0	1	1
$valid^{new}_{X_1}$	0	0	1	1

Table 3: The result of performing the bitwise operation to exclude all parent sets which include X_3. A "1" in the $valid_{X_1}$ bit vector means that the parent set does not include X_3 and can be used for selecting the optimal parents. The first set bit indicates the best possible score and parent set.

$parents_{X_1}$	$\{X_2, X_3\}$	$\{X_3\}$	$\{X_2\}$	$\{\}$
$valid_{X_1}$	0	0	1	1
$\sim X_2$	0	1	0	1
$valid^{new}_{X_1}$	0	0	0	1

Table 4: The result of performing the bitwise operation to exclude all parent sets which include either X_3 or X_2. A "1" in the $valid^{new}_{X_1}$ bit vector means that the parent set includes neither X_2 nor X_3. The initial $valid_{X_1}$ bit vector had already excluded X_3, so finding $valid^{new}_{X_1}$ only required excluding X_2.

mal parents for variable X out of a candidate set $\mathbf{U}$. With the sparse representation, we can simply scan the list of X starting from the beginning. As soon as we find the first parent set that is a subset of $\mathbf{U}$, we find the optimal parent set and its score. However, scanning the lists can be inefficient if not done properly. Since we have to do the scanning for each arc, the inefficiency will have a large impact on the whole search algorithm. We therefore propose the following incremental approach. Initially, we allow each variable X to use all the other variables as candidate parents, so the first element in the sorted score list must be optimal. For example, the first score in Table 1 must be $BestScore(X_1, \{X_2, X_3, X_4\})$. Suppose we remove X_2 from consideration as a candidate parent; we scan the list by continuing from where we last stopped and find a parent set which does not include X_2, which must be $BestScore(X_1, \{X_3, X_4\})$ ($\{X_3\}$ in this example). If we further remove X_3, we continue scanning the list until finding a parent set which includes neither X_2 nor X_3 to find $BestScore(X_1, \{X_4\})$ ($\{\}$ it is).

To further improve the efficiency, we propose the following efficient scanning technique. For each variable X, we first initialize an incumbent bit vector of length $\|scores_X\|$ called $valid_X$ to be all 1s. This indicates that all the parent scores in $scores_X$ are usable; the first score in the list will be the optimal score. Then, we create $n-1$ bit vectors also of length $\|scores_X\|$, one for each variable in $\mathbf{V} \setminus \{X\}$. The bit vector for variable Y is denoted as $parents_X(Y)$ and contains 1s for all the parent sets that contain Y and 0s for others. Table 2 shows the bit vectors for Table 1. Then, to exclude variable Y as a candidate parent, we perform the bit operation $valid^{new}_X \leftarrow valid_X \& \sim parents_X(Y)$. The $valid^{new}_X$ bit vector now contains 1s for all the parent sets that are subsets of $\mathbf{V} \setminus \{Y\}$. The *first set bit* corresponds to $BestScore(X, \mathbf{V} \setminus \{Y\})$. Table 3 shows an example of excluding X_3 from the set of possible parents for X_1, and the first set bit in the new bit vector corresponds to $BestScore(X_1, \mathbf{V} \setminus \{X_3\})$. If we further exclude X_2, the bit vector resulting from the last step becomes the incumbent bit vector, and a similar bit operation is applied: $valid^{new}_X \leftarrow valid_X \& \sim parents_{X_1}(X_2)$. The first set bit of the result corresponds to $BestScore(X_1, \mathbf{V} \setminus \{X_2, X_3\})$. Table 4 demonstrates this operation. Also, it is important to note that we exclude one variable at a time. For example, if, after excluding X_3, we wanted to exclude X_4 rather than X_2, we could take $valid^{new}_X \leftarrow valid_X \& \sim parents_X(X_4)$.

4.2 Creating the k-cycle conflict heuristic

We have two versions of the k-cycle conflict heuristic: dynamic and static pattern databases. To compute the dynamic pattern database, we use the *breadth-first search* to do a backward search for k layers in the order graph. The search starts from the goal node and expands the order graph backward layer by layer. A reverse arc from $\mathbf{U} \cup \{X\}$ to $\mathbf{U}$ has a cost equal to $BestScore(X, \mathbf{U})$. The reverse g cost to $\mathbf{U}$ is updated whenever a new path with a lower cost is found. Breadth-first search ensures that node $\mathbf{U}$ will obtain its optimal reverse g cost once the whole layer is processed. Its corresponding pattern $\mathbf{V} \setminus \mathbf{U}$ is pruned if the differential score is equal to that of any subset pattern. Otherwise, it is added to the pattern database together with both its pattern cost and differential cost.

The static pattern databases are calculated differently. For a static grouping $\mathbf{V} = \bigcup_i \mathbf{V_i}$, we need to compute a pattern database for each group $\mathbf{V_i}$, which is basically a full order graph containing all subsets of $\mathbf{V_i}$. We will also use a backward breadth first search to create the graph layer by layer starting from the node $\mathbf{V_i}$. However, the cost for any reverse arc from $\mathbf{U} \cup \{X\}$ to $\mathbf{U}$ in this order graph will be $BestScore(X, (\bigcup_{j \neq i} \mathbf{V_j}) \cup \mathbf{U})$.

4.3 Solving the shortest-path finding problem

After the pattern database heuristics are computed, we solve the shortest-path finding problem using a forward search in the order graph. We adapt both the A* (Yuan, Malone, and Wu 2011) and BFBnB (Malone, Yuan, and Hansen 2011) algorithms to utilize the new heuristic and the sparse parent graphs.

Originally, the A* algorithm first creates the full parent graphs and then expands the order graph in a best-first order starting from the top. For the improved version, we first create the unique score lists and the k-cycle conflict heuristic. During the search, the only difference appears in gener-

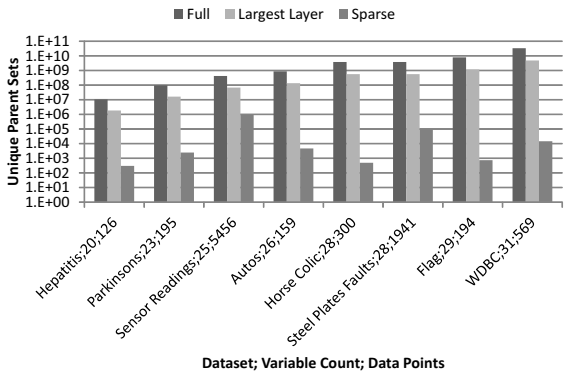


Figure 4: The number of parent sets stored in the full parent graphs ("Full"), the largest layer of the parent graphs ("Largest Layer"), and the sparse representation ("Sparse").

ating the successors of a node. For each successor $\mathbf{U} \cup \{X\}$ of node $\mathbf{U}$, we calculate its heuristic value according to the methods described in Sections 3.3 and 3.4. Looking up the cost $BestScore(X, \mathbf{U})$ for the arc $\mathbf{U} \rightarrow \mathbf{U} \cup \{X\}$ is achieved by using the sparse parent graphs.

The BFBnB algorithm is affected in a similar way. Originally it works by coordinating the expansion of the order graph and parent graphs layer by layer. In the improved version, the unique score lists and the heuristic are calculated first. The search part of the algorithm only needs to expand the order graph, during which generating successors works similarly as in the improved A* algorithm.

5 Empirical Results

We tested our new techniques on the A* and BFBnB algorithms by comparing to their original versions[1]. The experiments were performed on a PC with 3.07 GHz Intel i7 processor, 16 GB of RAM, 500 GB of hard disk space, and running Ubuntu 10.10. We used benchmark datasets from the UCI machine learning repository (Frank and Asuncion 2010) to test the algorithms. For all the datasets, records with missing values were removed. All variables were discretized into two states around their means.

5.1 Memory savings of sparse parent graphs

We first evaluated the memory savings made possible by using the sparse representation in comparison to the full parent graphs. In particular, we compared the maximum number of scores that have to be stored for all variables at once by each algorithm. A typical dynamic programming algorithm stores scores for all possible parent sets of all variables. BFBnB and memory-efficient dynamic programming (Malone, Yuan, and Hansen 2011) (assuming implementation optimizations) store all possible parent sets only in one layer of the parent graphs for all variables, so the size of the largest layer of all parent graphs is an indication of its space requirement. The sparse representation only stores the unique optimal parent sets for all variables at all layers.

Figure 4 shows the memory savings by the sparse representation. The number of unique scores stored by the sparse representation is typically several orders of magnitude smaller than the number of parent sets stored by the full representation. These results agree quite well with previously published results (de Campos and Ji 2010).

Due to Theorem 3, increasing the number of data records increases the maximum number of candidate parents. Therefore, the number of unique candidate parent sets increases as the number of records increases; however, many of the new parent sets are pruned. The number of variables also affects the number of candidate parent sets. Consequently, the number of unique scores increases as a function of the number of records and the number of variables. The amount of pruning is data-dependent, though, and is not easily predictable. In practice, we find the number of records to affect the number of unique scores more than the number of variables. Other scoring functions, such as BDe, exhibit similar behavior.

The results also suggest that the savings increase as the number of variables increases in the datasets. This implies that, while more variables necessarily increases the number of possible parent sets exponentially, the number of unique optimal parent sets increases much more slowly. Intuitively, even though we add more parents, only a small number of them are "good" parents for any particular variable.

5.2 Heuristics vs sparse representation

Both the new heuristic and the sparse representation can be used to improve the A* and BFBnB algorithms. It is beneficial to have an understanding on how much improvement each technique contributes. Also, the new heuristic has two versions: static and dynamic pattern databases; each of them can be parameterized in different ways. We applied various parameterizations of the new techniques to the algorithms on the datasets Autos and Flag. For the dynamic pattern database, we varied k from 2 to 4. For the static pattern databases, we tried groupings 9-9-8 and 13-13 for the Autos dataset and groupings 10-10-9 and 15-14 for the Flag dataset. The results are shown in Table 5.

The sparse representation helped both A* and BFBnB algorithms to achieve much better efficiency and scalability. A* ran out of memory on both of the datasets when using full parent graphs, but was able to solve both Autos (with

[1] A software package named *URLearning* ("you are learning") implementing the A* and BFBnB algorithms can be downloaded at `http://url.cs.qc.cuny.edu/software.html`.

Pattern Database			BFBnB, Full		BFBnB, Sparse		A*, Sparse	
Dataset	Type	Size	Time (s)	Nodes	Time (s)	Nodes	Time (s)	Nodes
Autos	Simple	26	2,690	62,721,601	461	62,721,601	674	35,329,016
Autos	Dynamic, k=2	41	2,722	52,719,774	449	52,719,793	148	6,286,142
Autos	Dynamic, k=3	116	2,720	49,271,793	468	49,271,809	76	2,829,877
Autos	Dynamic, k=4	582	2,926	48,057,187	699	48,057,205	67	2,160,515
Autos	Static, 9-9-8	1,280	2,782	57,002,704	495	57,002,715	228	9,763,518
Autos	Static, 13-13	16,384	2,747	48,814,324	211	48,814,334	125	4,762,276
Flag	Simple	29	OT	OT	OT	OT	OM	OM
Flag	Dynamic, k=2	45	OT	OT	1,222	132,431,610	824	19,359,296
Flag	Dynamic, k=3	149	OT	OT	788	79,332,390	207	5,355,085
Flag	Dynamic, k=4	858	OT	OT	1,624	84,054,443	350	7,377,817
Flag	Static, 10-10-9	2,560	OT	OT	2,600	249,638,318	OM	OM
Flag	Static, 15-14	49,152	OT	OT	720	88,305,173	136	4,412,232

Table 5: A comparison of the enhanced A* and BFBnB algorithms with various combinations of parent graph representations (full vs. sparse) and the heuristics (simple heuristic, dynamic pattern database with $k = 2$, 3, and 4, and static pattern databases with groupings 9-9-8 and 13-13 for the Autos dataset and groupings 10-10-9 and 15-14 for the Flag dataset). "Size" means the number of patterns stored; "Sparse" means the sparse parent graphs; "Full" means the full parent graphs; "Time" means the running time (in seconds), and "Nodes" means the number of nodes expanded by the algorithms; "OT" means the algorithm fail to finish within a 1-hour time limit set for this experiment; and "OM" means the algorithm used up all the RAM (16G). "A*, Full" is not included because it ran out of memory in all cases.

any heuristic) and Flag (with some of the best heuristics) when using sparse parent graphs. Similarly, BFBnB ran out of time on the Flag dataset within the one hour time limit when using full parent graphs, but was able to solve the dataset using the sparse representation (except when using the simple heuristic); on Autos, the sparse representation helped improve the time efficiency of BFBnB by up to an order of magnitude. One last note here is the numbers of expanded nodes by BFBnB are slightly different when using the two representations; it is only because of the randomness in the local search method used to compute the initial upper bound solution for BFBnB.

Both the static and dynamic pattern databases helped A* and BFBnB algorithms to improve efficiency and scalability. A* with both the simple heuristic and the static pattern database with grouping 10-10-9 ran out of memory on the Flag dataset. The other pattern database heuristics enabled A* to finish successfully. The dynamic pattern database with $k = 2$ helped to reduce the number of expanded nodes significantly for both algorithms on the datasets. Setting $k = 3$ helped even more. However, further increasing k to 4 often resulted in increased running time, and sometimes an increased number of expanded nodes as well. We believe that a larger k always results in a better heuristic; the occasional increase in expanded nodes is because the greedy strategy we used to choose patterns did not fully utilize the larger pattern database. The longer running time is reasonable though because it is less efficient to compute a heuristic value in larger pattern databases, and the inefficiency gradually overtook the benefit brought by the better heuristic. Therefore, $k = 3$ seems to be the best parametrization for the dynamic pattern database in general. For the static pattern databases, we were able to test much larger groups as we do not need to enumerate all groups up to a certain size. The results suggest that larger groupings tend to result in tighter heuristic.

The sizes of the static pattern databases are typically much larger than the dynamic pattern databases. However, they are still negligible in comparison to the number of expanded search nodes in all cases. It is thus cost effective to try to compute larger but affordable-size static pattern databases to achieve better search efficiency. The results show that the best static pattern databases typically helped A* and BFBnB to achieve better efficiency than the best dynamic pattern database, even when the number of expanded nodes is larger. The reason is calculating the heuristic values is more efficient when using static pattern databases.

5.3 Results on other datasets

Since static pattern databases seem to work better than dynamic pattern databases in most cases, we tested A* and BFBnB using static pattern database and sparse representation on all the datasets against the original algorithms. We used the simple static grouping of $\lceil \frac{n}{2} \rceil - \lfloor \frac{n}{2} \rfloor$ for all the datasets, where n is the number of variables. The results are shown in Table 6.

For the BFBnB algorithm, the improved version was around 5 times faster than the original version and sometimes even orders of magnitude faster (e.g. Flag). The reduction in the number of expanded nodes is not as dra-

Dataset				Results			
Name	n	N		BFBnB	BFBnB (SP)	A*	A* (SP)
Hepatitis	20	126	Time (s)	9	1	6	0
			Nodes	610,974	129,889	411,150	8,565
Parkinsons	23	195	Time (s)	100	19	100	15
			Nodes	8,388,607	4,646,877	8,388,607	1,152,576
Sensor Readings	25	5,456	Time (s)	632	3,121	OM	731
			Nodes	33,554,431	33,554,430	OM	3,286,650
Autos	26	159	Time (s)	1,170	211	OM	111
			Nodes	53,236,395	48,814,295	OM	4,762,276
Horse Colic	28	300	Time (s)	4,221	678	OM	OM
			Nodes	268,435,455	74,204,000	OM	OM
Steel Plates Faults	28	1,941	Time (s)	7,913	4,544	OM	OM
			Nodes	268,435,455	264,887,347	OM	OM
Flag	29	194	Time (s)	12,902	421	OM	147
			Nodes	354,388,170	88,305,173	OM	4,412,232
WDBC	31	569	Time (s)	93,382	26,196	OM	OM
			Nodes	1,353,762,809	273,746,036	OM	OM

Table 6: A comparison on the number of nodes expanded and running time (in seconds) of the A* and BFBnB algorithms enhanced by both static pattern database with grouping $\lceil \frac{n}{2} \rceil - \lfloor \frac{n}{2} \rfloor$, where n is the number of variables, and sparse representation of parent scores (denoted by "SP") against the original versions of these algorithms. "n" is the total number of variables, and "N" is the number of data points.

matic, however. The main reason is that the original BFBnB algorithm interleaves expanding the order graph and the full parent graphs during the search, while the improved version first calculates the sparse representation of parent scores, and then performs the search. It is much more efficient to compute the sparse representation than computing the full parent graphs. However, on the dataset Sensor Readings, the improved BFBnB algorithm runs slower than the original version. There are two potential explanations. First, this particular dataset has a large number of data points, which makes the sparse representation not truly sparse. Second, the new heuristic seems to be not much tighter than the simple heuristic on this dataset, because the numbers of expanded nodes are very similar in both cases.

The benefits of the new techniques are more obvious when applied to A*. For the datasets on which the original algorithm was able to finish, the improved algorithm was up to one order of magnitude faster; the number of expanded nodes is also significantly reduced. In addition, it was able to solve three larger datasets: Sensor Readings, Autos, and Flag. The running time on each of those datasets is pretty short, which indicates that once the memory consumption of the parent graphs was reduced, the A* algorithm was able to use more memory for the order graph and solved the search problems rather easily.

6 Concluding Remarks

The shortest-path finding formulation of the learning problem presented in (Yuan, Malone, and Wu 2011) makes two orthogonal directions of research natural. One is the development of search algorithms for learning optimal Bayesian networks, represented by the A* and BFBnB algorithms developed in (Yuan, Malone, and Wu 2011; Malone et al. 2011). One contribution of this paper is the sparse representation of the parent graphs which only store the unique optimal parent sets and scores. The method improves the time and space efficiency of the parent graph part of the search and thus falls in the first direction.

The second direction, which we believe is equally important, is the development of search heuristics. Another contribution of this paper is a new admissible heuristic called the k-cycle conflict heuristic developed based on the additive pattern databases. We tested the A* and BFBnB algorithms enhanced by the new heuristic and the sparse representation on a set of UCI machine learning datasets. The results show that both of the new techniques contributed to significant improvement in the efficiency and scalability of the algorithms. We therefore believe the new methods represent another significant step forward in exact Bayesian network structure learning.

As future work, we plan to investigate better approaches to obtaining the groupings for the static pattern databases. It could be based on prior knowledge, or some initial estimation of the correlation between the variables. Such groupings are expected to work better than the simple grouping we tested in this paper.

Acknowledgements This work was supported by NSF grants IIS-0953723 and EPS-0903787.

References

Culberson, J. C., and Schaeffer, J. 1998. Pattern databases. *Computational Intelligence* 14(3):318–334.

Cussens, J. 2011. Bayesian network learning with cutting planes. In *Proceedings of the Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, 153–160. Corvallis, Oregon: AUAI Press.

de Campos, C. P., and Ji, Q. 2010. Properties of bayesian dirichlet scores to learn bayesian network structures. In *Twenty-Fourth AAAI Conference on Aritificial Intelligence (AAAI-10)*, 431–436.

de Campos, C. P., and Ji, Q. 2011. Efficient structure learning of bayesian networks using constraints. 12:663–689.

Felner, A.; Korf, R. E.; and Hanan, S. 2004. Additive pattern database heuristics. *Journal of Artificial Intelligence Research (JAIR)* 22:279–318.

Frank, A., and Asuncion, A. 2010. UCI machine learning repository.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Heckerman, D. 1998. A tutorial on learning with Bayesian networks. In Holmes, D., and Jain, L., eds., *Innovations in Bayesian Networks*, volume 156 of *Studies in Computational Intelligence*. Springer Berlin / Heidelberg. 33–82.

Jaakkola, T.; Sontag, D.; Globerson, A.; and Meila, M. 2010. Learning Bayesian network structure using LP relaxations. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Koivisto, M., and Sood, K. 2004. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research* 549–573.

Malone, B.; Yuan, C.; Hansen, E.; and Bridges, S. 2011. Improving the scalability of optimal Bayesian network learning with external-memory frontier breadth-first branch and bound search. In *Proceedings of the Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, 479–488. Corvallis, Oregon: AUAI Press.

Malone, B.; Yuan, C.; and Hansen, E. A. 2011. Memory-efficient dynamic programming for learning optimal Bayesian networks. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11)*, 1057–1062.

Ott, S.; Imoto, S.; and Miyano, S. 2004. Finding optimal models for small gene networks. In *Pac. Symp. Biocomput*, 557–567.

Papadimitriou, C. H., and Steiglitz, K. 1982. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall.

Parviainen, P., and Koivisto, M. 2009. Exact structure discovery in Bayesian networks with less space. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. Montreal, Quebec, Canada: AUAI Press.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison & Wesley.

Rissanen, J. 1978. Modeling by shortest data description. *Automatica* 14:465–471.

Silander, T., and Myllymaki, P. 2006. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*. Arlington, Virginia: AUAI Press.

Singh, A., and Moore, A. 2005. Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University.

Teyssier, M., and Koller, D. 2005. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, 584–590. Arlington, Virginia: AUAI Press.

Tian, J. 2000. A branch-and-bound algorithm for MDL learning Bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 580–588. Morgan Kaufmann Publishers Inc.

Yuan, C.; Malone, B.; and Wu, X. 2011. Learning optimal Bayesian networks using A* search. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, 2186–2191.

FHHOP: A Factored Hybrid Heuristic Online Planning Algorithm for Large POMDPs

Zongzhang Zhang
School of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui 230027 China
zzz@mail.ustc.edu.cn

Xiaoping Chen
School of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui 230027 China
xpchen@ustc.edu.cn

Abstract

Planning in partially observable Markov decision processes (POMDPs) remains a challenging topic in the artificial intelligence community, in spite of recent impressive progress in approximation techniques. Previous research has indicated that online planning approaches are promising in handling large-scale POMDP domains efficiently as they make decisions "on demand" instead of proactively for the entire state space. We present a Factored Hybrid Heuristic Online Planning (FHHOP) algorithm for large POMDPs. FHHOP gets its power by combining a novel hybrid heuristic search strategy with a recently developed factored state representation. On several benchmark problems, FHHOP substantially outperformed state-of-the-art online heuristic search approaches in terms of both scalability and quality.

1 Introduction

Partially observable Markov decision processes (POMDPs) have been widely recognized as a powerful probabilistic model for planning and control problems in partially observable stochastic domains (Kaelbling et al. 1998). Solving POMDP problems exactly can be impossible due to their computational complexity (Madani et al. 1999). In the past decade, researchers have made impressive progress in designing approximate algorithms (Pineau et al. 2006; Kurniawati et al. 2008; Ross et al. 2008a) and have successfully applied them to various realistic robotic tasks (Hoey et al. 2007; Hsu et al. 2008).

Planning algorithms can be categorized as offline planning algorithms and online planning algorithms depending on the ways of solving problems. Offline approaches separate the policy-construction phase and the policy-execution phase. They devote significant pre-processing time to generate a policy over the whole belief space in the policy-construction phase, then use the resulting policy to make decisions during the policy-execution phase. Offline approaches can be beneficial for repeated POMDP planning tasks as the pre-processing time can be amortized over the various runs. In contrast, online approaches are a viable alternative for urgent or one-off POMDP planning tasks. They do not allocate significant time to pre-processing, but alternate between a time-limited policy-construction step for the current state and a policy-execution step (Ross et al. 2008a). Instead of finding policies that generalize to all possible situations, they focus on computing good local policies at each policy-construction step, and therefore can potentially produce a sequence of actions with high reward while spending much less overall time for policy construction and policy execution compared to offline algorithms.

In this paper, we aim to accelerate online POMDP planning algorithms by taking advantage of a more efficient heuristic search strategy and a factored state representation.

Our work on heuristics originates from the insight that lower bounds on the optimal value function have not been well exploited in current online algorithms. Current online algorithms maintain both the lower and upper bounds as a heuristic to find good policies. One key problem that determines the overall performance of current approaches is how to approximate the optimal policy according to the lower and upper bounds. In previous work (Ross et al. 2008a), the upper bound was usually more popular than the lower bound in the calculation. The search toward the action with the highest upper bound guarantees that an ϵ-optimal action can be found within finite time theoretically. In contrast, the lower bound is more difficult to be exploited since it can easily trap search into local optima. For example, the result of always exploring toward the action branch with the highest lower bound only makes us believe that this action branch is optimal, although it tends not to be so. However, the lower bound can guarantee the quality of policy, which is an advantage that the upper bound does not have. Thus, online algorithms typically return the best

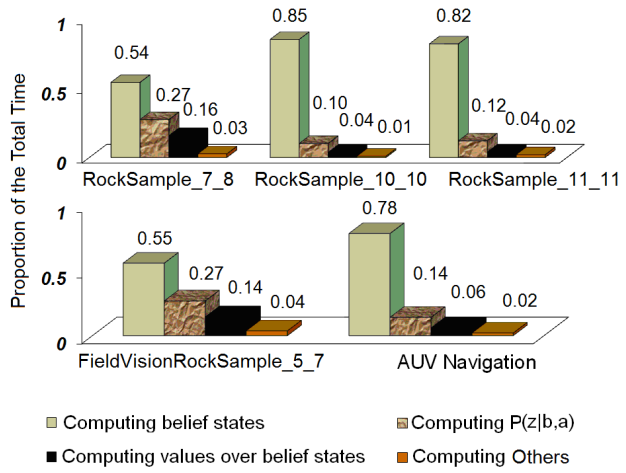


Figure 1: AEMS2's computational profile on several benchmark problems. The plot shows, for example, that on RockSample_7_8 AEMS2 spends 54%, 27% and 16% of its running time in computing new belief states, $P(z|b,a)$, and values over belief states given a value function, respectively. See text for further explanations.

action with the highest lower bound but not with the highest upper bound. In this paper, we discuss how to take full advantage of both the lower and upper bound in a novel hybrid way and avoid their disadvantages. The role of the lower bound in our heuristic method is to guide the search toward a set of promising policies whose quality may be better than the current best policy.

Our work on factorization was inspired by profiling data on several typical benchmark problems like those in Figure 1. Ignoring the details of the tasks for the time being, we can see that in all cases the anytime error minimization search 2 (AEMS2) algorithm, one of the most efficient online heuristic search algorithms (Ross and Chaib-draa 2007; Ross et al. 2008a), spends more than 95% of its overall running time in computing: (1) new belief states; (2) $P(z|b,a)$, the probabilities of observing z after taking action a at belief state b; and (3) values over belief states given a value function. These dramatic observations drive us to decrease overall computation time by reducing the time on these three operations. We show theoretically and empirically that it is possible to do so by taking advantage of the mixed observability MDP (MOMDP) representation (Ong et al. 2010). The MOMDP representation can be considered as an instance of a dynamic Bayesian network (DBN). It exploits mixed observability where some state variables are always observed and others are hidden to reduce the time complexity of state related operations.

We refer to our new algorithm based on the above two insights as Factored Hybrid Heuristic Online Planning (FHHOP). Experimental results reveal that FHHOP substantially outperformed the AEMS2 algorithm on all test problems. Especially, on some well-known benchmark problems, FHHOP has achieved more than an order of magnitude improvement in terms of runtime.

The remainder of the paper is organized as follows. Section 2 provides an overview of the POMDP model, online algorithms and the MOMDP representation. Section 3 describes the details in the FHHOP algorithm, which is our key contribution. Section 4 describes a set of experiments showing the efficacy of the FHHOP algorithm in terms of both runtime and solution quality. We conclude and discuss future work in Section 5.

2 Background and Related Work

In this section, we briefly introduce the POMDP model, online algorithms, and the MOMDP representation.

2.1 POMDP Model

POMDPs provide a very powerful mathematical model for an agent's decision making in partially observable domains. A discrete and discounted POMDP model can be formally defined by a tuple $(S, A, Z, T, \Omega, R, \gamma)$. In the tuple, S, A and Z are the finite and discrete state space, action space and observation space, respectively, $T(s,a,s') : S \times A \times S \to [0,1]$ is the state transition function ($P(s'|s,a)$), $\Omega(a,s',z) : A \times S \times Z \to [0,1]$ is the observation function ($P(z|a,s')$), $R(s,a) : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in (0,1)$ is the discount factor on the summed sequence of rewards. Because the agent's current state is not fully observable, the agent could have to rely on the complete history of past actions and observations to select a desirable current action. In the context of decision making, the *belief state* b is a sufficient statistic for the history of actions and observations (Smallwood and Sondik 1973). A belief state b is a discrete probability distribution over the state space, whose element $b(s)$ gives the probability that the agent's state is s. Let $\mathcal{B}$ be the space of all possible belief states and b_0 be the agent's *initial belief state*. Thus, $\mathcal{B}$ is an $|S|$-dimensional space, where $|S|$ is the number of states. In a so-called flat POMDP, all operations on beliefs and value functions are performed at the level of $|S|$-dimensional belief states.

When the agent takes action a at a belief state b and receives observation z, it will arrive at a new belief state $b^{a,z}$:

$$b^{a,z}(s') = \frac{1}{\eta}\Omega(a,s',z)\sum_{s\in S} T(s,a,s')b(s), \qquad (1)$$

where η is a normalizing constant (Kaelbling et al. 1998). The constant is the probability of receiving z after the agent

takes a at b and can be specified as:

$$P(z|b,a) = \sum_{s' \in S} \Omega(a,s',z) \sum_{s \in S} T(s,a,s')b(s). \quad (2)$$

A key goal in solving POMDPs is to find an *optimal policy* π^* that maximizes the *expected discounted reward* $V^\pi(b_0) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t)|b_0,\pi]$, where π denotes a policy that maps from belief states to actions, and s_t and a_t are the agent's state and action at time step t, respectively. The maximal expected discounted reward starting from any initial belief state in the whole belief space is captured by the optimal value function V^*, which can be defined via the fixed point of Bellman's equation (Bellman 1957):

$$V^*(b) = \max_{a \in A} Q^*(b,a), \quad (3)$$

where $Q^*(b,a) = R(b,a) + \gamma \sum_{z \in Z} P(z|b,a)V^*(b^{a,z})$ and $R(b,a) = \sum_{s \in S} R(s,a)b(s)$. The function V^* can be approximated infinitely closely by a piecewise linear and convex function:

$$V(b) = \max_{\alpha \in \Gamma} (\alpha \cdot b) = \max_{\alpha \in \Gamma} \sum_{s \in S} \alpha(s)b(s), \quad (4)$$

where Γ is a finite set of $|S|$-dimensional hyperplanes, called α-vectors, over $\mathcal{B}$ (Smallwood and Sondik 1973). The piecewise linear and convex property makes several exact and approximate value iteration algorithms feasible for solving POMDPs (Kaelbling et al. 1998; Kurniawati et al. 2008). Once V^* has been identified, computing the optimal policy π^* is a straightforward application of $\pi^*(b) = \text{argmax}_{a \in A} Q^*(b,a)$.

2.2 Approximate Online Algorithms

Online POMDP algorithms can be classified into three categories: branch-and-bound pruning, Monte-Carlo sampling and heuristic search (Ross et al. 2008a). In this paper, we only focus on one category, heuristic search online algorithms. The central idea in heuristic search online algorithms is to maintain lower and upper bounds on the value function, $\underline{V}(b) \leq V^*(b) \leq \bar{V}(b)$, and to iteratively improve these bounds at the current belief state b_c by expanding an AND/OR tree of reachable belief states from b_c. We denote $\underline{Q}(b,a)$ and $\bar{Q}(b,a)$, respectively, as the lower and upper bounds over $Q^*(b,a)$. Each OR-node in the tree represents a belief state and each AND-node represents an action choice from the belief state (node) above it, given the tree is drawn growing downwards. By abuse of notation, we will use b to represent a belief node in the tree and its associated belief state.

Algorithm 1 is a generic POMDP solver. It accepts three parameters: b_0, τ and ϵ, where b_0 is the initial belief state, τ is a bound on computation time allowed per policy-construction step, and ϵ is the desired precision on $V(b_c)$.

Algorithm 1: Generic Online POMDP Solver

Function OnlinePOMDPAlgorithm(b_0,τ,ϵ)
1: Initialize the bounds by offline algorithms;
2: $b_c = b_0$;
3: Build an AND/OR tree to contain b_c at the root;
4: **while** b_c is not a goal state
5: a =Search(b_c,τ,ϵ);
6: Take action a and receive a new observation z;
7: $b_c = b_c^{a,z}$;
8: Update the tree so that b_c is the new root;
9: **end while**

Algorithm 2: Policy Construction

Function Search(b_c,τ,ϵ)
1: StartTimer();
2: **while** ElapsedTime() $\leq \tau$ and $\bar{V}(b_c) - \underline{V}(b_c) > \epsilon$
3: b^* =ChooseBestNodetoExpand();
4: Expand(b^*);
5: UpdateAncestors(b^*);
6: **end while**
7: return $\text{argmax}_{a' \in A} \underline{Q}(b_c,a')$;

Online algorithms use this tree to alternate between policy construction and policy execution. The policy execution procedure (see Lines 6–8 in Algorithm 1) is consistent across all online algorithms.

Algorithm 2 is a general online policy-construction function. The ChooseBestNodetoExpand() procedure uses efficient heuristic search strategies to select a useful belief node to expand among the leaf nodes. The Expand() procedure expands a belief node, improving its lower and upper bound over V^* in the process. The UpdateAncestors() procedure concentrates on updating the bounds of a belief node's ancestors. On Line 7 of Algorithm 2, it returns the action branch at the root with the highest lower bound of $Q(b_c,a')$. We define the current best policy π_{best} to be the policy of always selecting the action $\pi_{best}(b) = \text{argmax}_{a' \in A} \underline{Q}(b,a')$.

To better describe our work on heuristics, we first use Figure 2 as an example to further illustrate how online algorithms work. Suppose that b_{c+k} is the leaf belief node that ChooseBestNodetoExpand() returns. Let b_{c+k} be a leaf node that is reachable from the root belief node b_c by following a k-step action-observation sequence, $a_0 z_1 a_1 z_2 \ldots a_{k-1} z_k$. After expanding b_{c+k}, an improved $\underline{V}(b_{c+k})$ and $\bar{V}(b_{c+k})$ can be obtained. Furthermore, the invocation of UpdateAncestors() results in the improved bounds at all belief nodes and Q nodes along this path. In other words, given that b_{c+k} is reachable from b_c by following some policy π, the execution of Line 3–5 in Algorithm 2 will lead to improved $\underline{V}^\pi(b_c)$ and $\bar{V}^\pi(b_c)$,

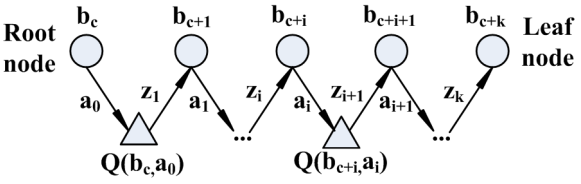


Figure 2: A path from the root belief node b_c to the leaf belief node b_{c+k}.

the lower and upper bounds on the expected discounted reward generated by following π at b_c. If $\underline{V}^{\pi_{best}}(b_c)$ in the previous round is surpassed by the new improved $\underline{V}^{\pi}(b_c)$, then a policy with higher lower bound will be found.

2.3 Heuristics in Current Online Algorithms

In current heuristic search online algorithms, each leaf belief node has an associated heuristic value. The invocation of ChooseBestNodetoExpand() returns the leaf belief node with the maximal heuristic value. If an oracle provides us with the optimal value function V^*, a heuristic function over a leaf node b_{c+k} with good theoretical guarantees can be defined as follows (Ross et al. 2008b):

$$H^*(b_{c+k}) = e^*(b_{c+k}) \prod_{t=0}^{k-1} \omega^*(b_{c+t}, a_t)\omega(b_{c+t}, a_t, z_{t+1}), \tag{5}$$

where $e^*(b_{c+k}) = V^*(b_{c+k}) - \underline{V}(b_{c+k})$, $\omega(b_{c+t}, a_t, z_{t+1}) = \gamma P(z_{t+1}|b_{c+t}, a_t)$, and

$$\omega^*(b_{c+t}, a_t) = \begin{cases} 1 & \text{if } a_t \in \operatorname{argmax}_{a' \in A} Q^*(b_{c+t}, a'), \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

In such a heuristic function, each quantity plays an important role in expanding the leaf nodes: $e^*(b_{c+k})$ encourages exploration of leaf nodes with loose bounds, $\omega^*(b_{c+t}, a_t)$ focuses the exploration toward the leaf nodes that are reachable from b_c under the optimal policy, and $\omega(b_{c+t}, a_t, z_{t+1})$ guides the search toward belief nodes that are likely to be encountered in the future. Empirical results in prior work (Ross et al. 2008a) teach us that ignoring some quantities in Equation 5 would damage the overall performance of online approaches. For example, the lack of $\omega(b_{c+t}, a_t, z_{t+1})$ in the BI-POMDP algorithm (Washington 1997) may explain its poor performance on some POMDP problems with large observation spaces.

However, the optimal policy or value function is not available for constructing H^*. Thus, the finding of a set of "promising" policies is one of the core issues for improving the speed of online algorithms. The approach of Satia and Lave (Satia and Lave 1973) assumes the set of promising policies are all possible optimal policies. In this approach, little heuristic information about $\underline{V}$ and $\bar{V}$ is exploited to focus its search toward actions that look promising, and therefore the approach of Satia and Lave is not able to scale well to large-scale POMDPs. The AEMS1 algorithm (Ross et al. 2008a) takes both $\underline{V}$ and $\bar{V}$ into account by favoring exploration of action branches with high average values of $\underline{Q}(b, a)$ and $\bar{Q}(b, a)$. Such an algorithm loses its competitive capability in large-scale POMDP problems partially because its search strategy does not use the highest lower or upper bound well. In contrast to AEMS1, the BI-POMDP and AEMS2 algorithms use $\bar{V}$ as an estimated substitute for V^*. $H_U(b_{c+k})$ is the AEMS2 algorithm's heuristic:

$$H_U(b_{c+k}) = e(b_{c+k}) \prod_{t=0}^{k-1} \omega(b_{c+t}, a_t)\omega(b_{c+t}, a_t, z_{t+1}), \tag{7}$$

where $e(b_{c+k}) = \bar{V}(b_{c+k}) - \underline{V}(b_{c+k})$ and

$$\omega(b_{c+t}, a_t) = \begin{cases} 1 & \text{if } a_t \in \operatorname{argmax}_{a' \in A} \bar{Q}(b_{c+t}, a'), \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

From the definition of $\omega(b_{c+t}, a_t)$, we can see AEMS2 assumes that the action with the maximal upper bound is in fact the optimal action. Such a definition, sometimes called the IE-MAX heuristic (Smith and Simmons 2004), leads AEMS2 to find an ϵ-optimal action within finite time (Ross et al. 2008b). The action branch according to the highest lower bound is deliberately ignored in AEMS2 simply because choosing leaf nodes under such an action branch to expand could only cause its lower bound to rise, and therefore, easily trap search in local optima. However, $\underline{Q}(b, a)$ may be a very useful heuristic in finding a better policy. A direct insight is that the action that Algorithm 1 returns is the action corresponding to the maximal $\underline{Q}(b, a)$ but not the maximal $\bar{Q}(b, a)$. One of our contributions is to suggest that using the lower bound could provide improvements compared to the heuristics used in AEMS2.

2.4 MOMDPs

A MOMDP can be generally specified as a tuple $(\mathcal{X}, \mathcal{Y}, A, Z, T_{\mathcal{X}}, T_{\mathcal{Y}}, \Omega, R, \gamma)$, where $\mathcal{X}$ is the set of fully observable state variables, $\mathcal{Y}$ is the set of partially observable state variables, $S = \mathcal{X} \times \mathcal{Y}$, $T_{\mathcal{X}}(x, y, a, x') : \mathcal{X} \times \mathcal{Y} \times A \times \mathcal{X} \to [0, 1]$ and $T_{\mathcal{Y}}(x, y, a, x', y') : \mathcal{X} \times \mathcal{Y} \times A \times \mathcal{X} \times \mathcal{Y} \to [0, 1]$ are the two corresponding probabilistic state-transition functions ($P(x'|x, y, a)$ and $P(y'|x, y, a, x')$), $\Omega(a, x', y', z) : A \times X \times Y \times Z \to [0, 1]$ is the observation function ($P(z|a, x', y')$), $R(x, y, a) : \mathcal{X} \times \mathcal{Y} \times A \to \mathbb{R}$ is the reward function, and the other quantities are the same as a POMDP model's elements. Since x is fully observable, a belief state b on the underlying state $s = (x, y)$ can be represented as $(x, b_{\mathcal{Y}}(x))$. Let $\mathcal{B}_{\mathcal{Y}}$ be the belief space over $\mathcal{Y}$, and $\mathcal{B}_{\mathcal{Y}}(x) = \{(x, b_{\mathcal{Y}}(x))|b_{\mathcal{Y}}(x) \in \mathcal{B}_{\mathcal{Y}}\}$, then $\mathcal{B} =$

$\bigcup_{x\in\mathcal{X}} \mathcal{B}_\mathcal{Y}(x)$. Thus, computations involving updating beliefs and value functions can be restricted to one of $|\mathcal{Y}|$-dimensional subspaces, $\mathcal{B}_\mathcal{Y}(x)$. Please note that $\mathcal{B}$ has $|\mathcal{X}||\mathcal{Y}|$ dimensions, while $\mathcal{B}_\mathcal{Y}$ has only $|\mathcal{Y}|$ dimensions, where $|\mathcal{X}|$ and $|\mathcal{Y}|$ are the number of fully and partially observable state variables, respectively.

After taking a and receiving z from $(x, b_\mathcal{Y}(x))$, a new belief $(x^{a,z}, b_\mathcal{Y}(x^{a,z}))$ results. The variable $x^{a,z}$ can be directly inferred from z, and $b_\mathcal{Y}(x^{a,z})$ can be computed as follows:

$$b_\mathcal{Y}(x^{a,z}, y') = \frac{1}{\eta}\Omega(a, x^{a,z}, y', z) \sum_{y\in\mathcal{Y}} T_{\mathcal{X}\mathcal{Y}} b_\mathcal{Y}(x, y), \quad (9)$$

where $T_{\mathcal{X}\mathcal{Y}} = T_\mathcal{X}(x, y, a, x^{a,z}) T_\mathcal{Y}(x, y, a, x^{a,z}, y')$ and η, the probability of receiving z after taking a at $(x, b_\mathcal{Y}(x))$ is

$$P(z|x, b_\mathcal{Y}(x), a, x^{a,z}) = \sum_{y'\in\mathcal{Y}} \Omega(a, x^{a,z}, y', z) \sum_{y\in\mathcal{Y}} T_{\mathcal{X}\mathcal{Y}} b_\mathcal{Y}(x, y). \quad (10)$$

For any given x, $V^*(x, b_\mathcal{Y}(x))$ can be accurately approximated by a finite set of $|\mathcal{Y}|$-dimensional vectors $\Gamma_\mathcal{Y}(x)$ over $\mathcal{B}_\mathcal{Y}(x)$:

$$V(x, b_\mathcal{Y}(x)) = \max_{\alpha\in\Gamma_\mathcal{Y}(x)} (\alpha \cdot b_\mathcal{Y}(x)). \quad (11)$$

From the description of MOMDPs, we can see any MDP can be written as a MOMDP and any POMDP can be written as a MOMDP. That is because the MDP states can reside in the fully observable part of the MOMDP description and the POMDP states can reside in the partially observable part of a MOMDP description. Similarly, any MOMDP can be written as a POMDP by treating all states as partially observable (Ong et al. 2010).

3 Factored Hybrid Heuristic Online Planning

In this section, we describe the FHHOP algorithm in detail. First, we construct a novel hybrid heuristic strategy by combining a new heuristic function using the lower bound with an existing heuristic function using the upper bound to improve the heuristic search mechanism in the ChooseBestNodetoExpand() procedure. Then, we concentrate on the reason that the MOMDP representation is useful to reduce the running time in the Expand() and UpdateAncestors() procedures. Finally, we discuss the convergence property of the FHHOP algorithm.

3.1 Constructing a Heuristic Function Using the Lower Bound

As mentioned in the last paragraph of Section 2.2, to find a better policy, we need to select a policy π and expand the leaf belief nodes that are reachable from b_c by following the policy π. Our goal here is to use the lower bound as a heuristic to select a set of promising policies so that the exploration toward them will lead to find an even better policy as quickly as possible. Our idea is to elicit a set of policies π from all possible policies, whose $\underline{V}^\pi(b_c)$ are close to $\underline{V}^{\pi_{best}}(b_c)$, according to the lower bound.

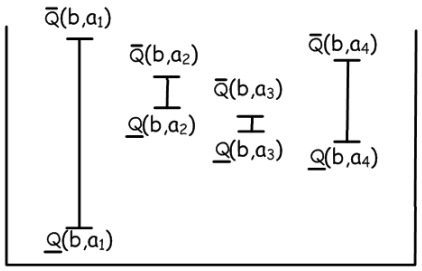


Figure 3: An example of action branches at b. $A_L = \{a_2\}$, $A_S = \{a_1, a_4\}$, and current second-best action is a_4. The action branch a_3 is not included in A_S because $\bar{Q}(b, a_3)$ is smaller than $\underline{Q}(b, a_2)$.

First, define $\omega_1(b, a)$, a term that guides the exploration toward belief nodes that are reachable from b_c under the current best policy, as follows:

$$\omega_1(b, a) = \begin{cases} 1 & \text{if } a \in \operatorname{argmax}_{a'\in A} \underline{Q}(b, a'), \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

Then, we consider the action branches at a single belief node b. We assume $A_L = \operatorname{argmax}_{a'\in A} \underline{Q}(b, a')$ and $A_S = \{a \in A \setminus A_L | \bar{Q}(b, a) > \max_{a'\in A} \underline{Q}(b, a')\}$. Thus, A_L is a current best action branch at b and A_S is a set of actions without both A_L and suboptimal action branches. The condition $\bar{Q}(b, a) > \max_{a'\in A} \underline{Q}(b, a')$ is used to prune suboptimal action branches at b, just like the branch-and-bound pruning technique in the approach of Satia and Lave. Then, we define

$$\omega_2(b, a) = \begin{cases} 1 & \text{if } a \in \operatorname{argmax}_{a'\in A_S} \underline{Q}(b, a'), \\ 0 & \text{otherwise,} \end{cases} \quad (13)$$

where $\operatorname{argmax}_{a'\in A_S} \underline{Q}(b, a')$ is called the current second-best action at b. So, $\omega_2(b, a)$ is a term that encourages the exploration toward belief nodes that are reachable from b's second-best action branch. We use Figure 3 to provide a direct explanation of these notions. The current second-best action looks a bit like the action branch with the second highest upper bound in this figure, but they actually have nothing to do with each other.

Finally, we figure out the way of eliciting a set of promising policies according to the lower bound. Obviously, the nodes that are reachable from b_c by following the set of promising policies constitute of a subtree of the current AND/OR tree. We are interested in all leaf nodes of

this subtree reachable from b_c through **one and only one** current second-best action branch and all other current best action branches. Such a strategy can be well depicted by $\omega_{1,2}(b_{c+k})$:

$$\omega_{1,2}(b_{c+k}) = \max_{i\in\{0,1,\ldots,k-1\}} \omega_2(b_{c+i}, a_i) \prod_{\substack{t=0 \\ t\neq i}}^{k-1} \omega_1(b_{c+t}, a_t), \tag{14}$$

where b_{c+i} is the only belief node in which its current second-best action branch is chosen. When $\omega_{1,2}(b_{c+k})$ equals 1, the leaf node b_{c+k} is reachable from b_c by following one of such a set of promising policies. We construct a new heuristic $H_L(b)$ for each leaf node using $\omega_{1,2}(b_{c+k})$ as follows:

$$H_L(b_{c+k}) = e(b_{c+k})\omega_{1,2}(b_{c+k}) \prod_{t=0}^{k-1} \omega(b_t, a_t, z_{t+1}). \tag{15}$$

From Equation 15, we can see that the leaf nodes being selected to expand using $H_L(b)$ are not reachable from b_c under the current best policy. However, they are reachable from b_c under a set of promising policies that are very similar to the current best policy since they have only one action branch that is different from the current best action branch. Such a heuristic $H_L(b)$ has several advantages, as follows:

- First, a better policy according to the lower bound would be quickly found using $H_L(b)$ only if the improved Q-value at the second-best action branch is greater than the Q-value at the current best action branch.
- Second, it focuses search toward a set of promising policies but not a single policy. The main drawback of expanding the leaf nodes that are reachable under a single policy is that the time of expanding is wasted if such a single policy is not better than the current best policy. Since $H_L(b)$ selects a set of promising policies, suboptimal policies will be eliminated as the depth of its leaf nodes increases. In other words, $H_L(b)$ favors exploration toward the highest potential policy among the set of promising policies as time goes on.
- Third, such a set of promising policies is only a very small subset in the set of all possible policies. Therefore, the technique can avoid exploring a large policy space to satisfy the requirement of real-time limitations in online algorithms.

However, such a heuristic may still be at the risk of trapping the search into local optima if the optimal policy is outside of the set of promising policies and the current best policy. Thus, our new heuristic function $H_L(b)$ might be only a greedy heuristic and not lead to optimal behavior in the end.

3.2 Constructing a Hybrid Heuristic Strategy

Before we construct a hybrid strategy, recall the strength and weakness of $H_L(b)$ and $H_U(b)$. $H_L(b)$ is a heuristic that depends on the lower bound. Its advantage is to guide search toward finding a better policy with a higher lower bound guarantee quickly. However, $H_L(b)$ limits its search effort to a set of promising policies. If both the current policy and the set of promising policies are suboptimal, the search led by $H_L(b)$ might become trapped in a local optimum. $H_U(b)$ is a heuristic strategy that depends on the upper bound. It favors exploration toward an ϵ-optimal action branch. The ϵ-optimal action can be found theoretically even if $H_L(b)$ is not used. However, the process of finding an ϵ-optimal action is usually very slow, especially in POMDP domains with large action and observation spaces. Because their strengths are complementary, it seems to be reasonable to combine them for obtaining a hybrid heuristic search strategy that leverages the advantages of both heuristics.

We suggest a way of constructing a hybrid strategy. Let $\mathcal{L}$ be a set of leaf belief nodes in the AND/OR tree,

$$b_U = \operatorname*{argmax}_{b\in\mathcal{L}} H_U(b) \tag{16}$$

and

$$b_L = \operatorname*{argmax}_{b\in\mathcal{L}} H_L(b). \tag{17}$$

Then, our hybrid strategy selects b^* by Equation 18:

$$b^* = \begin{cases} b_U & \text{if } C_U H_U(b_U) > C_L H_L(b_L), \\ b_L & \text{otherwise,} \end{cases} \tag{18}$$

where C_i denotes expected change value of both $\underline{V}(b_c)$ and $\bar{V}(b_c)$ if b_i is chosen to be expanded. We use the following formula to compute C_i in our experiments:

$$C_i = \frac{I_i + 1}{N_i + 1}, \tag{19}$$

where $i = U$ or L, I_i denotes the accumulative total change value of both $\underline{V}(b_c)$ and $\bar{V}(b_c)$ caused by expanding b_i for N_i times, and N_i represents the number of expanding b_i. Both I_i and N_i are recalculated at the beginning of each policy-construction step. The overhead of computing C_i is negligible in comparison to the Expand() and UpdateAncestors() procedures. Here, b_U (or b_L) refers to all belief nodes generated when $C_U H_U > C_L H_L$ (or otherwise), and expanding b_i once means calling Expand(b_i) and UpdateAncestors(b_i) once. The ones appearing in both numerator and denominator are used to make C_U and C_L equal 1 at the beginning of the online algorithm's run. C_U and C_L are variables for adjusting the importance of $H_U(b)$ and $H_L(b)$, respectively, in heuristic search. For example, imagine the expansions of b_U (in a period) brought no or only small change in both $\underline{V}(b_c)$ and

$\bar{V}(b_c)$. As time goes on, the value of C_U will decrease, and the probability of selecting b_U to expand will also drop.

Note that there are many other possible hybrid methods that combine $H_L(b)$ and $H_U(b)$. Here is only one feasible way. A further research in this direction may lead to a more efficient hybrid method.

3.3 Leveraging the MOMDP Representation

Now, we analyze why the MOMDP representation is a useful tool to reduce the time complexity of the operations related with belief-state computation and value-function computation. Reconsidering Equations 1, 2 and 4 in the context of POMDPs, we can see that computing $b^{a,z}$, $P(z|b,a)$ and $V(b)$ takes $|\mathcal{X}|^2|\mathcal{Y}|^2 + 2|\mathcal{X}||\mathcal{Y}|$, $|\mathcal{X}|^2|\mathcal{Y}|^2$ and $|\Gamma||\mathcal{X}||\mathcal{Y}|$ multiplications, respectively. However, by applying the MOMDP-specific versions of these equations—Equations 9, 10 and 11—these costs can be reduced to $2|\mathcal{Y}|^2 + 2|\mathcal{Y}|$, $2|\mathcal{Y}|^2$ and $|\Gamma_{\mathcal{Y}}(x)||\mathcal{Y}|$ multiplications, respectively. That is, the time complexity of computing $b^{a,z}$, $P(z|b,a)$ and $\underline{V}(b)$ can be reduced by $\frac{|\mathcal{X}|^2|\mathcal{Y}|^2+2|\mathcal{X}||\mathcal{Y}|}{2|\mathcal{Y}|^2+2|\mathcal{Y}|}(= \mathcal{O}(|\mathcal{X}|^2))$, $\frac{|\mathcal{X}|^2}{2}$ and $\frac{|\mathcal{X}||\Gamma|}{|\Gamma_{\mathcal{Y}}(x)|}$ times. Since $|\Gamma| = \sum_{x\in\mathcal{X}} |\Gamma_{\mathcal{Y}}(x)|$, the expectation $E_{x\sim\mathcal{X}}[\frac{|\Gamma|}{\Gamma_{\mathcal{Y}}(x)}]$ is $|\mathcal{X}|$. Thus, the MOMDP representation reduces each of the three operations' complexity by a factor of $\mathcal{O}(|\mathcal{X}|^2)$.

3.4 Convergence of FHHOP

The new online heuristic search algorithm using the hybrid heuristics in Section 3.2 and the POMDP representation in Section 3.3 is the FHHOP algorithm. Note that there are substantial differences between our hybrid strategy and the strategy included in AEMS1. AEMS1 merges $\underline{V}$ and $\bar{V}$ at the beginning of designing the action selection strategy in its heuristic function. However, FHHOP first separates $\underline{V}$ and $\bar{V}$ in order to define two almost independent heuristic functions, then merges the two heuristic functions with the goal of taking full advantage of each heuristic function. As a final note, Proposition 1 states the convergence property of the FHHOP algorithm.

Proposition 1. *Let $\epsilon > 0$ and b_c the current belief state. Then, the FHHOP algorithm is guaranteed to find an ϵ-optimal action for b_c within finite time.*

Proof. Because the FHHOP algorithm exploits the MOMDP representation to accelerate the computation of $b^{a,z}$, $P(z|b,a)$ and $\underline{V}(b)$ without losing accuracy, following Theorem 2 in (Ross et al. 2008b), we only need to prove that the probability of choosing b_U to expand in the FHHOP algorithm is always greater than 0 in this algorithm. The following is the proof by contradiction.

Suppose that after the M^{th} expansion of b_U or b_L, the FHHOP algorithm never selects b_U to expand, namely, $C_U H_U(b_U) \leq C_L H_L(b_L)$. Thus, I_U and N_U will never change, and therefore, C_U and $H_U(b_U)$ will never change after the M^{th} expansion. Let $\underline{V}_0(b)$ and $\bar{V}_0(b)$ be b's initialized lower and upper bounds, and $e_0(b) = \bar{V}_0(b) - \underline{V}_0(b)$. Then, we have

$$H_L(b_L) \leq e(b_L) \leq \max_{b\in\mathcal{B}} e_0(b),$$

and $I_L \leq e_0(b_c)$. Now, let

$$N_L > \max\{M, \frac{[e_0(b_c)+1]\max_{b\in\mathcal{B}} e_0(b)}{C_U H_U(b_U)} - 1\}.$$

Then,

$$\begin{aligned} C_U H_U(b_U) &> \frac{[e_0(b_c)+1]\max_{b\in\mathcal{B}} e_0(b)}{N_L+1} \\ &\geq \frac{I_L+1}{N_L+1} H_L(b_L) \\ &= C_L H_L(b_L). \end{aligned}$$

There is a contradiction. □

4 Experimental Results

In this section, we present detailed empirical results designed to test the performance of the FHHOP algorithm. In all test domains, we use the blind policy and the FIB method (Hauskrecht 2000) to initialize the lower and upper bounds, respectively. Except where stated otherwise, all experiments were run on an AMD dual core processor 3600+ 2.00GHz with 2GB memory. We implemented FHHOP and AEMS2 using C++, within APPL-0.93[1], an efficient POMDP solver. Results about SARSOP without attribution appeared in Table 1 were also obtained using APPL-0.93 in our experimental platform.

4.1 Benchmark Problems

a) *Hallway:* The Hallway domain (Littman et al. 1995) models a robot navigating in an office environment. The task of the robot in this problem is to arrive in a single goal location. The state of the robot is comprised of its current position and its current orientation. However, neither attribute is fully observable. There is a set of 5 actions for movements:{Forward, Turn-left, Turn-right, Turn-around, Stay-in-place}. The robot can partially observe the existence of walls in four directions using four independent, short-range, and noisy sensors.

b) *Tag:* The Tag problem was first described in (Pineau et al. 2003). In this environment, a mobile robot moves in a grid map with 29 positions with the goal of tagging an

[1]The software package is available from the web-page: http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/. All test problems in the MOMDP representation conform to the syntax of pomdpx specified on the same web site.

opponent that intentionally moves away. Both the robot and the opponent are located initially in independently selected random positions. The robot can choose either to move into one of four adjacent positions by actions {North, South, East, West} or to tag the opponent by a "Catch" action. The effect of the robot's action is deterministic. The robot can know its current position exactly, but the opponent's position is not observable for the robot unless they are in the same position. The robot tries to catch the opponent as quickly as possible to receive a good reward since each move for the robot is expensive. In the MOMDP representation for Tag, the fully observable state variable x represents the robot's position. The partially observable state variable y is the opponent's position.

c) *RockSample:* The RockSample domain was originally presented in (Smith and Simmons 2004) and has been frequently used in recent work to test new POMDP algorithms (Ross et al. 2008a; Bonet and Geffner 2009; Ong et al. 2010; Silver and Veness 2010). This domain models a planetary robot that has to explore an area represented as a grid map and sample rocks with a scientific value. The robot receives a positive or negative reward depending on whether the sampled rock has a scientific value. RockSample_n_k is a family of problems in the RockSample domain with a map size $n \times n$ and k rocks. In these problems, the robot's action set consists of $k + 5$ actions: {North, South, East, West, Sample, Check$_1$,... Check$_k$}. The robot always knows its own position and rock positions in the map exactly. That is rock locations are fixed and encoded in the map and need not be encoded in state variables. However, whether rock m has a scientific value is only partially observed via the action Check$_m$. The Check$_m$ action is responsible for gathering information about rock m using a noisy long-range sensor. The accuracy of the information gathered depends on the distance between the robot and the rock checked. In the MOMDP representation for this domain, the x variable is the robot's position, and the y variable is a Boolean that indicates whether a particular rock has a scientific value.

d) *FieldVisionRockSample:* The FieldVisionRockSample domain was initially introduced in the work on AEMS2 (Ross and Chaib-draa 2007). This domain can be viewed as a variant of the RockSample domain. The only difference between them is the way of perceiving the rocks in the environment. In contrast to perform a check action on a specific rock to observe its state in RockSample, in FieldVisionRockSample the states of all rocks are observed after any action by the same noisy sensor. Consequently, the robot can only perform 5 actions: {North, South, East, West, Sample}, and has an observation space with size 2^k for the problem with k rocks. What is different from the RockSample domain is that, in the MOMDP model for this task, the y variable represents k Boolean values, each of which indicates whether a rock has a scientific value.

e) *AUV Navigation:* The AUV Navigation problem first appeared in (Ong et al. 2009). This problem models a 3-D oceanic environment with 4 levels and a 7×20 grid map at each level. An autonomous underwater vehicle (AUV) can achieve reward by navigating from some uncertain starting position to some goal positions and avoiding rock formations. The AUV can perform 6 actions: {Forward, Stay, Left, Right, Up, Down}. All of them are stochastic due to control uncertainty or ocean currents. The AUV is equipped with an accurate pressure senor, an accurate compass, and a global positioning system (GPS), whose signals can only be received when the AUV is located at the surface level. In the MOMDP representation, the x variable models the AUV's depth and orientation, and the y variable models the AUV's horizonal location.

These benchmark problems were selected to illustrate specific characteristics. First, the Tag problem has a dynamic environment that changes over time even if the robot does not take any action. Second, the RockSample domain has a larger action space but smaller observation space, while the other domains have smaller action spaces but larger observation spaces. Third, the Hallway problem does not include the mixed observation structure. Fourth, all test problems except Hallway have large state spaces ranging from about 10^3 to 10^5 dimensions. Thus, we hope including problems with different features helps test FHHOP's overall performance thoroughly.

4.2 Performance Comparison

We applied FHHOP and existing state-of-the-art online and offline algorithms to these benchmark problems. Table 1 compares them in terms of the following characteristic metrics (Ross et al. 2008a): expected discounted reward (Reward), upper bound of online time per policy-construction step (τ) (seconds), offline time (seconds). Looking at the comparison between our implementation of AEMS2 and FHHOP, we can find that FHHOP obtains higher values in terms of reward on all test problems with online time less or equal to AEMS2's. Furthermore, at least on the Tag, RockSample and AUVNavigation domains, FHHOP has achieved more than an order of magnitude improvement compared to AEMS2. These empirical results suggest that FHHOP tends to have a better scalability especially in large-scale POMDP domains.

Previously published results for SARSOP, except those on the Hallway and FieldVisionRockSample domains from our experimental platform, are provided in the table for comparison. Through the comparison can be found that the FHHOP algorithm is very competitive, sometimes even better, in terms of expected discounted reward, and meanwhile have advantages such as a small initial offline planning time and a small re-planning time between action

Table 1: Multi-algorithm performance comparison on several benchmark problems. See text for explanations.

Method	Reward	Online Time τ	Offline Time	Method	Reward	Online Time τ	Offline Time
Hallway				**Tag**			
$(\|S\|=61, \|\mathcal{X}\|=1, \|\mathcal{Y}\|=61, \|A\|=5, \|Z\|=21)$				$(\|S\|=870, \|\mathcal{X}\|=30, \|\mathcal{Y}\|=29, \|A\|=5, \|Z\|=30)$			
AEMS2	0.50±0.01	0.20	0.02	AEMS2	-6.51±0.14	1.00	0.28
FHHOP	**0.52±0.01**	**0.20**	**0.02**	**FHHOP**	**-6.04±0.14**	**0.10**	**0.12**
SARSOP	0.52±0.01	0.00	1.05	SARSOP(Ong)	-6.03±0.12	0.00	16.50
RockSample_7_8				**RockSample_11_11**			
$(\|S\|=12545, \|\mathcal{X}\|=50, \|\mathcal{Y}\|=256, \|A\|=13, \|Z\|=2)$				$(\|S\|=247809, \|\mathcal{X}\|=122, \|\mathcal{Y}\|=2048, \|A\|=16, \|Z\|=2)$			
AEMS2	20.66±0.29	1.00	3.34	AEMS2	21.11±0.28	10.00	56.81
FHHOP	**21.45±0.30**	**0.10**	**0.51**	**FHHOP**	**21.49±0.32**	**1.00**	**7.97**
POMCP(Silver)	20.71±0.21	1.00	n.v.	POMCP(Silver)	20.01±0.23	1.00	n.v.
SARSOP(Ong)	21.39±0.01	0.00	810.00	SARSOP(Ong)	21.56±0.11	0.00	1369.00
FieldVisionRockSample_5_5				**FieldVisionRockSample_5_7**			
$(\|S\|=801, \|\mathcal{X}\|=26, \|\mathcal{Y}\|=32, \|A\|=5, \|Z\|=32)$				$(\|S\|=3201, \|\mathcal{X}\|=26, \|\mathcal{Y}\|=128, \|A\|=5, \|Z\|=128)$			
AEMS2	21.02±0.32	1.00	0.10	AEMS2	22.15±0.32	1.00	0.71
FHHOP	**22.56±0.33**	**0.20**	**0.02**	**FHHOP**	**24.46±0.34**	**0.20**	**0.12**
SARSOP	23.20±0.33	0.00	508.40	SARSOP	29.48±0.34	0.00	1029.38
RockSample_10_10				**AUV Navigation**			
$(\|S\|=102401, \|\mathcal{X}\|=101, \|\mathcal{Y}\|=1024, \|A\|=15, \|Z\|=2)$				$(\|S\|=13536, \|\mathcal{X}\|=96, \|\mathcal{Y}\|=141, \|A\|=6, \|Z\|=144)$			
AEMS2	20.78±0.27	10.00	11.24	AEMS2	1034.89±8.36	100.00	83.21
FHHOP	**21.35±0.34**	**1.00**	**2.82**	**FHHOP**	**1059.22±8.49**	**10.00**	**16.02**
SARSOP(Ong)	21.47±0.11	0.00	1589.00	SARSOP(Ong)	1019.80±9.70	0.00	409.00

n.v.=not available (Ong)= (Ong et al. 2009) (Silver)= (Silver and Veness 2010)

executions. Note that there also exists other state-of-the-art offline algorithms (Smith and Simmons 2004; Pineau et al. 2006; Shani et al. 2007; Bonet and Geffner 2009). Here, we only use SARSOP as a representative offline algorithm to compare with our online algorithms.

In addition, published results on the partially observable Monte-Carlo planning (POMCP) algorithm (Silver and Veness 2010) for some RockSample problems are also presented. POMCP is a promising online Monte-Carlo algorithm, especially in large-scale POMDP problems. From these data, we can see FHHOP exceeds POMCP in terms of both overall running time and qualities of generated policies on these RockSample problems.

5 Conclusion and Future Work

This paper presents FHHOP, a novel factored hybrid heuristic online planning algorithm for large POMDPs. To the best of our knowledge, FHHOP is the most efficient online heuristic search algorithm, on the whole, yet proposed amongst existing fully implemented domain-independent online POMDP solvers. A major contribution of this algorithm is a new hybrid heuristic search strategy that takes full advantage of both lower and upper bounds on the optimal value function. Its foundation stone is a clever way of constructing a heuristic function using the lower bound. A minor contribution of this algorithm is to integrate MOMDP, a recently developed factored state representation, with a cutting-edge online algorithm and reveal its efficacy empirically. Experimental results reported here provide a comprehensive picture of FHHOP and state-of-the-art online and offline approaches on five popular benchmark domains.

There are some interesting directions for future work. First, we noticed some efficient structure learning algorithms had been proposed in factored state MDPs (Li et al. 2008; Diuk et al. 2009). An interesting research direction is to extend these existing algorithms to automatically learn the mixed observation structure in POMDPs. Second, we believe it should be possible to use other cutting-edge data structures, such as algebraic decision diagrams (Poupart 2005) or topology (Brunskill and Russell 2010), in FHHOP to succinctly represent more complex structures. Third, reparameterization (Ong et al. 2009) may give POMDPs without the mixed observation structure a chance to benefit from the MOMDP representation. Fourth, we could leverage the graph structure instead of the AND/OR tree in FHHOP to avoid redundant computations on duplicate belief states. Finally, we would also like to investigate further hybrid heuristic strategies for better performance.

Acknowledgments

We thank Michael L. Littman, David Hsu, Stéphane Ross, Trey Smith and anonymous reviewers for their thoughtful suggestions. This work was supported in part by the China Scholarship Council, the National Science Foundation of China under Grant Nos. 60745002, 61175057, 61105039, and the National High-Tech Research and Development Plan of China under Grant No. 2008AA01Z150.

References

Bellman, R. 1957. Dynamic programming. *Princeton University Press, Princeton, NJ, USA*.

Bonet, B., and Geffner, H. 2009. Solving POMDPs: RTDP-Bel vs. point-based algorithms. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-2009)*, 1641–1646.

Brunskill, E., and Russell, S. 2010. Rapid: A reachable anytime planner for imprecisely-sensed domains. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI-2010)*.

Diuk, C.; Li, L.; and Leffler, B. R. 2009. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML-2009)*, 249–256.

Hauskrecht, M. 2000. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research* 13:33–94.

Hoey, J.; Von Bertoldi, A.; Poupart, P.; and Mihailidis, A. 2007. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *Proceedings of the 5th International Conference on Vision Systems (ICVS-2007)*.

Hsu, D.; Lee, W. S.; and Rong, N. 2008. A point-based POMDP planner for target tracking. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-2008)*, 2644–2650.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics: Science and Systems (RSS-2008)*.

Li, L.; Littman, M. L.; and Walsh, T. J. 2008. Knows what it knows: A framework for self-aware learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML-2008)*, 568–575.

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Learning policies for partially observable environments: Scaling up. In *Proceedings of International Conference on Machine Learning (ICML-1995)*, 362–370.

Madani, O.; Hanks, S.; and Condon, S. 1999. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI-1999)*, 541–548.

Ong, S. C.; Png, S. W.; Hsu, D.; and Lee, W. S. 2009. POMDPs for robotic tasks with mixed observability. In *Proceedings of Robotics: Science and Systems (RSS-2009)*.

Ong, S. C.; Png, S. W.; Hsu, D.; and Lee, W. S. 2010. Planning under uncertainty for robotic tasks with mixed observability. *International Journal of Robotics Research*.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, 1025–1032.

Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research* 27:335–380.

Poupart, P. 2005. Exploiting structure to efficiently solve large scale POMDPs. *Ph.D. Dissertation, University of Toronto*.

Ross, S., and Chaib-draa, B. 2007. AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-2007)*, 2592–2598.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-Draa, B. 2008a. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research* 32:663–704.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-Draa, B. 2008b. Theoretical analysis of heuristic search methods for online POMDPs. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NIPS-2008)*, 1233–1240.

Satia, J. K., and Lave, R. E. 1973. Markovian decision processes with probabilitic observation of state. *Management Science* 20(1):1–13.

Shani, G.; Brafman, R. I.; and Shimony, S. E. 2007. Forward search value iteration for POMDPs. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-2007)*, 2619–2624.

Silver, D., and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS-2010)*, 2164–2172.

Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21(5):1071–1088.

Smith, T., and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-2004)*, 520–527.

Washington, R. 1997. BI-POMDP: bounded, incremental partially observable Markov model planning. In *Proceedings of the 4th European Conference on Planning*, 440–451.

Guess Who Rated This Movie: Identifying Users Through Subspace Clustering

Amy Zhang
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, MA
amyzhang@mit.edu

Nadia Fawaz, Stratis Ioannidis
Technicolor
Palo Alto, CA
nadia.fawaz@technicolor.com
stratis.ioannidis@technicolor.com

Andrea Montanari
Departments of Statistics
and Electrical Engineering
Stanford University
Stanford, CA
montanari@stanford.edu

Abstract

It is often the case that, within an online recommender system, multiple users share a common account. Can such shared accounts be identified *solely on the basis of the user-provided ratings*? Once a shared account is identified, can the different users sharing it be identified as well? Whenever such user identification is feasible, it opens the way to possible improvements in personalized recommendations, but also raises privacy concerns. We develop a model for composite accounts based on unions of linear subspaces, and use subspace clustering for carrying out the identification task. We show that a significant fraction of such accounts is identifiable in a reliable manner, and illustrate potential uses for personalized recommendation.

1 Introduction

Online commerce services such as Netflix provide personalized recommendations by collecting user ratings about a universe of items, to which we refer here as 'movies'. Typically, multiple people within a single household (family members, roommates, *etc.*) may share the same account for both viewing and rating movies. Service providers are avoid deploying multiple accounts as log-in screens are perceived as a nuisance and a barrier to using the service. This is especially true on a keyboard-less devices, such as televisions or gaming platforms. Account sharing persists even when providers offer the option of registering secondary accounts, as the latter may have access to a subset of the services enjoyed by the primary account. Finally, sharing might be regarded as a partial (if unconscious) privacy protection mechanism, hindering the release of the household's composition and demographics.

The use of a single account by multiple individuals poses a challenge in providing accurate personalized recommendations. Informally, the recommendations provided to a "composite" account, comprising the ratings of two dissimilar users, may not match the interests of either of these users. More concretely, as discussed in Section 3, collaborative filtering methods such as matrix factorization assume that ratings follow a linear model of user and movie profiles of small dimension. Though such methods may perform well for most cases, they can fail on composite accounts, as we show in Section 6. This is because "mixing" ratings from different users may yield a rating set that can no longer be explained by a linear model.

Can composite accounts within a recommender system be identified? Can the individuals sharing such an account be identified? Can accurate profiles of different users' behaviors be learnt? We address these questions in the most challenging setting, namely when no information is available apart from the ratings users provide. Our contributions are as follows:

(a) We develop a model of composite accounts as unions of linear subspaces. This allows us to apply a number of *linear subspace clustering* algorithms (Ma et al., 2008) to the present problem.

(b) Based on this model, we develop a statistical test that can be used as indicator of 'compositeness', and a model selection procedure to determine the number of users sharing the same account. We systematically apply and evaluate these methods on real datasets.

(c) In particular, we show that a significant fraction of composite accounts can be reliably identified. In a dataset made of both single-user and composite accounts, a subset S of accounts can be selected that comprises roughly 70% of the composite accounts, while only 40% of the accounts in S are single-user accounts.

(d) The users sharing an account can be identified

with good accuracy. For the accounts in the above set, more than 60% of the movies were identified correctly (a result that we estimate to be significant with $p < 0.05$).

(e) We apply these mechanisms on 54K Netflix users that rated more than 500 movies, and identify 4 072 composite users with high confidence.

(f) Finally, we demonstrate how the above methods can be applied to improve recommendations.

We consider this ability to identify multiple users behind an account quite surprising, in view that no information is used apart from users' ratings. In particular, *all publicly available datasets are susceptible to this identification*. Beyond personalized recommendations, this ability is useful/worrisome for a number of reasons. On one hand, it can aid in determining the household's demographics. Such information can be subsequently monetized, *e.g.*, through targeted advertising. On the other hand, user identification can be considered as a privacy breach, and calls for a careful privacy assessment of recommender systems.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 develops our statistical model. Sections 4 and 5 apply and evaluate our new methods to recommender system datasets. Finally, Section 6 uses these methods to improve personalized recommendations.

2 Related Work

The problem of user identification from ratings has received attention only recently. The 2nd Challenge on Context-Aware Movie Recommendation (Said et al., 2011) addressed a "supervised" variant. Movie ratings generated by users in the same household as well as the ids of the users was provided as a training set. The test set included movie ratings attributed to households, and contestants were asked to predict which household members rated these movies. In contrast, we study an unsupervised version of the problem, where the mapping of movies to users is not a priori known.

To the best of our knowledge, we are the first to study user identification as a subspace clustering problem. Beyond EM and GPCA, several subspace clustering algorithms have been recently proposed (Elhamifar and Vidal, 2009; Liu et al., 2010; Soltanolkotabi and Candes, 2011; Eriksson et al., 2011). Preliminary simulations using these methods did not yield significant improvements.

3 Statistical Modeling

Consider a dataset of ratings on M movies provided by N accounts, each corresponding to a different household. Ratings are available for a subset of all $N \times M$ possible pairs: we denote by $\mathcal{M}_H \subseteq [M]$, where $m_H \equiv |\mathcal{M}_H|$, the set of movies rated by account/household H, and by $r_{Hj} \in \mathbb{R}$ the rating of movie $j \in \mathcal{M}_H$.

Each movie $j \in [M]$ is associated with a feature vector $\mathbf{v}_j \in \mathbb{R}^d$, where $d \ll N, M$. We use matrix factorization to extract the latent features for each movie, as described in Section 3.3. If explicit information (*e.g.*, genres or tags) is available, this can be easily incorporated in our model by extending the vectors $\mathbf{v}_j$.

Each household H may comprise one *or more* users that actually rated the movies in $\mathcal{M}_H$. Abusing notation, we denote by H the set of users in this household, and by $n_H = |H|$ the household size. For each $i \in H$, we denote by $A_i^* \subseteq \mathcal{M}_H$ the set of movies rated by i, and by $I^*(j) \in H$ the user that rated $j \in \mathcal{M}_H$.

Note that neither the household size n_H nor the mapping $I^* : \mathcal{M}_H \to H$ are a priori known. We would like to perform the following inference tasks.

(a) *Model Selection*: determine the household size n_H. A closely related problem is the one of determining whether the account is *composite* (*i.e.*, $|H| > 1$) or not.

(b) *User Identification:* identify movies that have been viewed by the same user—*i.e.*, recover I^*, up to a permutation, and use this knowledge to profile the individual users.

We also explore the impact of user identification on targeted recommendations. The 'dual' impact on user privacy will be the object of a forthcoming publication.

3.1 Linear Model

We focus now on a single household, and omit the index H hereafter. We thus denote by n the household size, $\mathcal{M}$ and m the set of movies rated by this household and its size, respectively, and by r_j the rating given to movie $j \in \mathcal{M}$.

Our main modeling assumption is that the rating r_j generated by a user $i \in H$ for a movie $j \in \mathcal{M}$ is determined by a linear model over the feature vector $\mathbf{v}_j$. That is, for each $i \in H$ there exists a vector $\mathbf{u}_i^* \in \mathbb{R}^d$ and a real number $z_i^* \in \mathbb{R}$ (the *bias*), such that

$$r_j = \langle \mathbf{u}_i^*, \mathbf{v}_j \rangle + z_i^* + \epsilon_j, \quad \text{for all } j \in A_i^*, i \in H, \quad (1)$$

where $\epsilon_j \in \mathbb{R}$ are i.i.d. Gaussian random variables with mean zero and variance σ^2. Such linear models are

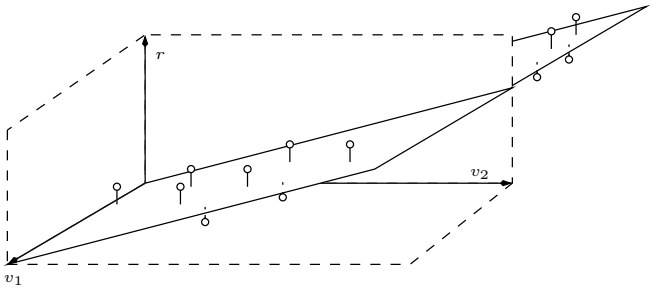


Figure 1: For all movies $j \in A_i$ rated by user $i \in H$, the points $\mathbf{x}_j = (\mathbf{v}_j, 1, r_j) \in \mathbb{R}^{d+2}$ lie slightly off a hyperplane whose normal is $(\mathbf{u}_i, z_i, -1) \in \mathbb{R}^{d+2}$.

used extensively by rating prediction methods that rely on matrix factorization (Srebro and Jaakkola, 2003; Srebro et al., 2005; Koren et al., 2009), and are known to perform very well in practice.

Assuming that the household size is known, the model parameters of (1) are (a) the *user profiles* $\boldsymbol{\Theta}^* = \{\boldsymbol{\theta}_i^*\}_{i \in H} \in \mathbb{R}^{n \times d+1}$, where $\boldsymbol{\theta}_i^* = (\mathbf{u}_i^*, z_i^*) \in \mathbb{R}^{d+1}$, $i \in H$, as well as (b) the mapping $I^* : \mathcal{M} \to H$. Given two estimators $\boldsymbol{\Theta}, I$ of $\boldsymbol{\Theta}^*, I^*$, the log-likelihood of the observed sequence of pairs $\{(\mathbf{v}_j, r_j)\}_{j \in \mathcal{M}}$, is given by

$$L(\boldsymbol{\Theta}, I) = -\frac{1}{2\sigma^2} \sum_{j \in \mathcal{M}} \big(r_j - z_{I(j)} - \langle \mathbf{u}_{I(j)}, \mathbf{v}_j \rangle\big)^2. \quad (2)$$

Estimating the maximum likelihood model parameters thus amounts to minimizing the mean square error:

$$\min_{\boldsymbol{\Theta}, I} \ \mathrm{MSE}(\boldsymbol{\Theta}, I) = \frac{1}{m} \sum_{j \in \mathcal{M}} \big(r_j - z_{I(j)} - \langle \mathbf{u}_{I(j)}, \mathbf{v}_j \rangle\big)^2, \quad (3)$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{n \times d+1}$, $I \in \mathcal{I}$, the set of all mappings from $\mathcal{M}$ to H. Note that (3) is not convex. Nevertheless, as discussed in Section 4.1, fixing I results in a quadratic program, while fixing $\boldsymbol{\Theta}$ results in a combinatorial problem solvable in $O(nm)$ time.

3.2 Subspace Arrangements

We obtain an insightful geometric interpretation of the minimization (3) by studying the points $\mathbf{x}_j = (\mathbf{v}_j, 1, r_j) \in \mathbb{R}^{d+2}$, *i.e.*, the $d+2$-dimensional vectors resulting from appending $(1, r_j)$ to the movie profiles. Eq. (1) implies that although the points x_j live in an ambient space of dimension $d+2$, they actually lie on a lower-dimensional manifold: the union of n hyperplanes, *i.e.*, $d+1$-dimensional linear subspaces of $\mathbb{R}^{d+2}$.

To see this, let $\mathbf{n}_i^* = (\mathbf{u}_i, z_i, -1) \in \mathbb{R}^{d+2}$ be the vector obtained by appending the bias z_i^* and -1 to $\mathbf{u}_i^*$. Then, $|\langle \mathbf{n}_i^*, x_j \rangle| = |\langle \mathbf{u}_i^*, \mathbf{v}_j \rangle + z_i^* - r_j| = |\epsilon_j|$, for every $j \in A_i$. Hence, provided that the variance σ^2 is small, the points $\mathbf{x}_j$ lie very close to the hyperplane with normal $\mathbf{n}_i^*$ that crosses the origin (see Figure 1).

A union of such affine subspaces is called a *subspace arrangement*. Given that the data x_j, $j \in \mathcal{M}$, "almost" lie on such a manifold, minimizing the MSE has the following appealing geometric interpretation. First, mapping a movie j to a user amounts to identifying the hyperplane to which x_j is closest to. Second, once movies are thus mapped to users, profiling a user amounts to computing the normal to its corresponding hyperplane. Finally, identifying the number of users in a household amounts to determining the number of hyperplanes in the arrangement.

These tasks are known collectively as the *subspace estimation* or *subspace clustering* problem, which has numerous applications in computer vision and image processing (Vidal, 2010). In Section 4.1, we exploit this connection to apply algorithms for subspace clustering on user identification (namely, EM and GPCA).

3.3 Datasets

We test our algorithms on two datasets:

CAMRa2011 dataset. The CAMRa2011 dataset was released at the Context-Aware Movie Recommendation (CAMRa) challenge at the 5th ACM International Conference on Recommender Systems (RecSys) 2011. This dataset consists of 4 536 891 5-star ratings provided by $N = 171\,670$ users on $M = 23\,974$ movies, as well as additional information about household membership for a subset of 602 users. The 290 households comprise 272, 14 and 4 households of size 2, 3 and 4 users, respectively. We use the entire dataset to compute the movie profiles $\mathbf{v}_j$ through matrix factorization, using $d = 10$ (found to be optimal through cross validation). In the sequel, we restrict our attention to the 544 users belonging to households of size 2. To simulate a composite account, we merge the ratings provided by users belonging to the same household. The original mapping of ratings to household members serves as the ground truth.

Netflix Dataset. The second dataset contains 5-star ratings given by $N = 480\,189$ users for $M = 17\,770$ movies. We again obtain the movie profiles $\mathbf{v}_j$ through matrix factorization on the entire dataset, with $d = 30$. We then restrict our attention to the subset of 54 404 users who rated at least 500 movies. We also generate 300 'synthetic' households of size 2 by pairing the ratings of 600 randomly selected users; we select these among the accounts that our model-selection methods, described in Section 5.3, classify as non-composite.

Matrix factorization is likely to be unreliable for extracting account feature vectors, as the latter may be

composite. On the other hand, it appears to perform well for movies. We use the OptSpace algorithm of Keshavan et al. (2010) in both datasets for matrix factorization, which will not be further discussed.

4 User Identification

In this section, we address the user identification problem assuming that the household size n is a priori known. This amounts to obtaining estimators of I^* and $\boldsymbol{\theta}_i^* = (\mathbf{u}_i^*, z_i^*)$ for each user $i \in H$. We first describe four algorithms for solving this problem and then evaluate them on our two datasets. We present methods for determining the size n in Section 5.

In the absense of any additional information, we cannot distinguish between two mappings $I : \mathcal{M} \to H$ that partition $\mathcal{M}$ identically. As such, we have no hope of identifying the correct "label" $i \in H$ of a user; we thus assume in the sequel, w.l.o.g., that $H = [1, \ldots, n]$.

4.1 Algorithms

Clustering. Our first approach consists of two steps. First, we obtain a mapping $I : \mathcal{M} \to [n] = H$ by clustering the rating events $(\mathbf{v}_j, r_j) \in \mathbb{R}^{d+1}$, $j \in \mathcal{M}$ into n clusters. Second, given I, we estimate $\boldsymbol{\theta}_i = (\mathbf{u}_i, z_i)$, $i \in [n]$, by solving the quadratic program:

$$\min_{\boldsymbol{\Theta}} \mathrm{MSE}(\boldsymbol{\Theta}, I), \tag{4}$$

where MSE is given by (3). This is separable in each $\boldsymbol{\theta}_i$, so the latter can be obtained by solving

$$\min_{(\mathbf{u}_i, z_i)} \sum_{j \in A_i} (r_j - \langle \mathbf{u}_i, \mathbf{v}_j \rangle - z_i)^2. \tag{5}$$

where $A_i = \{j \in \mathcal{M} : I(j) = i\}$, which amounts to linear regression w.r.t. the model (1).

We perform the clustering in the first step using either (a) K-means or (b) spectral clustering. Each yields a distinct mapping; we denote the resulting two user identification algorithms by K-Means and Spectral, respectively. Intuitively, these methods treat the rating as "yet another" feature, and tend to attribute movies with very similar profiles $\mathbf{v}$ to the same user, even if they receive quite distinct ratings.

Expectation Maximization. The EM algorithm (Dempster et al., 1977) identifies the parameters of mixtures of distributions. It naturally applies to subspace clustering—technically, this is "hard" or "Viterbi" EM. Proceeding over multiple iterations, alternately minimizing the MSE in terms of the movie-user mapping I and the user profiles $\boldsymbol{\Theta}$. Initially, a mapping $I^0 \in \mathcal{I}$ is selected uniformly at random; at step $k \geq 1$, the profiles and the mapping are computed as follows.

$$\boldsymbol{\Theta}^k = \operatorname*{arg\,min}_{\boldsymbol{\Theta} \in \mathbb{R}^{n \times (d+1)}} \mathrm{MSE}(\boldsymbol{\Theta}, I^{k-1}) \tag{6a}$$

$$I^k = \operatorname*{arg\,min}_{I \in \mathcal{I}} \mathrm{MSE}(\boldsymbol{\Theta}^k, I) \tag{6b}$$

The minimization in (6a) can be solved as in (4) through linear regression. Eq. (6b) amounts to identifying the profile that best predicts each rating, *i.e.*,

$$I^k(j) = \arg\min_{i \in H} (r_j - z_i^k - \langle \mathbf{u}_i^k, \mathbf{v}_j \rangle)^2, \quad j \in \mathcal{M}. \tag{7}$$

which can be computed in $O(nm)$ time.

Generalized PCA. The Generalized Principal Components Analysis (GPCA) algorithm, originally proposed by Vidal et al. (2005), is an algebraic-geometric algorithm for solving the general subspace clustering problem, as defined in section 3.2.

To give some insight on how GPCA works, we consider first an idealized case where the noise ϵ_j in the linear model (1) is zero. Then, the points $\mathbf{x}_j = (\mathbf{v}_j, 1, r_j)$, $j \in A_i^*$, lie exactly on a hyperplane with normal $\mathbf{n}_i^* = (\mathbf{u}_i^*, z_i^*, -1)$. Thus, every $\mathbf{x}_j$, $j \in \mathcal{M}$, is a root of the following homogeneous polynomial of degree n:

$$\begin{aligned} P_{\mathbf{c}}(\mathbf{x}) &= \prod_{i \in H} \langle \mathbf{n}_i^*, \mathbf{x} \rangle = \prod_{i \in H} \sum_{k=1}^{d+2} n_{ik}^* x_{jk} \\ &= \sum_{k_1 + \ldots + k_{d+2} = n, \forall l \;\; k_\ell \geq 0} c_{k_1, \ldots, k_{d+2}} x_1^{k_1} \ldots x_{d+2}^{k_{d+2}} \end{aligned} \tag{8}$$

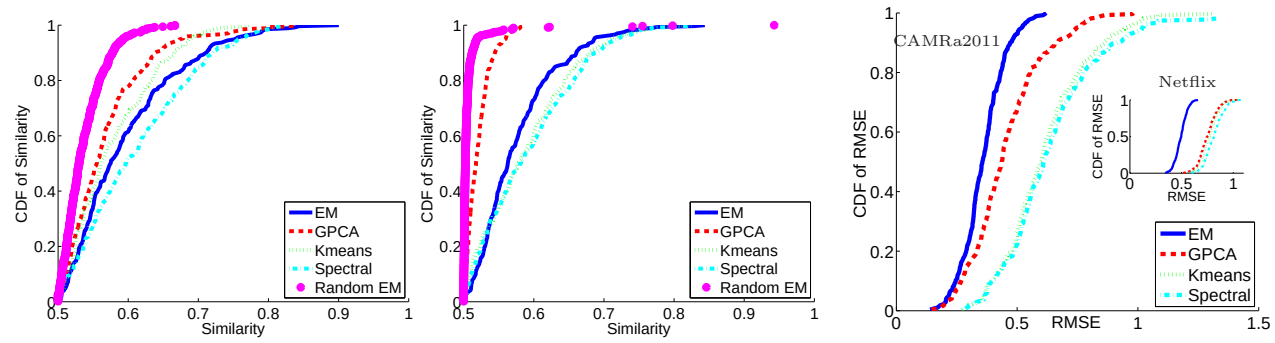
We denote by $\mathbf{c} \in \mathbb{R}^{K(n,d)}$, where $K(n,d) = \binom{n+d+1}{n}$, the vector of the monomial coefficients $c_{k_1, \ldots, k_{d+2}}$. Note that $P_{\mathbf{c}}$ is uniquely determined by $\mathbf{c}$. Moreover, provided that $m = |\mathcal{M}| \geq K(n,d) = O(\min(n^d, d^n))$, $\mathbf{c}$ can be computed by solving the system of linear equations $P_{\mathbf{c}}(\mathbf{x}_j) = 0$, $j \in \mathcal{M}$.

Knowledge of $\mathbf{c}$ can be used to *exactly recover* I^*, up to a permutation. This is because, by (8), for any $j \in A_i^*$, the gradient $\nabla P_{\mathbf{c}}(x_j)$ is proportional to the normal $\mathbf{n}_i^*$. Hence, the partition in of points $\{A_i^*\}$ can be recovered by grouping together points with co-linear gradients (Vidal et al., 2005).

Unfortunately, this result does not readily generalize in the presence of noise (see, *e.g.*, Ma et al. (2008)). In this case, one approach is to estimate $\mathbf{p}$ by solving the (non-convex) optimization problem

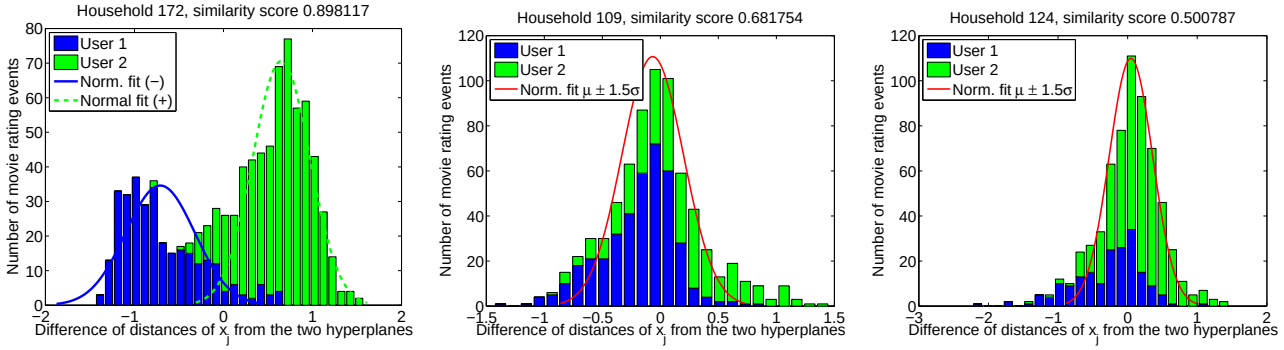
$$\begin{aligned} \text{Minimize:} \quad & \sum_{j \in [m]} ||\mathbf{x}_j - \widehat{\mathbf{x}}_j||_2^2 \\ \text{subject to:} \quad & P_{\mathbf{c}}(\widehat{\mathbf{x}}_j) = 0 \end{aligned} \tag{9}$$

We use the heuristic of Ma et al. (2008) for solving (9) through a first order approximation of $P_{\mathbf{c}}$ and cluster gradients using the "voting" method also by Ma et al.



(a) CAMRa2011 Similarity (b) Netflix Similarity (c) CAMRa2011 and Netflix (inset) RMSE

Figure 2: Similarity and RMSE performance of K-Means, Spectral, EM, and GPCA, for households of size 2 in the CAMRa2011 and Netflix datasets.



(a) High similarity (b) Intermediate similarity (c) Low similarity

Figure 3: PDF of the difference in distance of $\mathbf{x}_j$ from the two hyperplanes, computed by EM for three different households of size 2, ordered in decreasing similarity.

4.2 Evaluation

We evaluate the four algorithms, namely K-Means, Spectral, EM, and GPCA over CAMRa2011 and Netflix. In the CAMRa2011, we focus on the 272 composite accounts obtained by merging the ratings of users belonging to households of size 2. In Netflix, we focus on the 300 composite accounts obtained by pairing 600 users. For each composite account H, the original mapping $I^* : \mathcal{M} \to \{1, 2\}$ serves as ground truth.

Similarity and RMSE. We measure the performance of each algorithm two ways. First, we compare the mapping $I : \mathcal{M} \to \{1, 2\}$ obtained to the ground truth through the following *similarity* metric:

$$s(I, I^*) = \max_{\pi \in \Pi(\{1,2\})} \frac{1}{m} \sum_{j \in \mathcal{M}} \mathbb{1}\{\pi(I(j)) = I^*(j)\}$$

where $\Pi(\{1, 2\})$ is the set of permutations of $\{1, 2\}$. In other words, the similarity between I and I^* is the fraction of movies in $\mathcal{M}$ which I and I^* agree, up to a permutation. Notice that, by definition $s(I, I^*) \geq 0.5$. Second, we compute how well the obtained profiles $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i \in H}$ fit the observed data by evaluating the root mean square error: $\text{RMSE}(\boldsymbol{\Theta}, I) = \sqrt{\text{MSE}(\boldsymbol{\Theta}, I)}$, where MSE is given by (3).

Figures 2a and 2b show the cumulative distribution function (CDF) of the similarity metric s across all CAMRa2011 and Netflix composite accounts, respectively. Spectral performs the best in terms of similarity with EM (in CAMRa2011) and K-Means (in Netflix) being close seconds. The fact that clustering methods perform so well, in spite of treating ratings as "yet another" feature, suggests that users in these composite accounts indeed tend to watch different types of movies. Nevertheless, though K-Means and Spectral are comparable to EM in terms of s, they exhibit roughly double the RMSE of EM, as seen in Figure 2c. This is because, by grouping together similar movies with dissimilar ratings, these methods partition $\mathcal{M}$ in sets in which the linear regression (5) performs poorly.

Statistical Significance. In order to critically assess our results, we investigated the statistical significance of user identification performance under EM. We generated a null model by converting each of the 544 (600) users in the CAMRa2011 (Netflix) dataset into a composite account, by splitting the movies they rated into two random sets, thereby creating two fictitious users. Our random selection was such that the size ratio between the two sets in this partition followed the same distribution as the corresponding ratios in the real composite accounts. Our construction thus corresponds to a random "ground truth" that exhibits similar statistical properties as the original dataset. We subsequently ran EM over these 544 (600) fictitious composite accounts, and computed the similarity w.r.t. the random ground truth.

The resulting similarity metric CDF is indicated on Figures 2a and 2b as "Random EM". In CAMRa2011 (resp. Netflix), this curve indicates that any similarity $s > 0.59$ in CAMRa2011 (resp. $s > 0.52$) yields a p-value (probability of the similarity being larger or equal to s under the null hypothesis) below 0.05. This corresponds to 41% and 88% of the composite accounts, respectively in each dataset. For these households, we can be confident that the high similarity performance is not due to random fluctuations.

Precision at the Tail. The similarity metric captures the performance of user identification in the aggregate across all movies in $\mathcal{M}$. Nevertheless, even when the similarity metric is extremely low, we can still attribute some movies to distinct users with very high confidence. As we will see in Section 5.3, this is important, because identifying even a few movies that a user has watched can be quite informative.

Let $(\mathbf{u}_i, z_i)$, $i \in \{1, 2\}$, be the profiles computed by EM for a given household H. Figure 3 shows histograms of the difference

$$\Delta_j = |r_j - \langle u_1, v_j \rangle - z_2| - |r_j - \langle u_2, v_j \rangle - z_2|, \quad (10)$$

for $j \in \mathcal{M}$, for three different composite accounts in CAMRa2011. Note that EM classifies j as a movie rated by user 1 when $\Delta_j < 0$, and as a movie rated by user 2 otherwise. The three figures show the histograms of Δ_j for three households with high (0.90), intermediate (0.68), and low (0.50) similarity, respectively. The blue and green colors of each bar indicate the number of movies truly rated by user 1 and user 2, respectively. The total height of each bar corresponds to the total number of movies with that value of Δ_j.

The household in Figure 3a exhibits a clear separation between the two users; indeed, Δ_j is negative for most movies rated by user 1 and positive otherwise. In contrast, in Figures 3b and 3c the distribution of

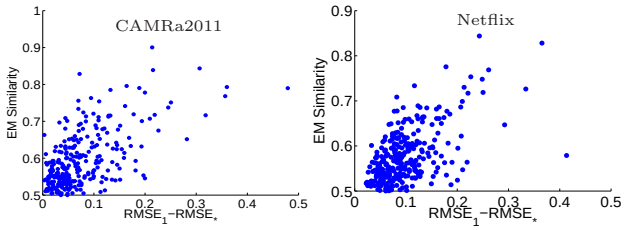


Figure 4: EM Similarity vs. gap in RMSE.

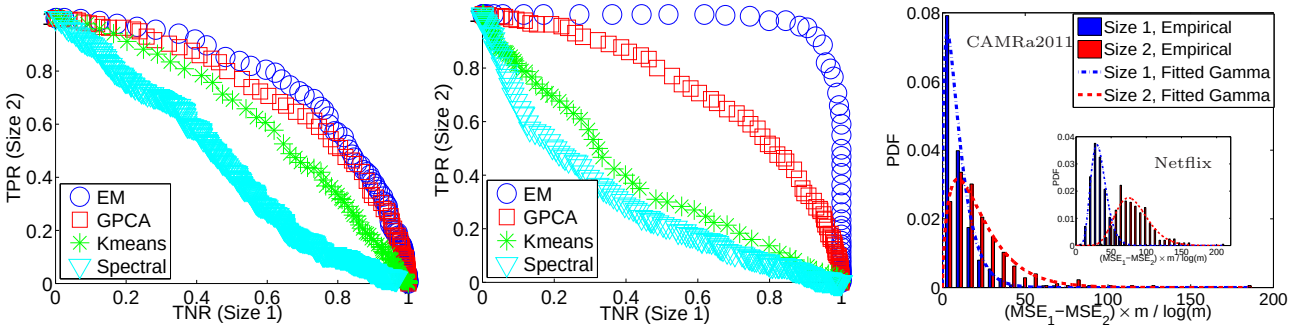
Δ_j is concentrated at zero. Intuitively, a large number of movies are difficult to classify between users 1 and 2. Nevertheless, the tails of this distribution are overwhelmingly biased towards one of the two users. In other words, when labeling movies that lie on the tails of these distributions, our confidence is very high. We determine formally the tails of these curves by fitting a Gaussian on these histograms, after discarding points whose distance from the mean exceeds 1.5 standard deviations. Indeed, mapping the tails above these curves to distinct users identifies them accurately.

Similarity Correlation to Diversity. For each composite account, we computed the RMSE assuming that *all ratings were generated by a single user*: *i.e.*, we obtained a single profile $\boldsymbol{\theta}_1$ solving the regression (5), assuming that $I(j) = 1$ for all $j \in \mathcal{I}$, and used this to obtain an RMSE, denoted by RMSE_1. We also computed the RMSE assuming that *the mapping of ratings to users is known*: *i.e.*, we obtained two profiles $\boldsymbol{\theta}_1^*$ and $\boldsymbol{\theta}_2^*$ by solving the regression (5), assuming that $I = I^*$, and used these profiles to obtain a new RMSE, denoted by RMSE_*.

Figure 4 shows the similarity metric $s(I, I^*)$ for each composite account, computed using EM, versus the gap $\text{RMSE}_1 - \text{RMSE}_*$ for a particular household. We observe a clear correlation between the two values for both datasets. Intuitively, the EM method fails to identify users precisely on households where users *have similar profiles*, and for which distinguishing the users has little impact on the RMSE. The method performs well when users are quite distinct, and a single profile does not fit the observed data well.

5 Model Selection

The user identification methods presented in the previous section assume a priori knowledge of the number of users sharing a composite account. However, this information may not be readily available; in fact, determining if an account is composite or not is an interesting problem in itself. In this section, we propose and evaluate algorithms for this task.



(a) CAMRa2011 (b) Netflix (c) Normalized MSE Gap Distribution for CAMRa2011 and Netflix (inset).

Figure 5: (a)-(b): ROC curves, where TPR (TNR) is the number households correctly labeled as size two (one) over the number of households labeled as size two (one). (c): Normalized MSE gaps for sizes 1 and 2

5.1 Model Selection

The problem of estimating the number of unknown parameters in a model is known as *model selection*—see, *e.g.*, Hansen and Yu (2001). Denoting by $\mathbf{\Theta}_n \in \mathbb{R}^{n\times(d+1)}, I_n \in \mathcal{I}$ the estimators of the parameters $\mathbf{\Theta}^*, I^*$ of the linear model (1) for size n, the general method for model selection amounts to determining n that minimizes $-\frac{1}{m}L(\mathbf{\Theta}_n, I_n) + \frac{C(\mathbf{\Theta}_n, I_n)}{m}$ where $L(\mathbf{\Theta}_n, I)$ is the log-likelihood of the data, given by (2), and C is a metric capturing the *model complexity*, usually as a function of the number of parameters n. Several different approaches for defining C exist; we report our results only for the *Bayesian Information Criterion* (BIC), by Schwarz (1978), as we observed that it performs best over our datasets.

The BIC for a household H of size $|H| = n$ is given by

$$BIC_n := \frac{1}{2\sigma^2}\text{MSE}(\mathbf{\Theta}_n, I_n) + \frac{2n(d+1)\log m}{m}. \quad (11)$$

where σ^2 is the variance of the Gaussian noise in (1). Note that different methods for obtaining the estimators $\mathbf{\Theta}_n, I_n$ lead to different values for BIC_n.

We tested BIC on our two datasets as follows. For the CAMRa2011 (Netflix) dataset, we created a combined dataset comprising the 272 (300) composite accounts of $n = 2$ as well as as the 544 (600) individuals of size $n = 1$ that are included in these households, yielding a total of 816 (900) accounts. For each of these accounts, we first computed the MSE under the assumption that $n = 1$; this amounted to solving the regression 5 for a single profile $\boldsymbol{\theta}_1 = [\mathbf{u}_1, z_1]$ under $I(j) = 1$, for all $j \in \mathcal{M}$, obtaining an MSE we denote by MSE_1. Subsequently, we used each of the four identification methods (EM, GPCA, K-Means, and Spectral) to obtain a mapping $I : \mathcal{M} \to H$, and vectors $\boldsymbol{\theta}_i = (\mathbf{u}_i, z_i)$, $i \in \{1, 2\}$: each of these yielded an MSE for $n = 2$, denoted by MSE_2.

Using these values, we constructed the following classifier: we labeled an account as composite when

$$(MSE_1 - MSE_2) - \tau \log m/m > 0 \quad (12)$$

By varying τ, we can make the classifier more or less conservative towards declaring accounts as composite. For $\tau = 2\sigma^2(d+2)$, this classifier coincides with BIC.

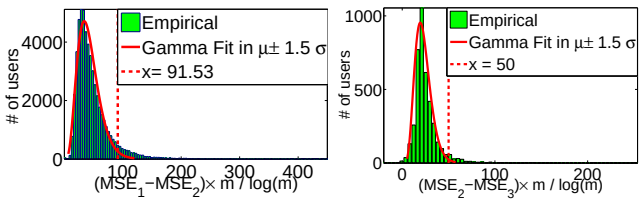
5.2 ROC Curves

The ROC curves obtained under different estimator functions for the model parameters can be found in Figure 5a for CAMRa2011. There is a clear ordering of the performance of different estimators as follows: EM (AUC=0.7711), GPCA (0.7455), K-Means (0.6111) and Spectral (0.4458). In particular, EM and GPCA yield very good classifiers. The performance on Netflix (Figure 5b) is even more striking, where EM (AUC=0.9796) significantly outperforms GPCA (0.7287), while K-Means (0.3879) and Spectral (0.2934) perform very poorly.

In Figure 5c, we plot the distribution of the normalized gap $(\text{MSE}_1 - \text{MSE}_2) \times m/\log m$ under EM for accounts of size 1 and 2, respectively. We see that in both datasets the distributions are well approximated by gamma distributions. Most importantly, accounts of size 2 exhibit a heavier tail. This is why labeling the outliers in the normalized gap distribution as households of size 2 as in (12) performs well.

5.3 Finding Composite Accounts on Netflix

Model Selection in Netflix. Armed with the above classification method, we turn our attention to the 54 390 users of the Netflix dataset that rated more than 500 movies. A natural question to ask is how many users in this dataset are in fact composite.



(a) Gap 1–2 in Netflix (b) Gap 2–3 in Netflix

Figure 6: PDF of RMSE gap in 54K Netflix users that rated more than 500 movies.

We first applied BIC with EM as a user identification method on these users. That is, we applied EM under the assumption the household size is n =1,2, and 3, and labeled a household with the value n that minimized BIC_n. We estimated the noise variance σ^2 through the mean square error of the matrix factorization applied on the entire dataset. The resulting classification labeled 36 832, 14 789, and 2 769 accounts as of size 1, 2, and ≥ 3, respectively.

We also applied an alternative method (akin to the empirical Bayes approach of Efron (2009)). First, we plotted the histogram of the normalized gap in the MSE from a model of 1 to 2 users (Figure 6a). We then identified the outliers of this curve, and labeled them as accounts of size 2 and above. To identify the outliers, we fitted a gamma distribution to the portion of the histogram that lies within 1.5 standard deviations from the mean. Superimposing the two distributions, we found the normalized gap value (91.53) at which the tail of the original distribution (which has a heavier tail) had twice the value of the fitted gamma distribution; all accounts with a higher normalized gap were labeled as outliers. To identify accounts with size 3 and above, we repeated the above process only on the outliers, using now the normalized gap between models of 2 and 3 users (Figure 6b).

The resulting classification is compared to the classification under BIC in the following table:

BIC\Outl.	1	2	≥ 3	Tot.
1	36832	0	0	36832
2	12712	2071	6	14789
≥ 3	774	1805	190	2769
Tot.	50318	3876	196	54390

Note that the above method of outliers is more conservative when labeling accounts as composite, labeling only 4 072 users as composite. Nevertheless, we know that this method performs well over the datasets on which we have ground truth (c.f. Figure 5c).

Visual Inspection. Though we cannot assess the accuracy of this classification (we lack ground truth), a visual inspection of the accounts that were labeled as composite yield some interesting observations. Recall that, in each composite account, there are a few movies that we can assign to different users with very high confidence: these are precisely the movies that lie close to one of the two hyperplanes computed by EM and far from the other (*c.f.* Figure 3).

User 1	User 2
TLOTR: The Fellowship of the Ring†(5), TLOTR: The Return of the King†(5), TLOTR: The Two Towers†(5), The Whole Nine Yards(4), Immortal†(1), The Deep End(2), Toys†(4), The Addams Family(5)	H.R. Pufnstuf(5), Sex and the City: Season 5$^{\heartsuit}$(1), Me Myself & Irene(1), All the Real Girls$^{\heartsuit\triangle}$(5), Titanic$^{\heartsuit}$(5), George Washington$^{\triangle}$(5), The Siege(1), In the Bedroom$^{\triangle}$(5)
User 1	**User 2**
Monsters Inc.$^{\diamondsuit}$(5), Finding Nemo$^{\diamondsuit}$(5), Whale Rider(5), Con Air(4), Lilo and Stitch$^{\diamondsuit}$(4), Ice Age$^{\diamondsuit}$(5), Ring of Fire(4), Star Trek: Nemesis(3),	In America$^{\spadesuit}$(2), Super Size Me(2), A Very Long Engagement$^{\spadesuit}$(1), Bend It Like Beckham(2), 21 Grams$^{\spadesuit}$(1), Airplane II: The Sequel(4), Spun$^{\spadesuit}$(1), Fahrenheit 9/11(1)

Table 1: Movies rated by accounts labeled as composite in the Netflix dataset. We split each account using EM and show movies j with most positive and most negative Δ_j. Symbols indicate labels from the Netflix website: $\dagger$ = "Sci-Fi & Fantasy", $\heartsuit$ = "Romantic", $\triangle$ = "Understated", $\diamondsuit$ = "Children & Family Movies", $\spadesuit$ = "Drama"

Using this intuition, we ran EM on several accounts declared as composite (size 2) by both BIC and the outlier method, and computed Δ_j, given by (10) for each movie j rated by these accounts. Table 1 shows the titles of the 8 most positive and 8 most negative movies for 2 such accounts. Looking up these titles on the Netflix website indicates clearly that these accounts exhibit a bimodal behavior. In the first account, 5/8 movies rated by User 1 are labelled as "Sci Fi & Fantasy", while 5/8 movies rated by User 2 are labelled either "Romantic" or "Understated". Similarly, in the second household, 4/8 movies rated by User 1 are labelled "Children & Family Movies", while 4/8 movies rated by User 2 are labelled as "Dramas", suggesting movies viewed by a child and an adult, respectively.

In many accounts we inspected, sequels (*e.g.*, "Lord of the Rings", "Star Wars", *etc.*) or seasons of the same TV show (*e.g.* "Sex and the City", "Friends", *etc.*) were grouped together (*i.e.*, attributed to the same user). The first account in Table 1 illustrates this.

We stress that we did not use any labeling or title information in our classification, as neither was available for both datasets. Nevertheless, as noted Section 3, such information can be incorporated in our model by extending v_j to include any additional features.

6 Targeted Recommendations

In this section, we illustrate how knowledge of household composition can be used to improve recommen-

dations. In a typical setup, a user accesses the account and the recommender system suggests a small set of movies from a catalog, recommending movies that are likely to be rated highly. However, even if the recommender knows the household composition and the user profiles, it still does not know who might be accessing the account at a given moment. In the absence of side information, we can circumvent this problem as follows. Assume the recommender has a budget of K movies to be displayed; it can then recommend the union of the K/n movies that are most likely to be rated highly by each of the n users. This exploits household composition, without requiring knowledge of who is presently accessing the account.

To investigate the benefit of user identification, we performed a 5-fold cross validation in each of the 272 households in CAMRa2011, whereby user profiles where trained in 4/5ths of $\mathcal{M}$ (the training set), and used to predict ratings in the remaining 1/5th (the test set). As, in the real-life setting, we can circumvent identifying which user is accessing an account when recommending movies, we focus on predicting the ratings of users accurately. Ideally, we would like to assess our rating prediction over the test set for both users; unfortunately, we have the true rating of only one user for each movie. As a result, we assume that the mapping of movies to users is priori known on the test set (but *not* on the training set): to generate a prediction for a movie in the test set, we generate a single rating using the profile of the user that truly generated it.

We tested the following 4 methods. The first, termed *Single*, ignores the household composition; a unique profile $\boldsymbol{\theta}_S = (\mathbf{u}_i, z_i)$ is computed over the training set for both users through ridge regression over (5) using $I(j) = 1$ for all j in the training set. The regularization parameter is chosen through cross validation. The second method, termed *Oracle*, assumes that the mapping of movies to users is known in the trainset; profiles $\theta_i^* = (\mathbf{u}_i^*, z_i^*)$, $i \in \{1, 2\}$ are obtained on the trainset using again ridge regression over (5) using $I = I^*$.

The third method, termed EM, uses the EM method outlined in Section 4 to obtain user profiles $\boldsymbol{\theta}_i = (\mathbf{u}_i, z_i)$. The EM algorithm is modified by adding regularization factor to the MSE, and using ridge rather than linear regression in each step. Finally, the last method, termed CNV for "convex", uses as a profile a linear combination of the common profile computed by Single and the specialized profile computed by EM. *I.e.* the profile of user i is given by $\alpha\boldsymbol{\theta}_S + (1-\alpha)\boldsymbol{\theta}_i$, with α computed through cross validation.

We evaluate the performance of these methods in terms of two metrics. The first is the RMSE of the predicted ratings on the test set. The second, which we call *overlap*, is computed by generating a list of 6 movies and calculating the number of common elements with the 3 top rated movies by each user in the test set. For Single, the list is generated by picking the 6 movies in the test set with the highest predicted rating. For the remaining methods, we generate the list by picking the 3 movies in the test set with the highest predicted rating for each user, and combining these two lists.

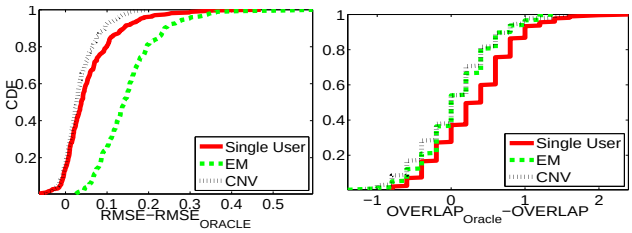


Figure 7: RMSE and OVERLAP performance compared to an oracle for the 272 composite users of CAMRa2011, when using (a) a single profile, (b) EM, and (c) a convex combination of the two.

Figure 7 shows the CDFs of the perfromance of the three mechanisms w.r.t. the distance of each metric from the corresponding metric under Oracle. We first observe that Oracle outperforms all other methods for the majority of the households, having an RMSE 0.60 and overlap 1.87, on average. This indicates that fitting a single profile to a composite account leads to poor predictions, which improve when the household composition is known. EM clearly outperforms Single w.r.t. the overlap metric, having a 14% higher overlap on average; however, it does worse w.r.t. RMSE also by roughly 14%. This is because, as observed in Figure 3, the bulk of movies are rated similarly by users, which dominates behavior in the RMSE; EM performs better on metrics that depend on the performance of outliers, such as overlap. In both metrics, CNV yields an improvement on both EM and Single, showing that the relative benefits of both methods can be combined.

7 Conclusion

We proposed methods for user identification solely on the ratings provided by users based on subspace clustering. Evaluating such methods in the presence of additional information is a potential future direction of this work. We also believe modeling rating data as a subspace arrangement can provide insight on a variety of applications, including privacy in recommender systems. In particular, altering or augmenting one's rating profile to *appear* as a composite user, with the purpose of obscuring, *e.g.*, one's gender, is an interesting research topic.

References

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1).

Efron, B. (2009). Empirical Bayes estimates for large-scale prediction problems. *Journal of the American Statistical Association*, 104(487):1015–1028.

Elhamifar, E. and Vidal, R. (2009). Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797. IEEE.

Eriksson, B., Balzano, L., and Nowak, R. (2011). High-rank matrix completion and subspace clustering with missing data. arXiv:1112.5629.

Hansen, M. and Yu, B. (2001). Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774.

Keshavan, R. H., Montanari, A., and Oh, S. (2010). Matrix completion from a few entries. *IEEE Trans. Inform. Theory*, 56(6):2980–2998.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Liu, G., Lin, Z., and Yu, Y. (2010). Robust subspace segmentation by low-rank representation. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*.

Ma, Y., Yang, A., Derksen, H., and Fossum, R. (2008). Estimation of subspace arrangements with applications in modeling and segmenting mixed data. *SIAM review*, 50(3):413–458.

Said, A., Berkovsky, S., Luca, E. W. D., and Hermanns, J., editors (2011). *CAMRa '11: Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation*, New York, NY, USA. ACM.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Soltanolkotabi, M. and Candes, E. (2011). A geometric analysis of subspace clustering with outliers. arXiv:1112.4258.

Srebro, N. and Jaakkola, T. (2003). Weighted low-rank approximations. In *20th International Conference on Machine Learning*, pages 720–727. AAAI Press.

Srebro, N., Rennie, J. D. M., and Jaakola, T. S. (2005). Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press.

Vidal, R. (2010). A tutorial on subspace clustering. *IEEE Signal Processing Magazine*.

Vidal, R., Ma, Y., and Sastry, S. (2005). Generalized principal component analysis (gpca). *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(12):1945–1959.

AUAI Press
P.O. Box 866
Corvallis, Oregon 97339
USA