Optimal Algorithms for Learning Bayesian Network Structures: **Introduction and Heuristic Search**

Changhe Yuan

UAI 2015 Tutorial
Sunday, July 12th, 8:30-10:20am
http://auai.org/uai2015/tutorialsDetails.shtml#tutorial_1

About tutorial presenters

- **Dr. Changhe Yuan (Part I)**
  - Associate Professor of Computer Science at Queens College/City University of New York
  - Director of the Uncertainty Reasoning Laboratory (URL Lab).

- **Dr. James Cussens (Part II)**
  - Senior Lecturer in the Dept of Computer Science at the University of York, UK

- **Dr. Brandon Malone (Part I and II)**
  - Postdoctoral researcher at the Max Planck Institute for Biology of Ageing

Bayesian networks

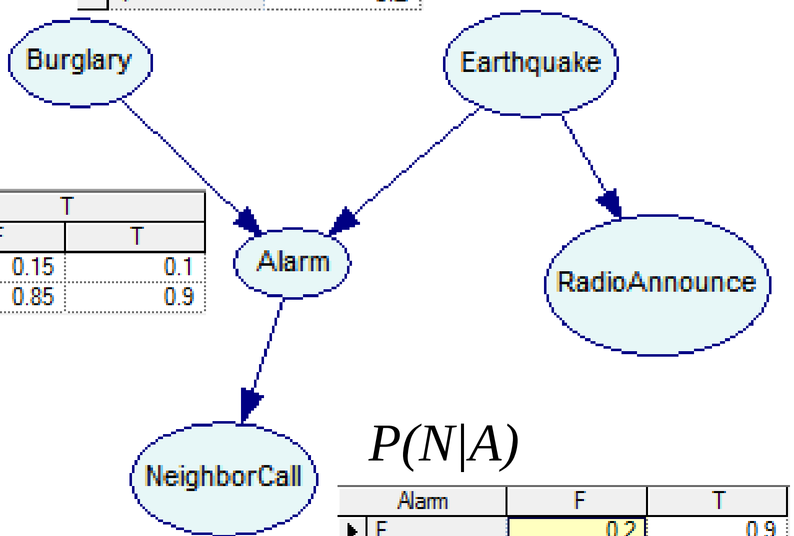
- A **Bayesian Network** is a directed acyclic graph (DAG) in which:
  - A set of random variables makes up the nodes in the network.
  - A set of directed links or arrows connects pairs of nodes.
  - Each node has a conditional probability table that *quantifies* the effects the parents have on the node.

$P(B)$

F	0.8
T	0.2

$P(E)$

F	0.9
T	0.1



$P(A|B,E)$

Burglary	F		T	
Earthquake	F	T	F	T
F	0.8	0.2	0.15	0.1
T	0.2	0.8	0.85	0.9

$P(R|E)$

Earthquake	F	T
F	0.95	0.99
T	0.05	0.01

$P(N|A)$

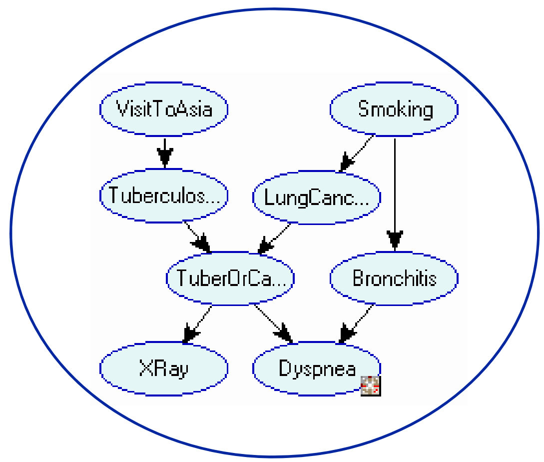
Alarm	F	T
F	0.2	0.9
T	0.8	0.1

Learning Bayesian networks

- Very often we have data sets
- We can learn Bayesian networks from these data

100	100	100	90	390	97.5%
100	95	100	80	375	93.8%
100	100	100	90	390	97.5%
80	95	100	90	365	91.3%
100	100	100	100	400	100.0%
100	100	100	100	400	100.0%
90	95	100	90	375	93.8%
90	95	100	90	375	93.8%
100	100	100	90	390	97.5%
100	100	100	100	400	100.0%
100	90	100	90	380	95.0%
95	90	100	80	365	91.3%
100	95	100	80	375	93.8%
100	95	100	80	375	93.8%
100	100	100	100	400	100.0%

data



structure

Success	0.2
Failure	0.8

Success	Success	Failure
Good	0.4	0.1
Moderate	0.4	0.3
Poor	0.2	0.6

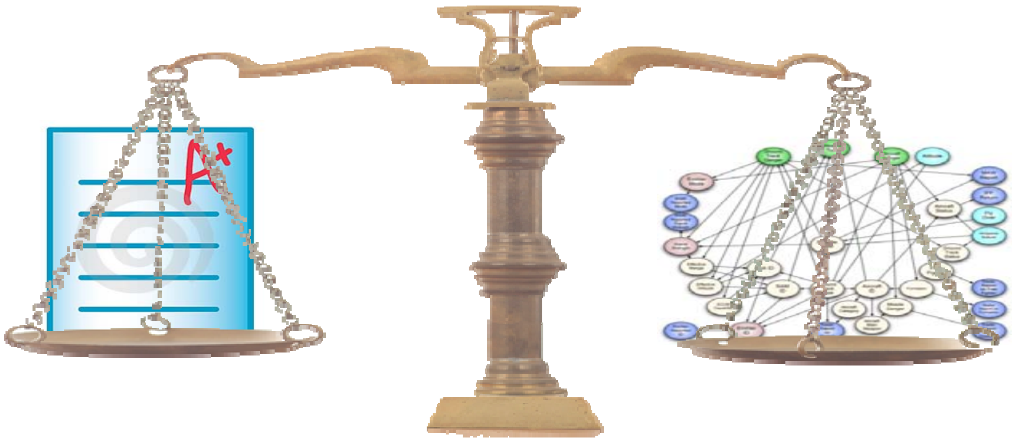
numerical parameters

Major learning approaches

- **Score-based structure learning**
  - **Find the highest-scoring network structure**
    - » **Optimal algorithms (FOCUS of TUTORIAL)**
    - » **Approximation algorithms**
- **Constraint-based structure learning**
  - **Find a network that best explains the dependencies and independencies in the data**
- **Hybrid approaches**
  - **Integrate constraint- and/or score-based structure learning**
- **Bayesian model averaging**
  - **Average the prediction of all possible structures**

Score-based learning

- **Find a Bayesian network that optimizes a given scoring function**



- **Two major issues**
  - **How to define a scoring function?**
  - **How to formulate and solve the optimization problem?**

Scoring functions

- **Bayesian Dirichlet Family (BD)**
  - **K2**
- **Minimum Description Length (MDL)**
- **Factorized Normalized Maximum Likelihood (fNML)**
- **Akaike's Information Criterion (AIC)**
- **Mutual information tests (MIT)**
- **Etc.**

Decomposability

- **All of these are expressed as a sum over the individual variables, e.g.**

BDeu	$\sum_{i}^{n}\sum_{j}^{q_i}\log\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})}+\sum_{k}^{r_i}\log\frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})}$
MDL	$\sum_{i}^{n}-LL(X_i\|PA_i)+\frac{\log N}{2}(r_i-1)q_i$
fNML	$\sum_{i}^{n}\sum_{j}^{q_i}\sum_{k}^{r_i}-N_{ijk}\log\frac{N_{ijk}}{N_{ij}}-C(r_i,N_{ij})$

- **This property is called *decomposability* and will be quite important for structure learning.**

$$Score(G)=\sum_{i}^{n}Score(X_i|PA_i)$$

[Heckerman 1995, etc.]

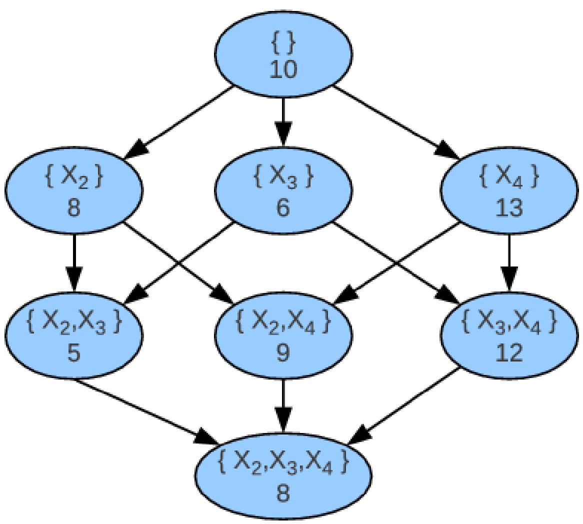
Querying best parents

$$BestScore(X, \mathbf{U}) = \min_{PA_X \subseteq \mathbf{U} \setminus \{X\}} Score(X|PA_X)$$

e.g., $BestScore(X_1, \{X_2, X_4\}) = \min_{PA_{X_1} \subseteq \{X_2, X_4\}} Score(X_1|PA_{X_1})$

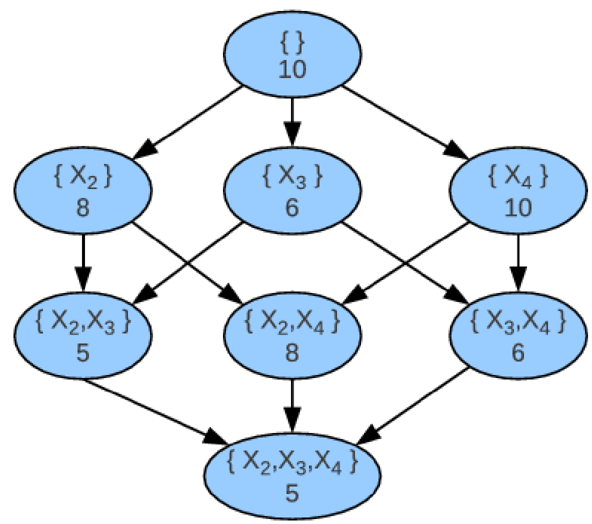
Naive solution: Search through all of the subsets and find the best

$Score(X_1|PA_1)$



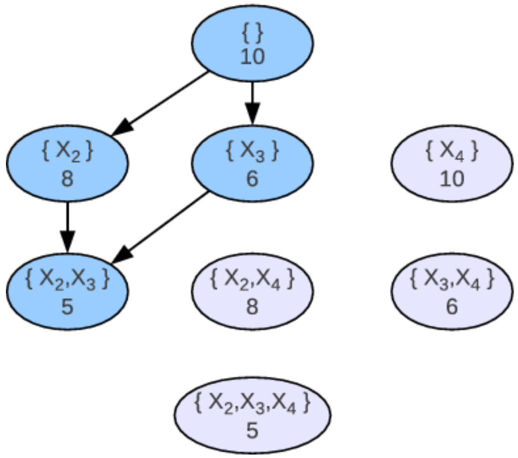
Solution: Propagate optimal scores and store as hash table.

$BestScore(X_1|PA_1)$



Score pruning

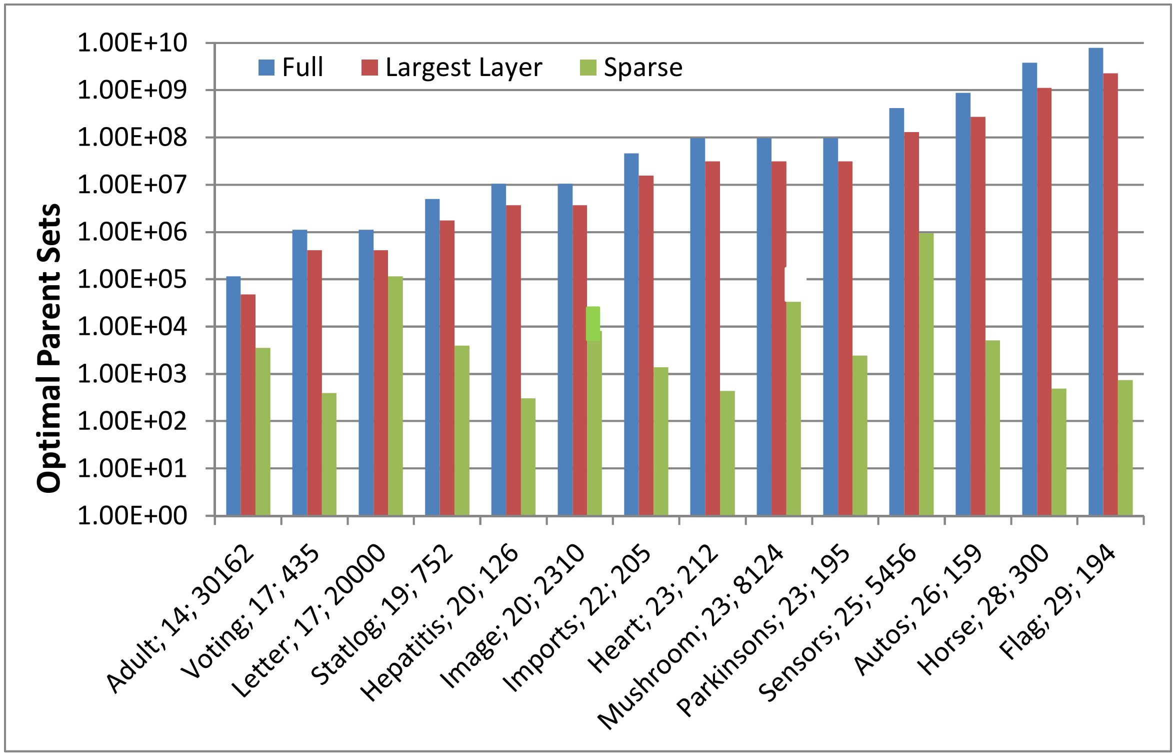
- **Theorem: Say $PA_i \subset PA'_i$ and $Score(X_i|PA_i) < Score(X|PA'_i)$. Then PA'_i is not optimal for X_i.**
- **Ways of pruning:**
  - **Compare $Score(X_i|PA_i)$ and $Score(X|PA'_i)$**
  - **Using properties of scoring functions without computing scores (e.g., exponential pruning)**
- **After pruning, each variable has a list of possibly optimal parent sets (POPS)**
  - **The scores of all POPS are called local scores**



$POPS(X_1|PA(X_1))$

[Teyssier and Koller 2005, de Campos and Ji 2011, Tian 2000]

Number of POPS



The number of parent sets and their scores stored in the full parent graphs ("Full"), the largest layer of the parent graphs in memory-efficient dynamic programming ("Largest Layer"), and the possibly optimal parent sets ("Sparse").

Practicalities

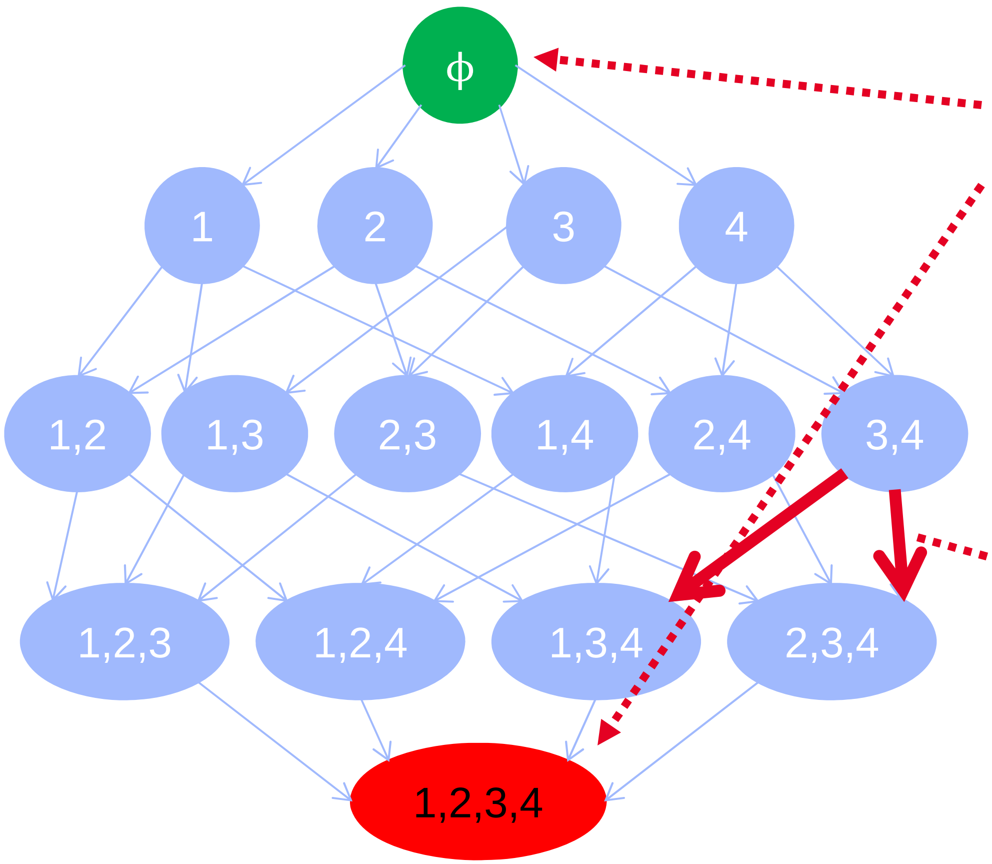
- Empirically, the sparse AD-tree data structure is the best approach for collecting sufficient statistics.
- A breadth-first score calculation strategy maximizes the efficiency of exponential pruning.
- Caching significantly reduces runtime.
- Local score calculations are easily parallelizable.

Graph search formulation

- **Formulate the learning task as a shortest path problem**
  - **The shortest path solution to a graph search problem corresponds to an optimal Bayesian network**

[Yuan, Malone, Wu, IJCAI-11]

Search graph (Order graph)



Formulation:

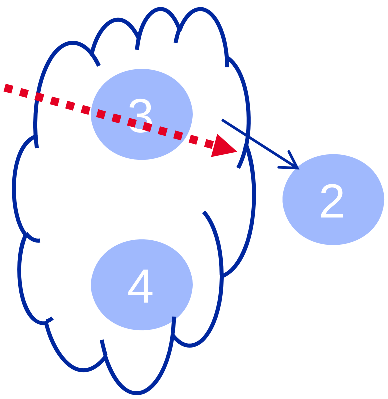
Search space: Variable subsets

Start node: Empty set

Goal node: Complete set

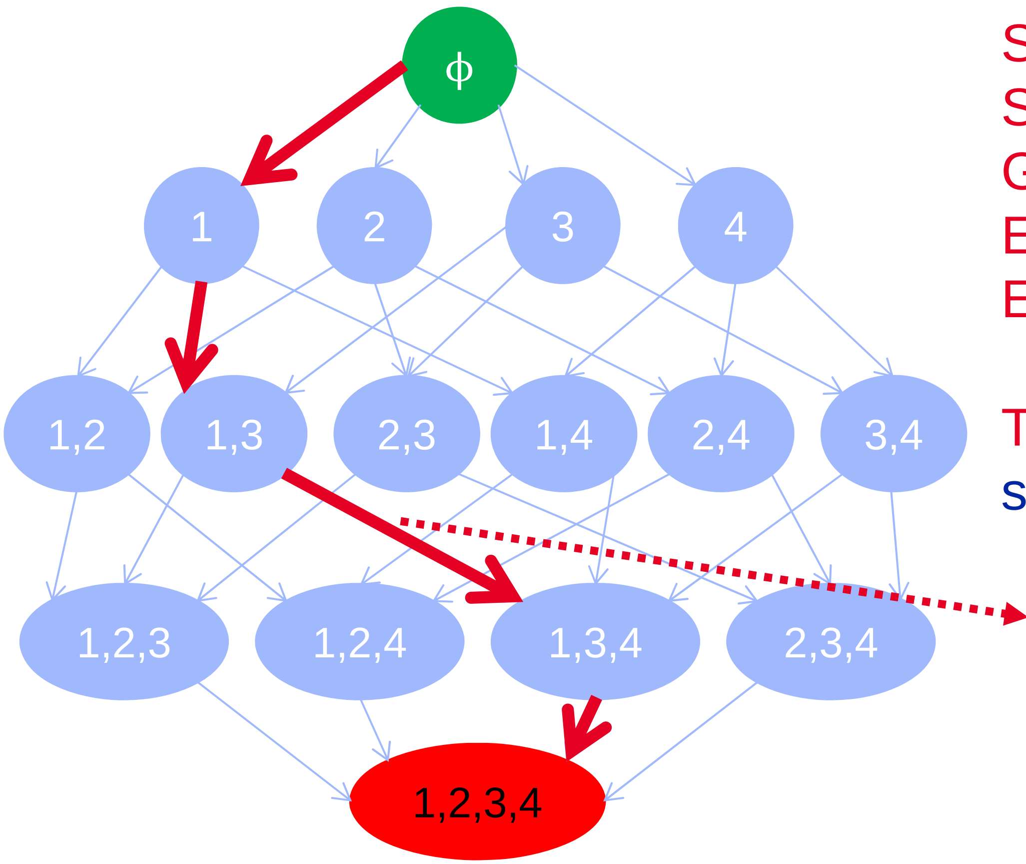
Edges: Add variable

Edge cost: BestScore(X,U) for edge $U \to U \cup \{X\}$



[Yuan, Malone, Wu, IJCAI-11]

Search graph (Order graph)



Formulation:

Search space: Variable subsets

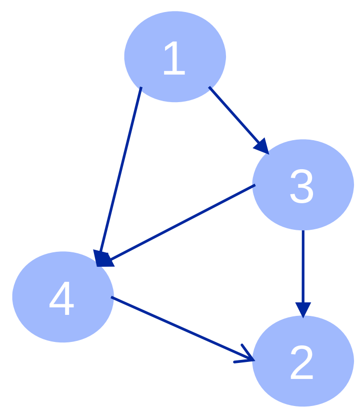
Start node: Empty set

Goal node: Complete set

Edges: Add variable

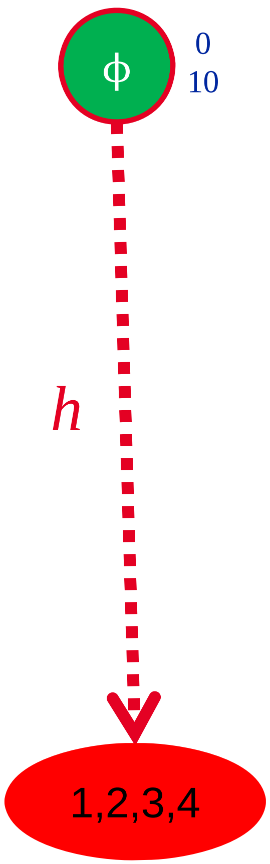
Edge cost: BestScore(X,U) for edge $U \rightarrow U \cup \{X\}$

Task: find the shortest path between start and goal nodes



[Yuan, Malone, Wu, IJCAI-11]

A* algorithm



A* search: Expands the nodes in the order of quality: $f=g+h$

$g(U) = Score(U)$

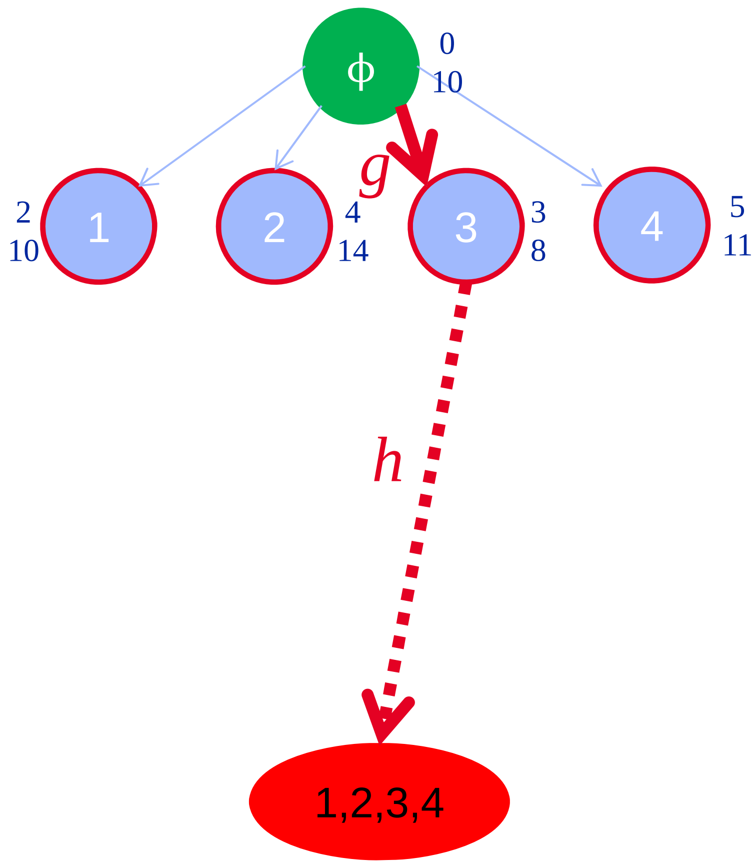
$h(U) = estimated\ distance\ to\ goal$

Notation:

- g: g-cost
- h: h-cost
- Red shape-outlined: open nodes
- No outline: closed nodes

[Yuan, Malone, Wu, IJCAI-11]

A* algorithm



A* search: Expands the nodes in the order of quality: $f=g+h$

$g(U) = Score(U)$

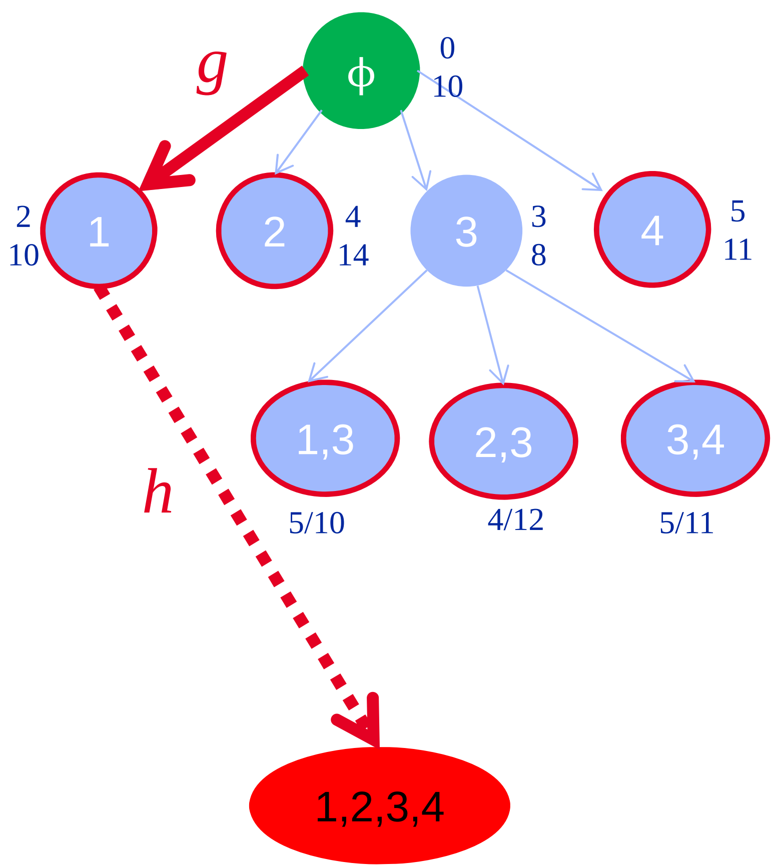
$h(U) =$ *estimated distance to goal*

Notation:

g:	g-cost
h:	h-cost
Red shape-outlined:	open nodes
No outline:	closed nodes

[Yuan, Malone, Wu, IJCAI-11]

A* algorithm



A* search: Expands the nodes in the order of quality: $f=g+h$

$g(U) = Score(U)$

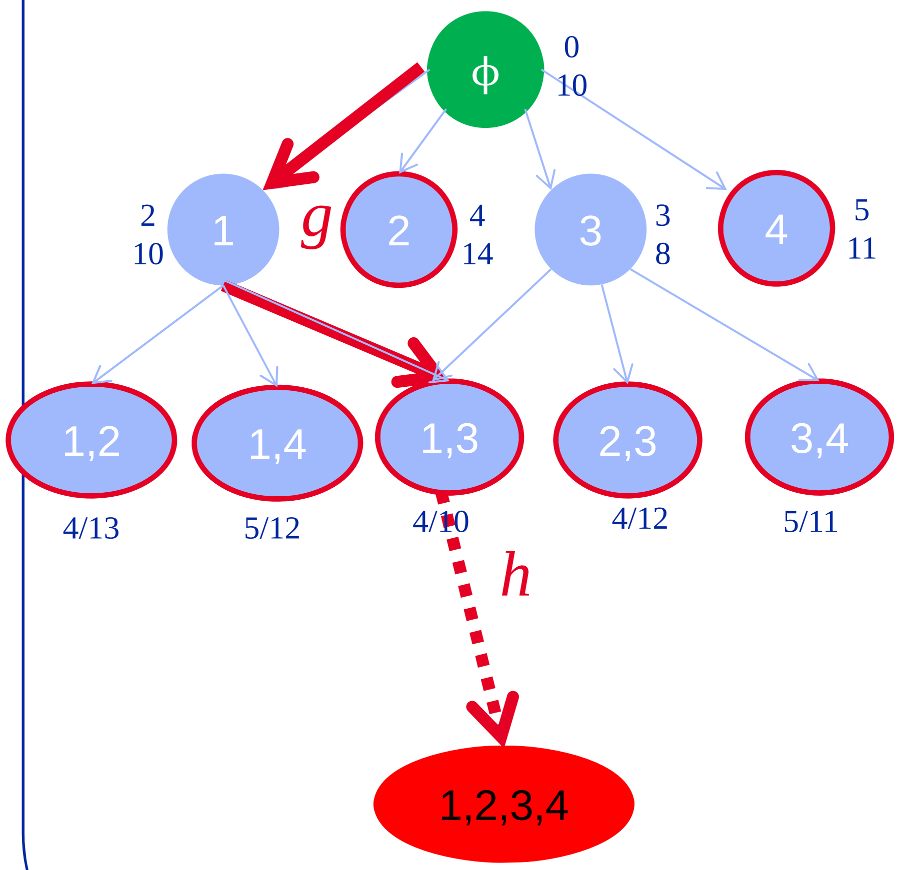
$h(U) = \text{estimated distance to goal}$

Notation:

- *g*: g-cost
- *h*: h-cost
- Red shape-outlined: open nodes
- No outline: closed nodes

[Yuan, Malone, Wu, IJCAI-11]

A* algorithm



A* search: Expands the nodes in the order of quality: $f = g + h$

$g(U) = Score(U)$

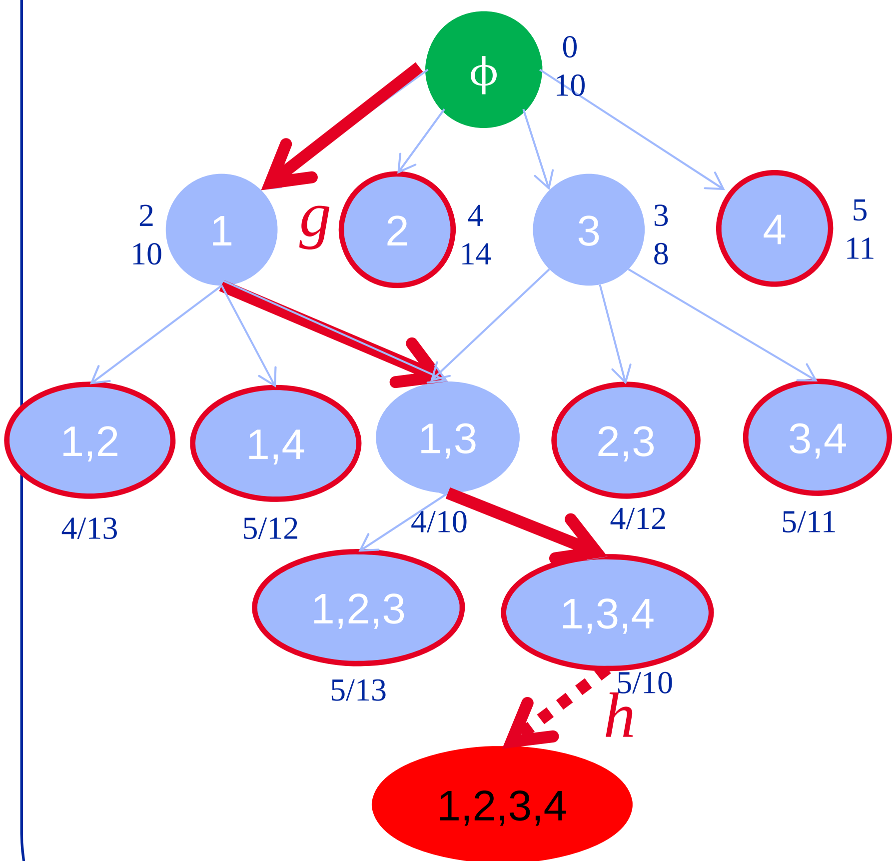
$h(U)$ = *estimated distance to goal*

Notation:

g:	g-cost
h:	h-cost
Red shape-outlined:	open nodes
No outline:	closed nodes

[Yuan, Malone, Wu, IJCAI-11]

A* algorithm



A* search: Expands the nodes in the order of quality: $f = g + h$

$$g(U) = Score(U)$$

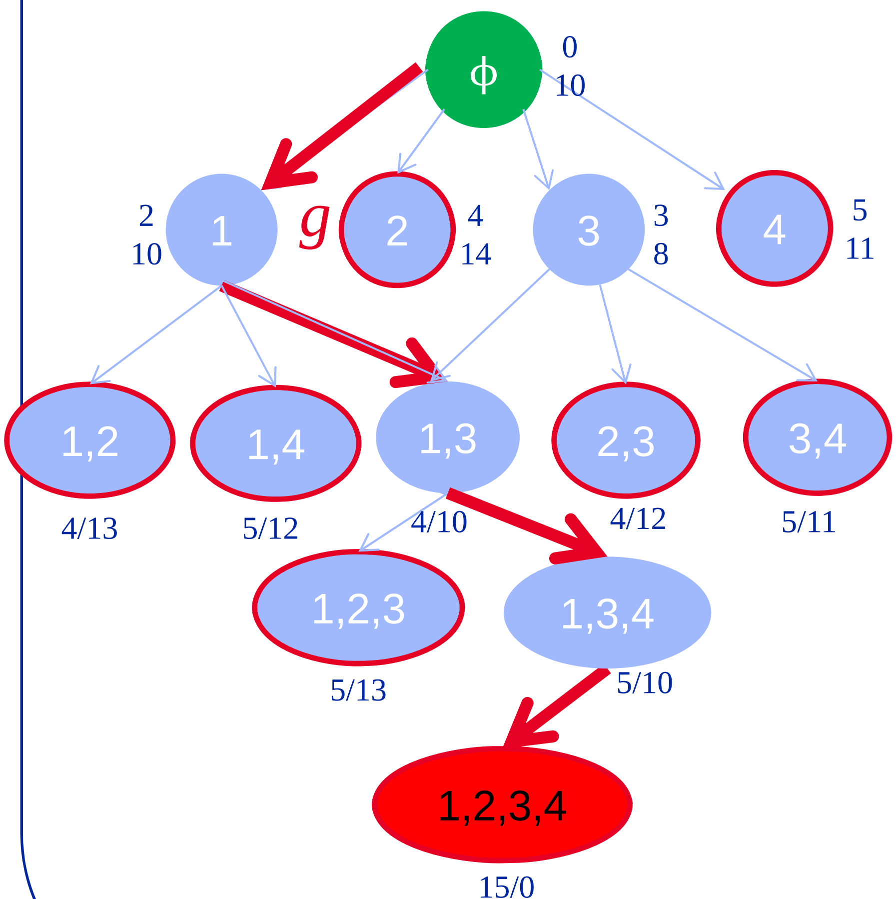
$$h(U) = \text{estimated distance to goal}$$

Notation:

- *g*: g-cost
- *h*: h-cost
- Red shape-outlined: open nodes
- No outline: closed nodes

[Yuan, Malone, Wu, IJCAI-11]

A* algorithm



A* search: Expands the nodes in the order of quality: $f=g+h$

$g(U) = Score(U)$

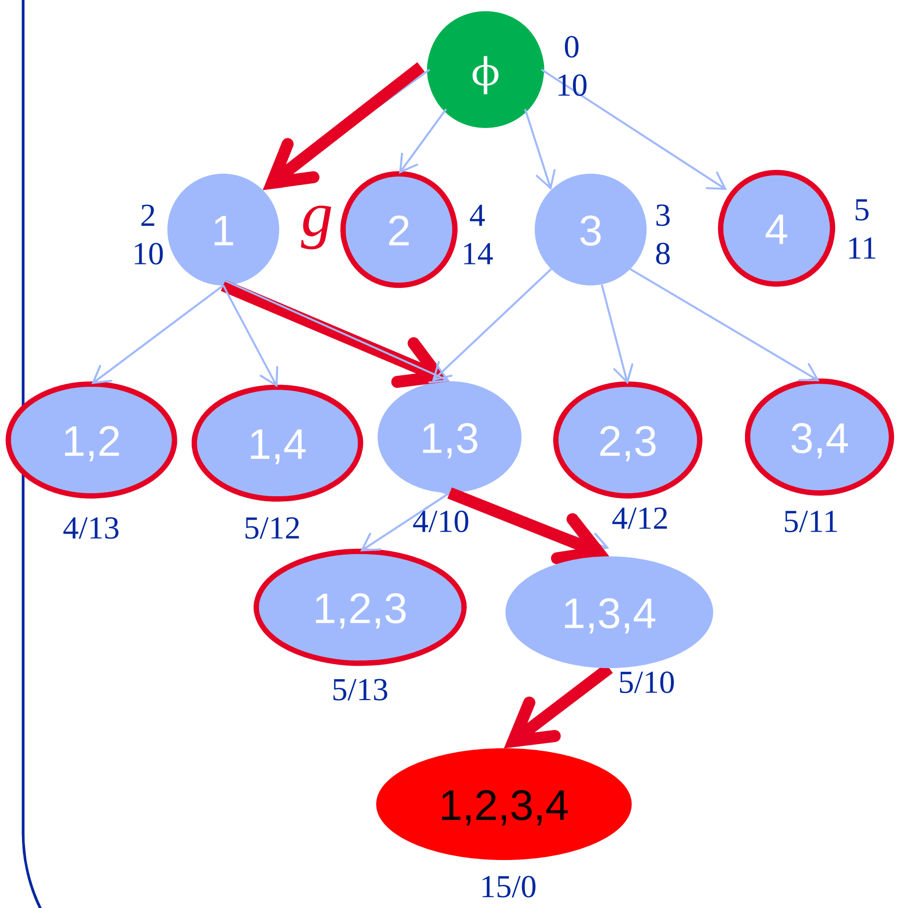
$h(U) = \text{estimated distance to goal}$

Notation:

g:	g-cost
h:	h-cost
Red shape-outlined:	open nodes
No outline:	closed nodes

[Yuan, Malone, Wu, IJCAI-11]

A* algorithm



A* search: Expands the nodes in the order of quality: f=g+h

$g(U) = Score(U)$

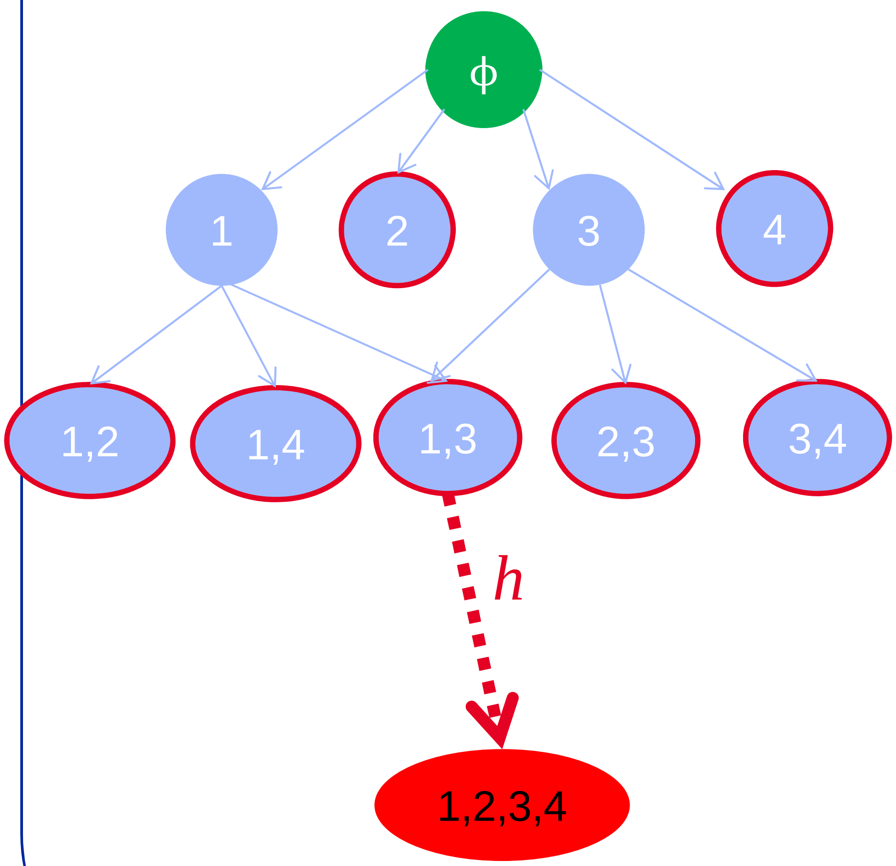
$h(U) = estimated\ distance\ to\ goal$

Notation:

g:	g-cost
h:	h-cost
Red shape-outlined:	open nodes
No outline:	closed nodes

[Yuan, Malone, Wu, IJCAI-11]

Simple heuristic

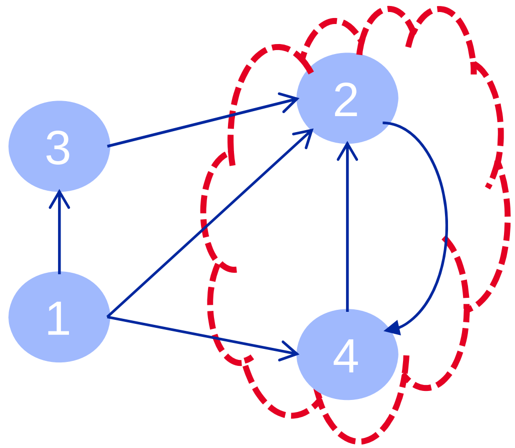


A* search: Expands nodes in order of quality: $f = g + h$

$$g(U) = Score(U)$$

$$h(U) = \sum_{X \in V \setminus U} BestScore(X, V \setminus \{X\})$$

$h(\{1,3\})$:



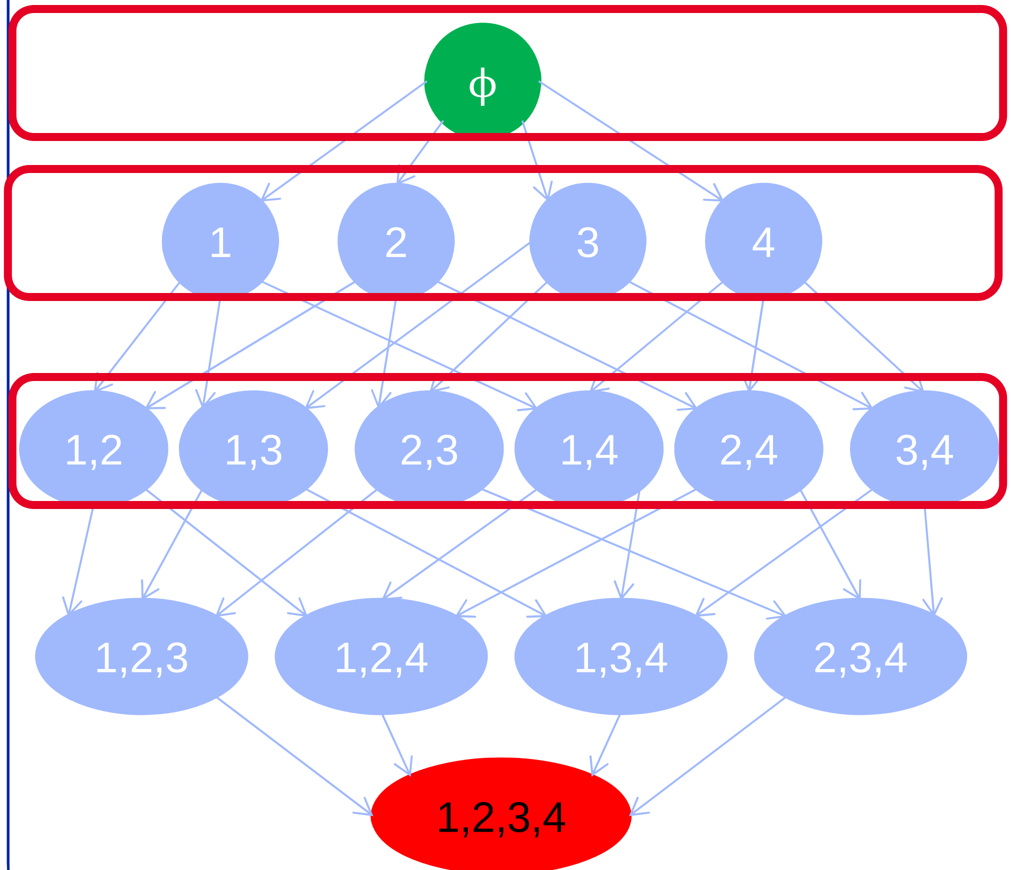
[Yuan, Malone, Wu, IJCAI-11]

Properties of the simple heuristic

- **Theorem: The simple heuristic function *h* is admissible**
  - **Optimistic estimation: never overestimate the true distance**
  - **Guarantees the optimality of A***
- **Theorem: *h* is also consistent**
  - **Satisfies triangular inequality, yielding a monotonic heuristic**
  - **Consistency => admissibility**
  - **Guarantees the optimality of *g* cost of any node to be expanded**

[Yuan, Malone, Wu, IJCAI-11]

BFBnB algorithm

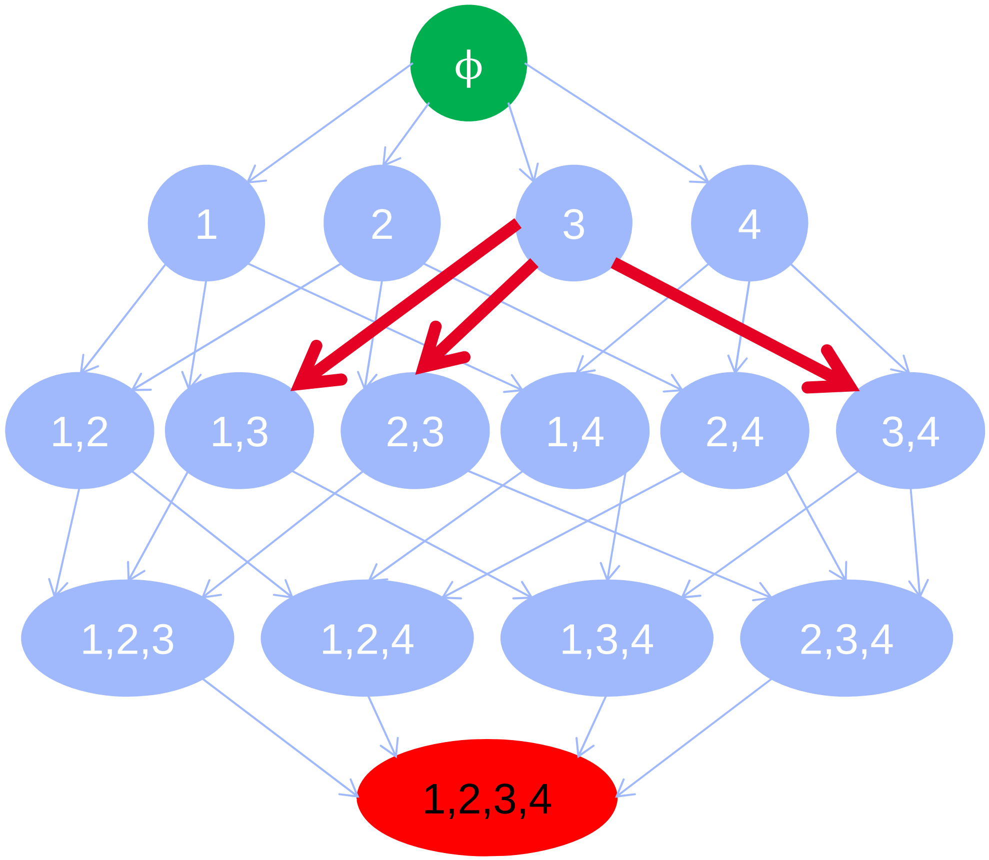


Breadth-first branch and bound search (BFBnB):

- Motivation:
  Exponential-size order&parent graphs
- Observation:
  Natural layered structure
- Solution:
  Search one layer at a time

[Malone, Yuan, Hansen, UAI-11]

BFBnB algorithm

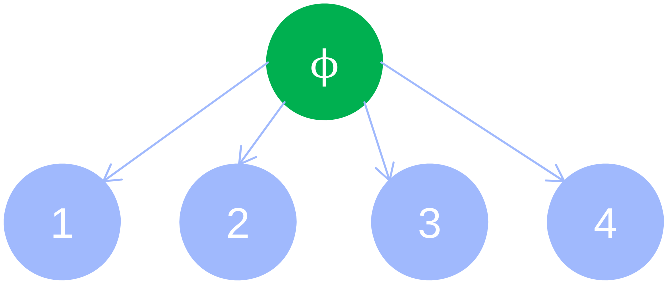


Breadth-first branch and bound search (BFBnB):

- Motivation:
  - Exponential-size order&parent graphs
- Observation:
  - Natural layered structure
- Solution:
  - Search one layer at a time

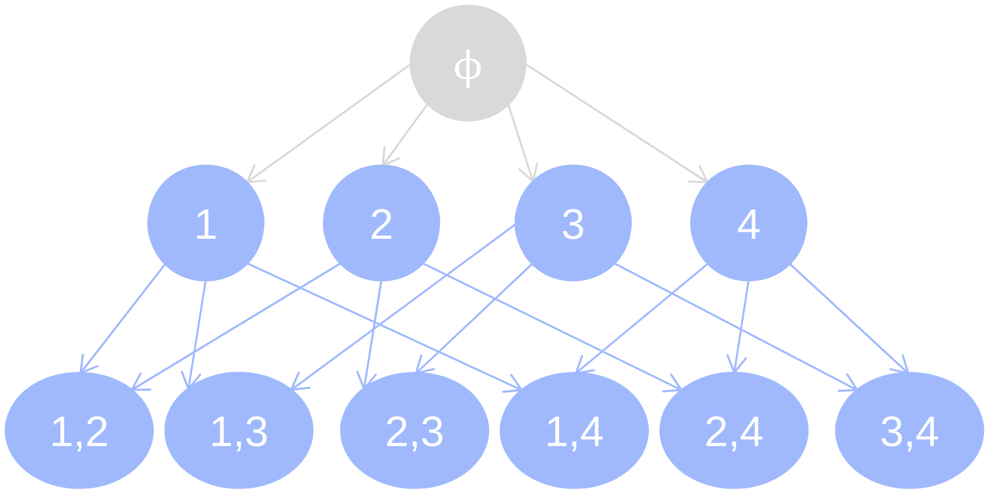
[Malone, Yuan, Hansen, UAI-11]

BFBnB algorithm



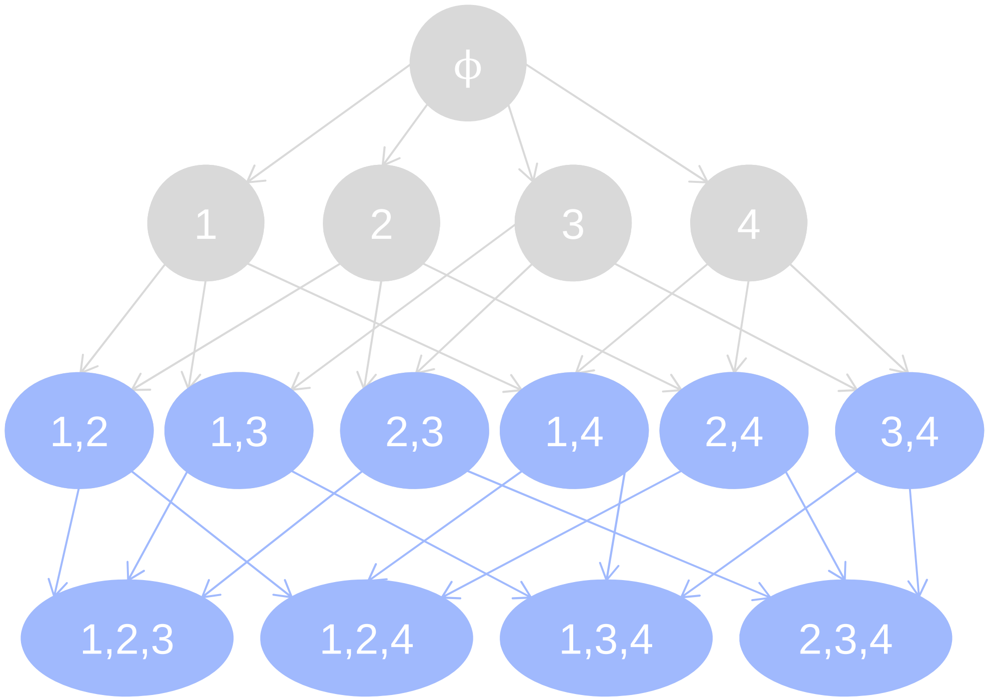
[Malone, Yuan, Hansen, UAI-11]

BFBnB algorithm



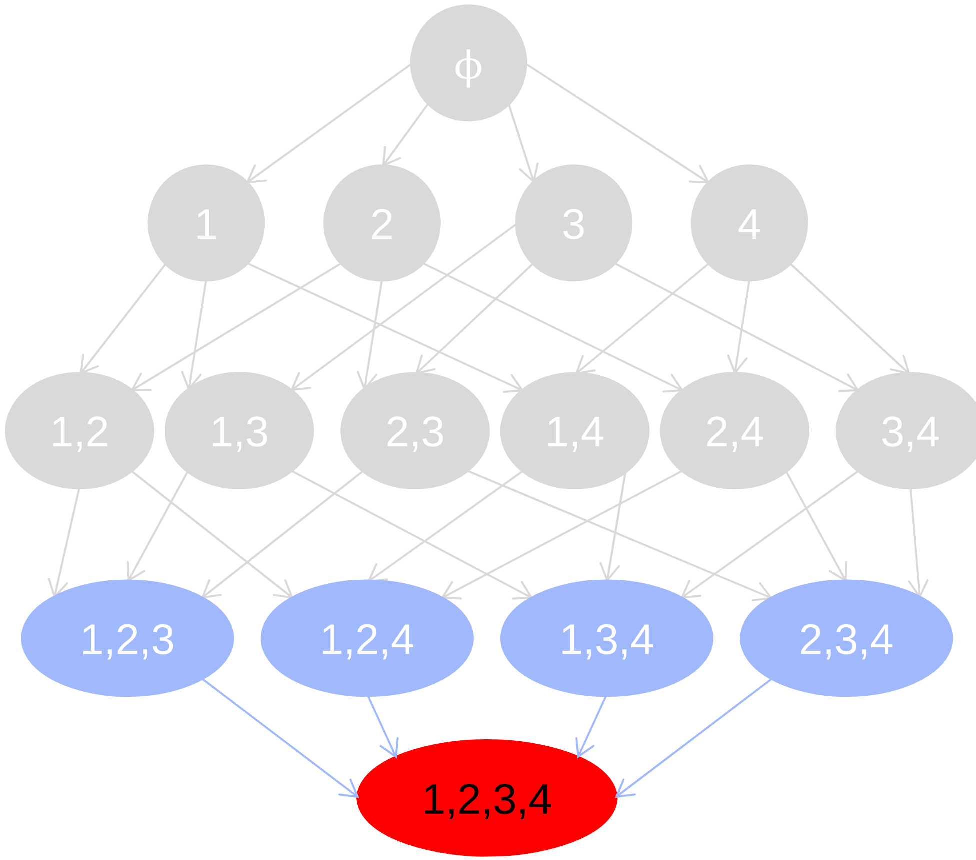
[Malone, Yuan, Hansen, UAI-11]

BFBnB algorithm



[Malone, Yuan, Hansen, UAI-11]

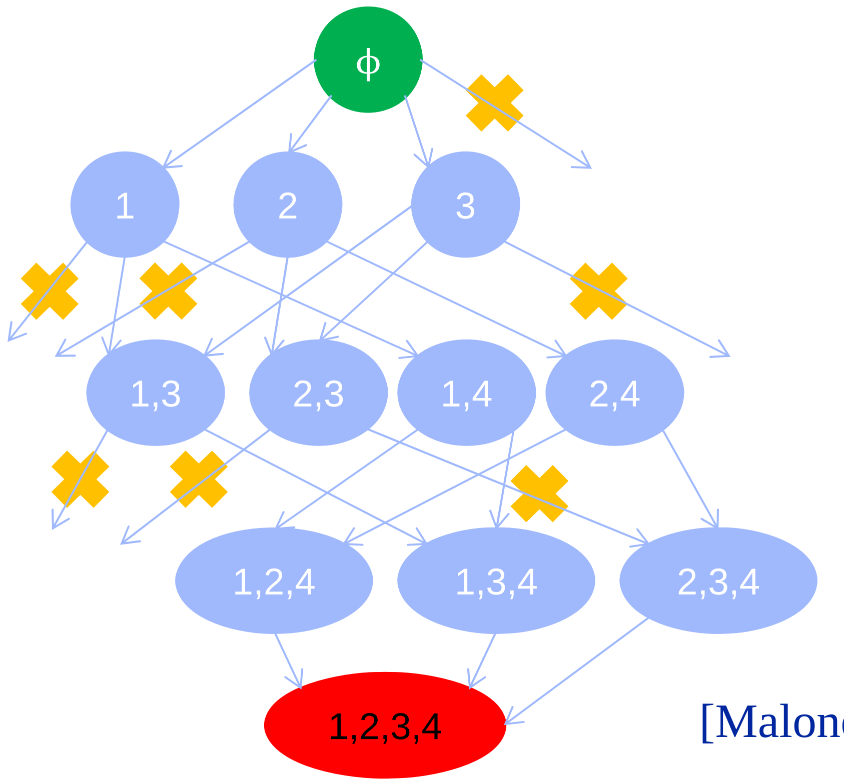
BFBnB algorithm



[Malone, Yuan, Hansen, UAI-11]

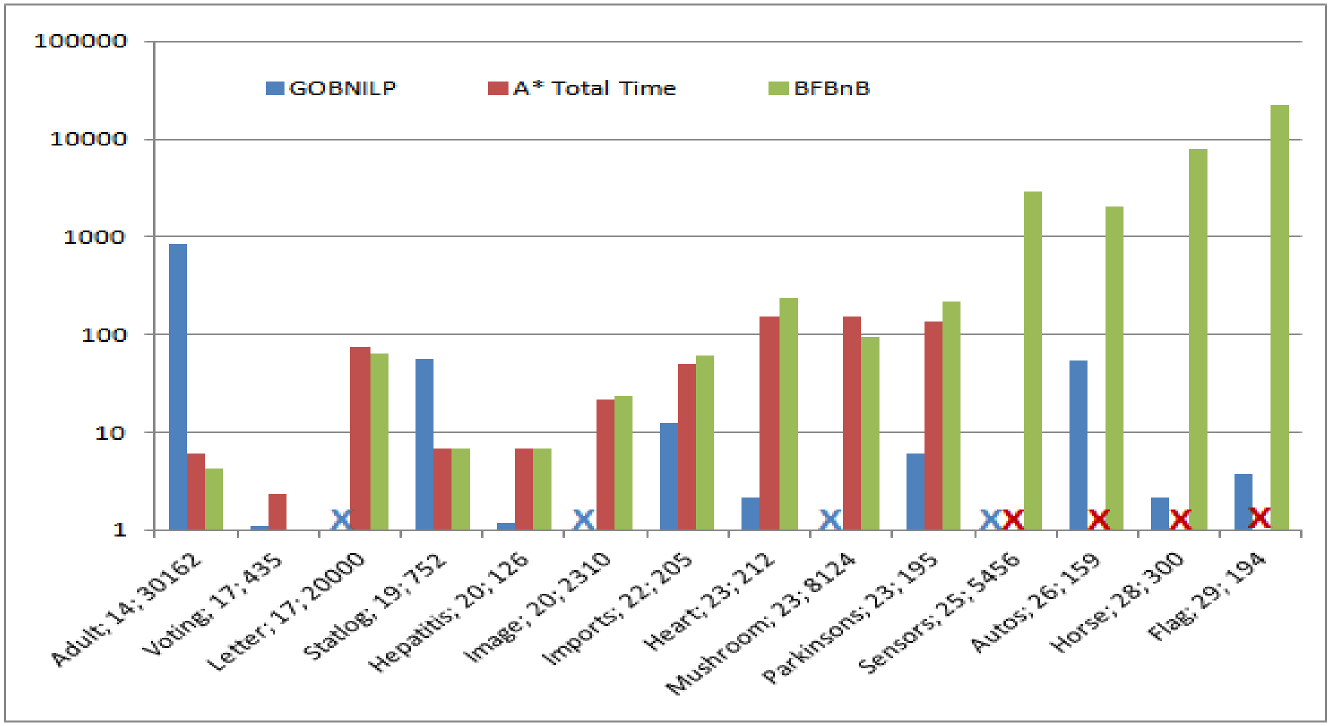
Pruning in BFBnB

- **For pruning, estimate an upper bound solution before search**
  - **Can be done using anytime window A***
- **Prune a node when *f*-cost > upper bound**



[Malone, Yuan, Hansen, UAI-11]

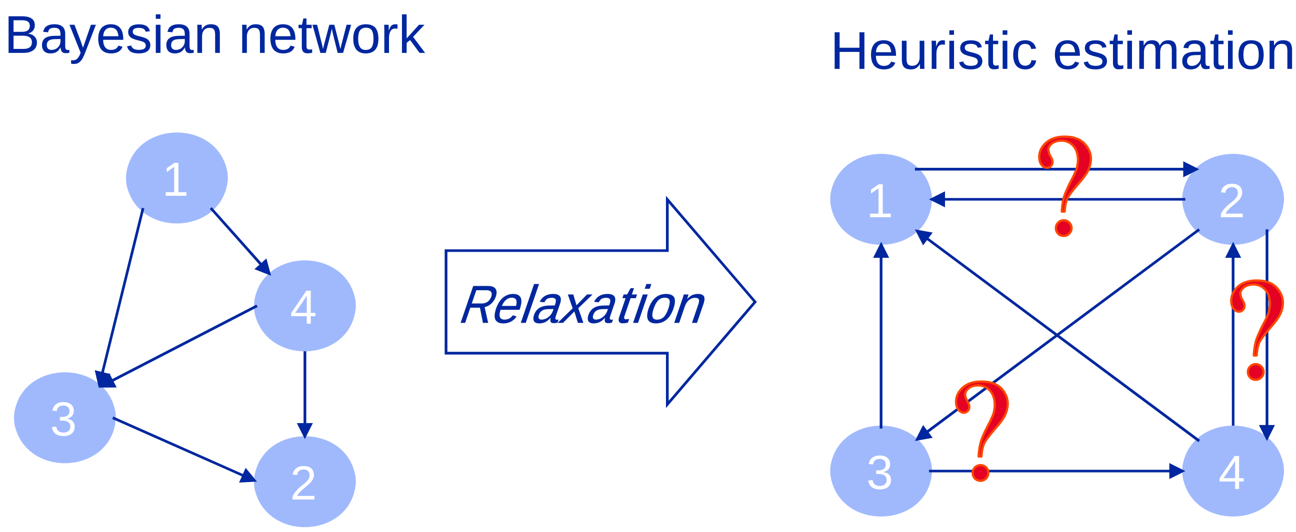
Performance of A* and BFBnB



A comparison of the total time (in seconds) for GOBNILP, A*, and BFBnB. An "X" means that the corresponding algorithm did not finish within the time limit (7,200 seconds) or ran out of memory in the case of A*.

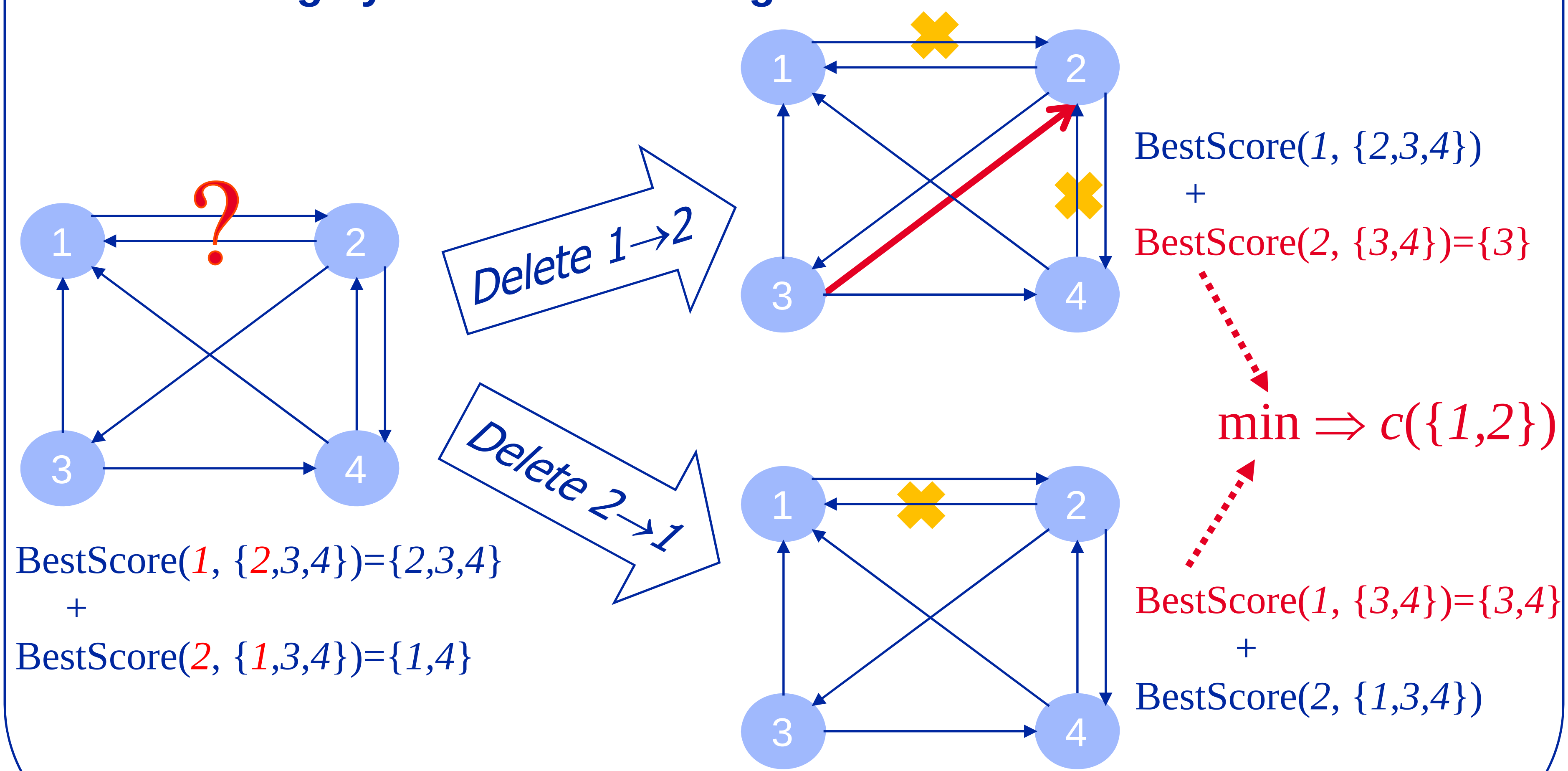
Drawback of simple heuristic

- **Let each variable to choose optimal parents from all the other variables**
- **Completely relaxes the acyclic constraint**



Potential solution

- **Breaking cycles to obtain a tighter heuristic**

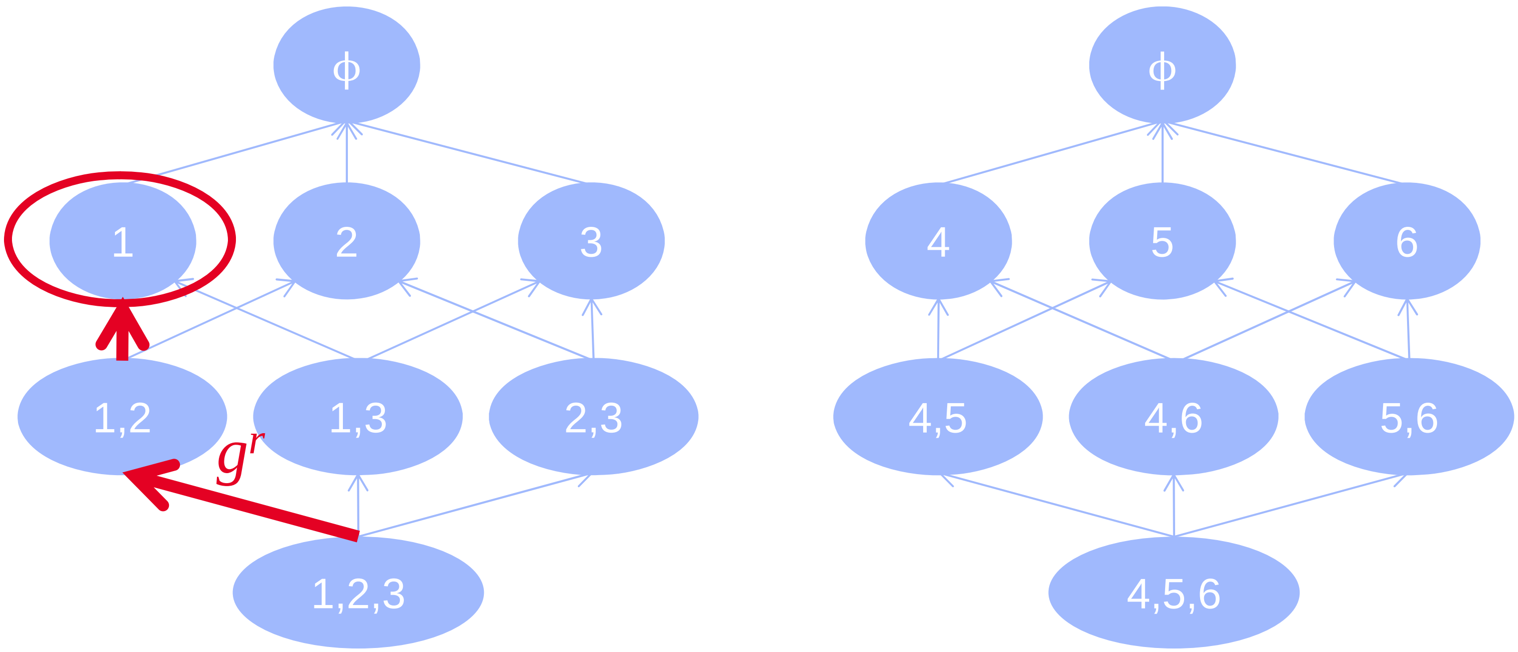


[Yuan, Malone, UAI-12]

Static k-cycle conflict heuristic

- **Also called static pattern database**
- **Calculate joint costs for all subsets of non-overlapping static groups by enforcing acyclicity within a group:**

$$\{1,2,3,4,5,6\} \Rightarrow \{1,2,3\}, \{4,5,6\}$$

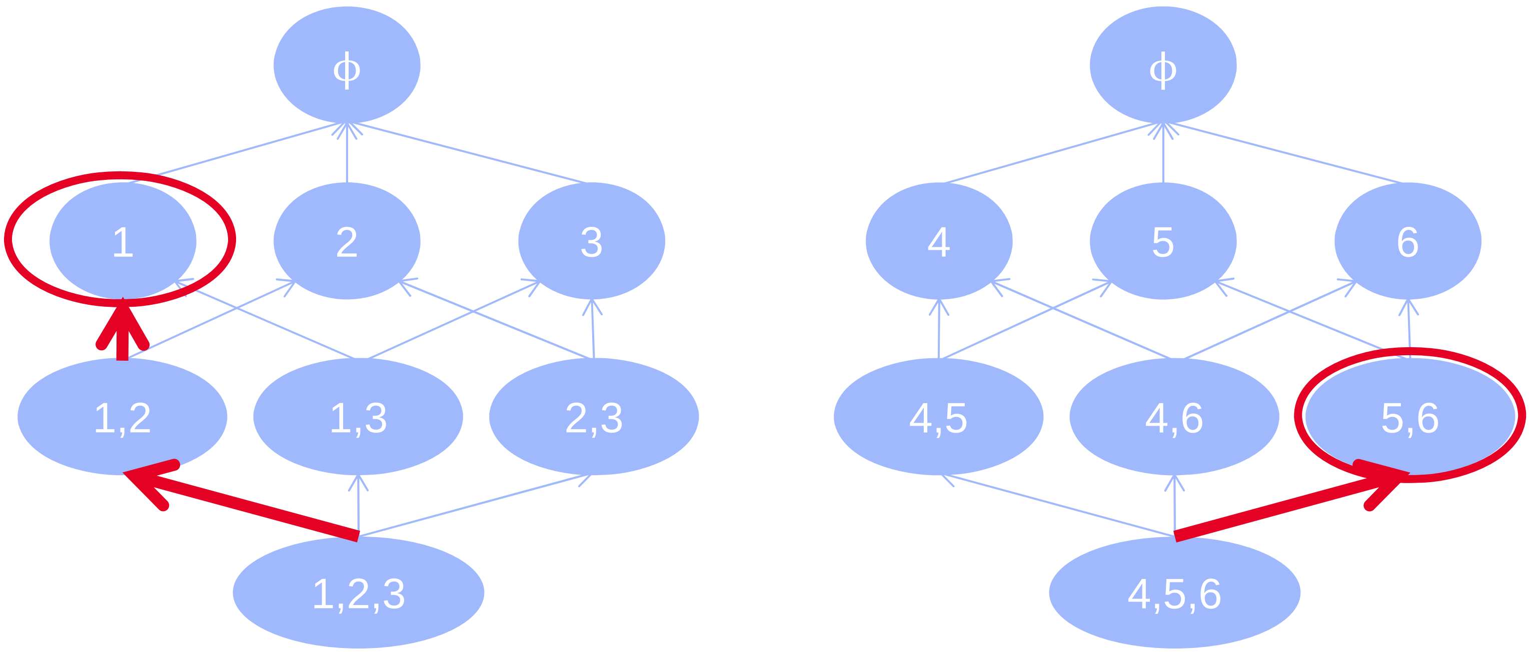


$$h(\{1\}) = g^r(\{1\})$$

[Yuan, Malone, UAI-12]

Computing heuristic value using static PD

- **Sum costs of pattern databases according to static grouping**



$$h(\{1,5,6\}) = h(\{1\}) + h(\{5,6\})$$

[Yuan, Malone, UAI-12]

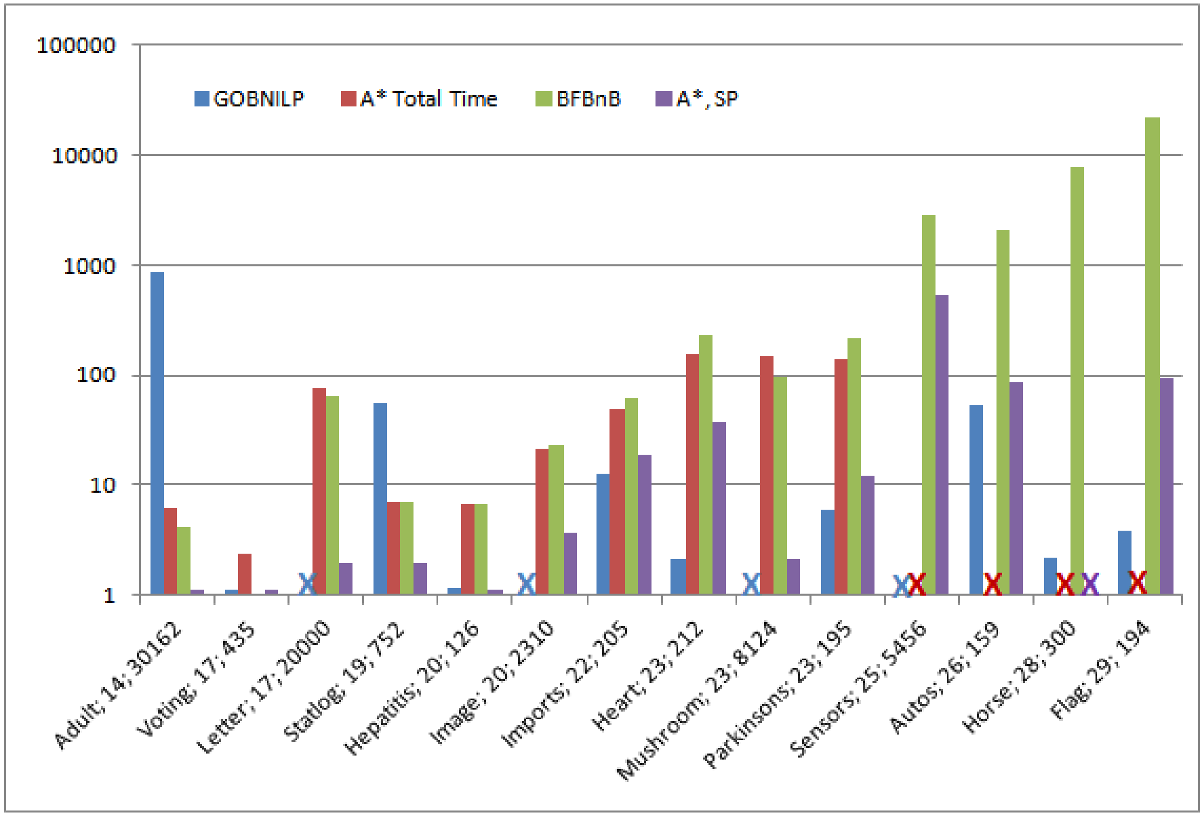
Properties of static *k*-cycle conflict heuristic

- **Theorem: The static *k*-cycle conflict heuristic is admissible**

- **Theorem: The static *k*-cycle conflict heuristic is consistent**

[Yuan, Malone, UAI-12]

Enhancing A* with static k-cycle conflict heuristic



A comparison of the search time (in seconds) for GOBNILP, A*, BFBnB, and A* with pattern database heuristic. An "X" means that the corresponding algorithm did not finish within the time limit (7,200 seconds) or ran out of memory in the case of A*.

Learning decomposition

- **Potentially Optimal Parent Sets (POPS)**
  - **Contain all *parent-child* relations**

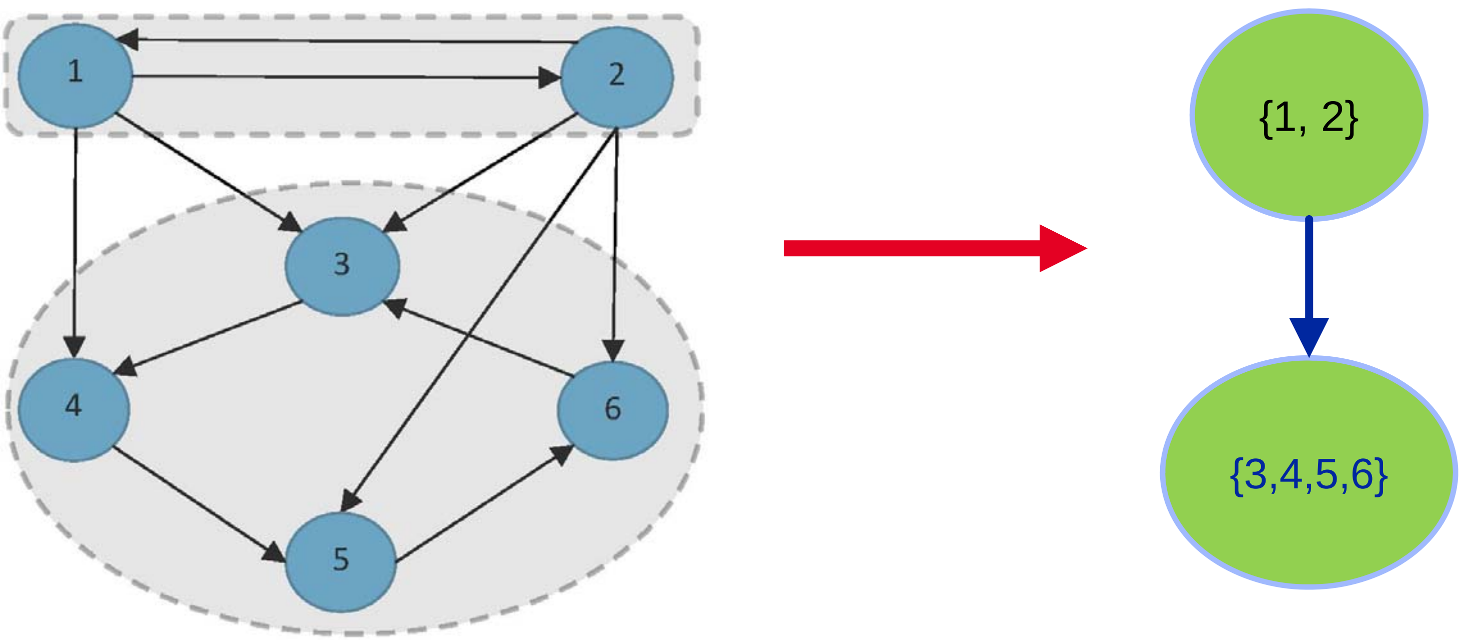
variable	POPS					
X_1	$\{X_2\}$	$\{\}$				
X_2	$\{X_1\}$	$\{\}$				
X_3	$\{X_1, X_2\}$	$\{X_2, X_6\}$	$\{X_1, X_6\}$	$\{X_2\}$	$\{X_6\}$	$\{\}$
X_4	$\{X_1, X_3\}$	$\{X_1\}$	$\{X_3\}$	$\{\}$		
X_5	$\{X_4\}$	$\{X_2\}$	$\{\}$			
X_6	$\{X_2, X_5\}$	$\{X_2\}$	$\{\}$			

- **Observation**: ***Not all variables can possibly be ancestors of the others.***
  - **E.g., any variables in $\{X_3, X_4, X_5, X_6\}$ can not be ancestor of X_1 or X_2**

[Fan, Malone, Yuan, UAI-14]

POPS Constraints

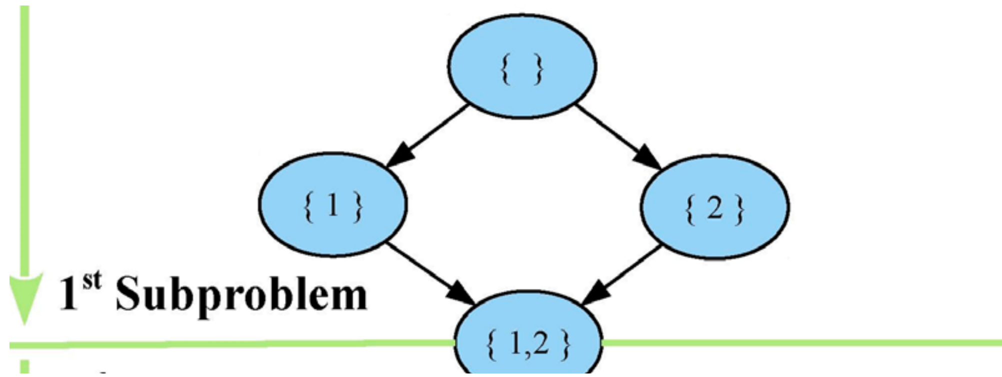
- **Parent Relation Graph**
  - **Aggregate all the *parent-child* relations in POPS Table**
- **Component Graph**
  - **Strongly Connected Components (SCCs)**
  - **Provide ancestral constraints**

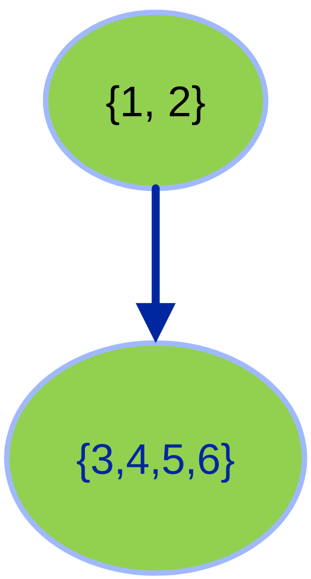


[Fan, Malone, Yuan, UAI-14]

POPS Constraints

- **Decompose the problem**
  - Each SCC corresponds to a smaller subproblem
  - Each subproblem can be solved independently.

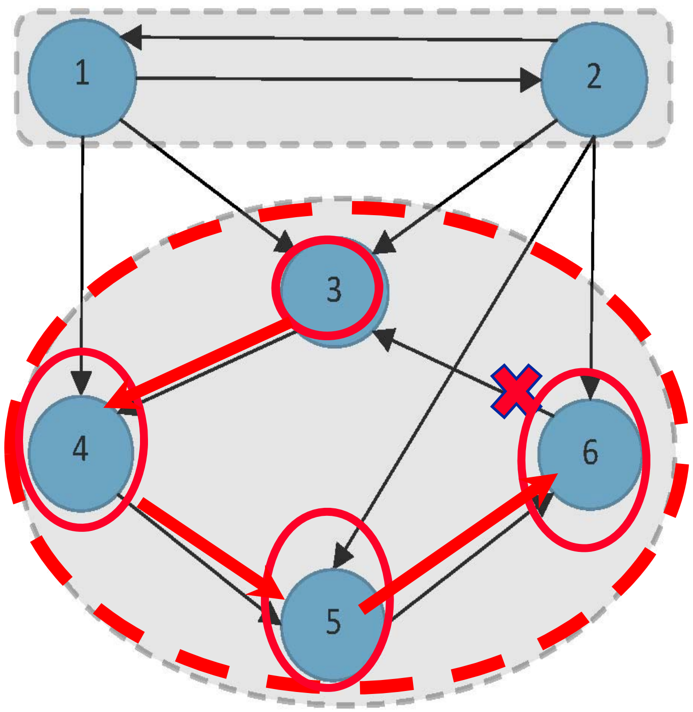


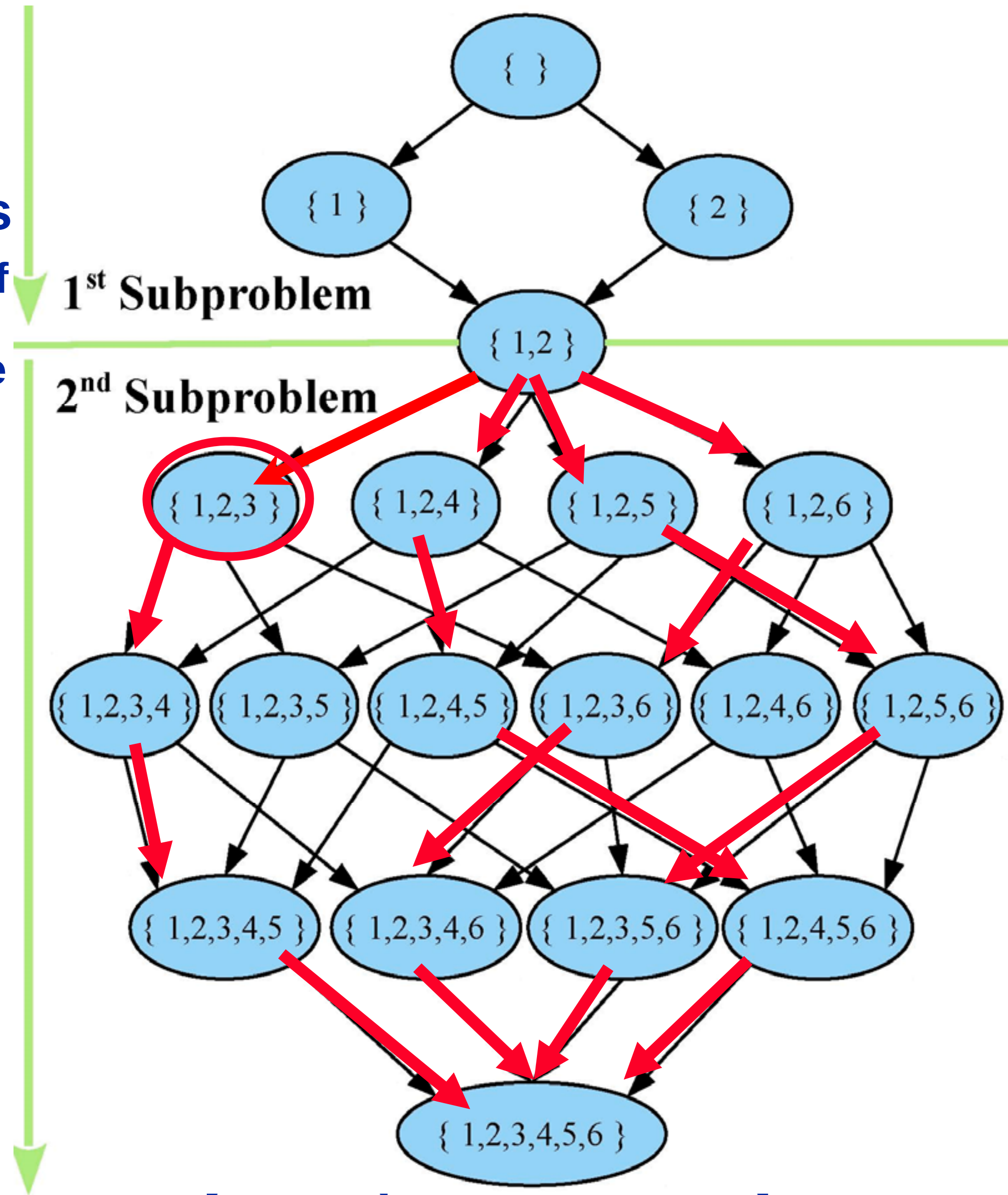


[Fan, Malone, Yuan, UAI-14]

POPS Constraints

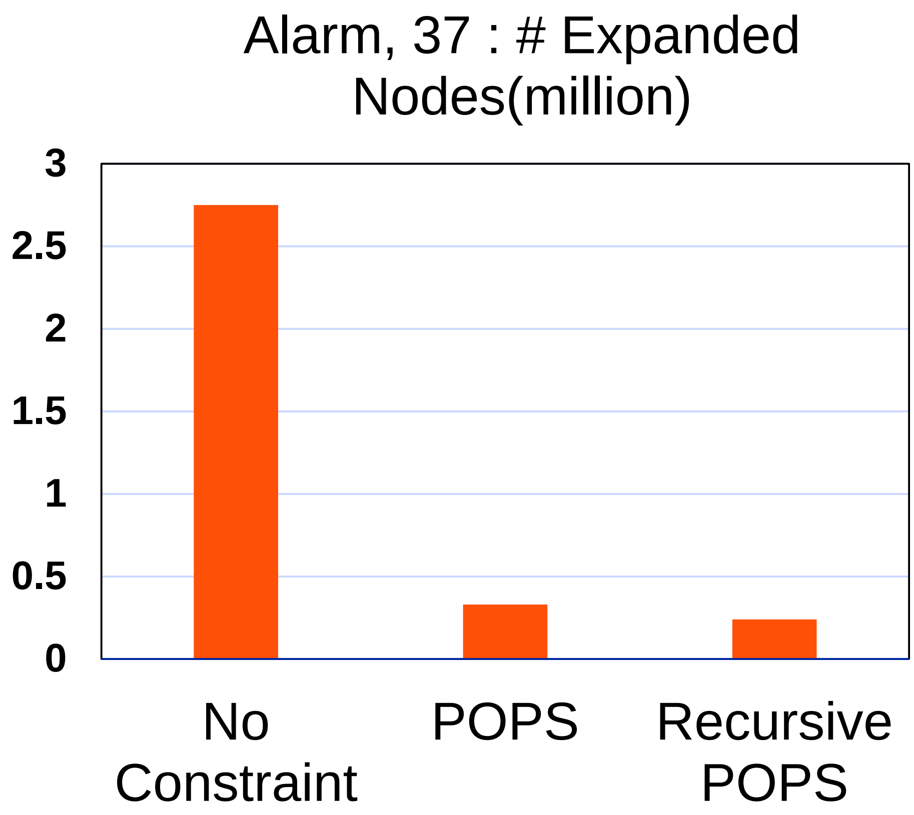
- **Recursive POPS Constraints**
  - Selecting the parents for one of the variables has the effect of removing that variable from the parent relation graph.

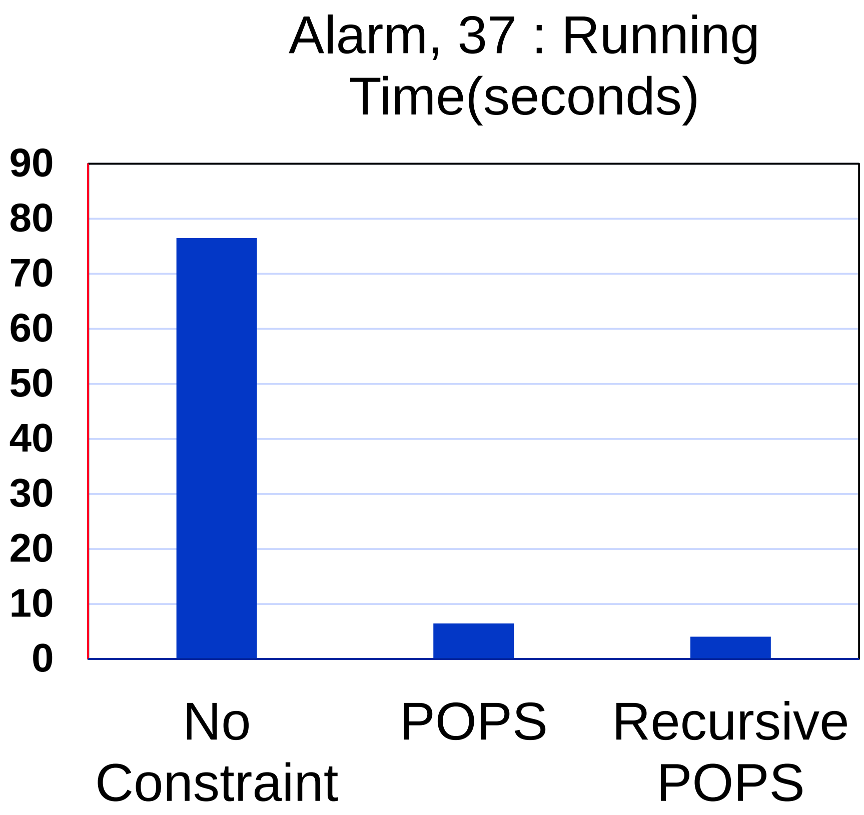




[Fan, Malone, Yuan, UAI-14]

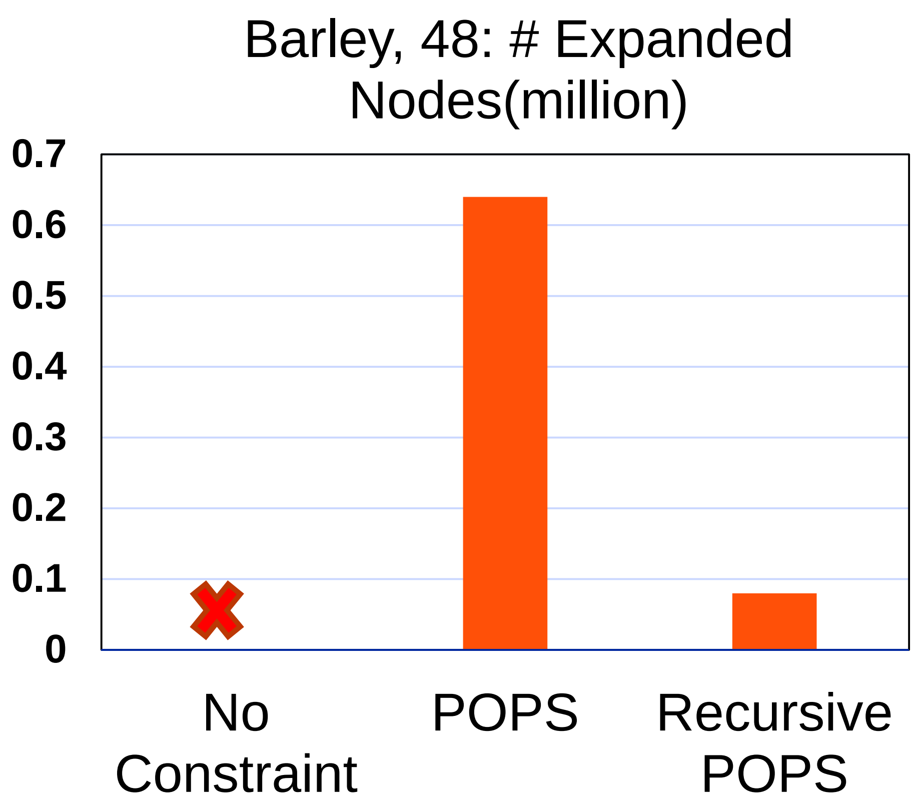
Evaluating POPS and recursive POPS constraints

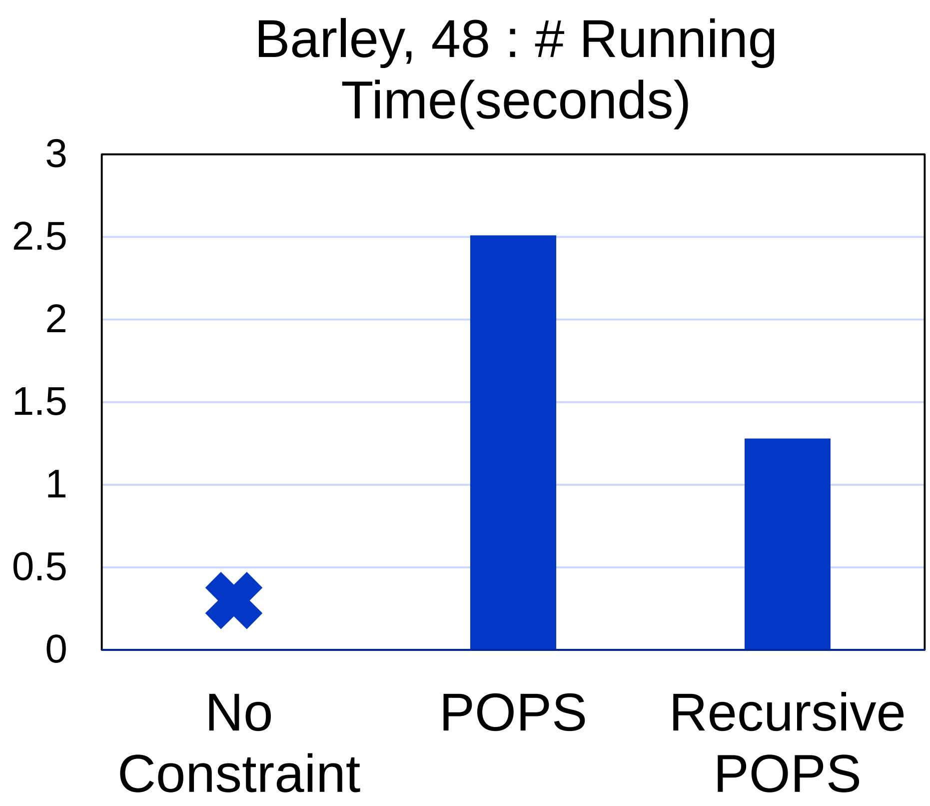




[Fan, Malone, Yuan, UAI-14]

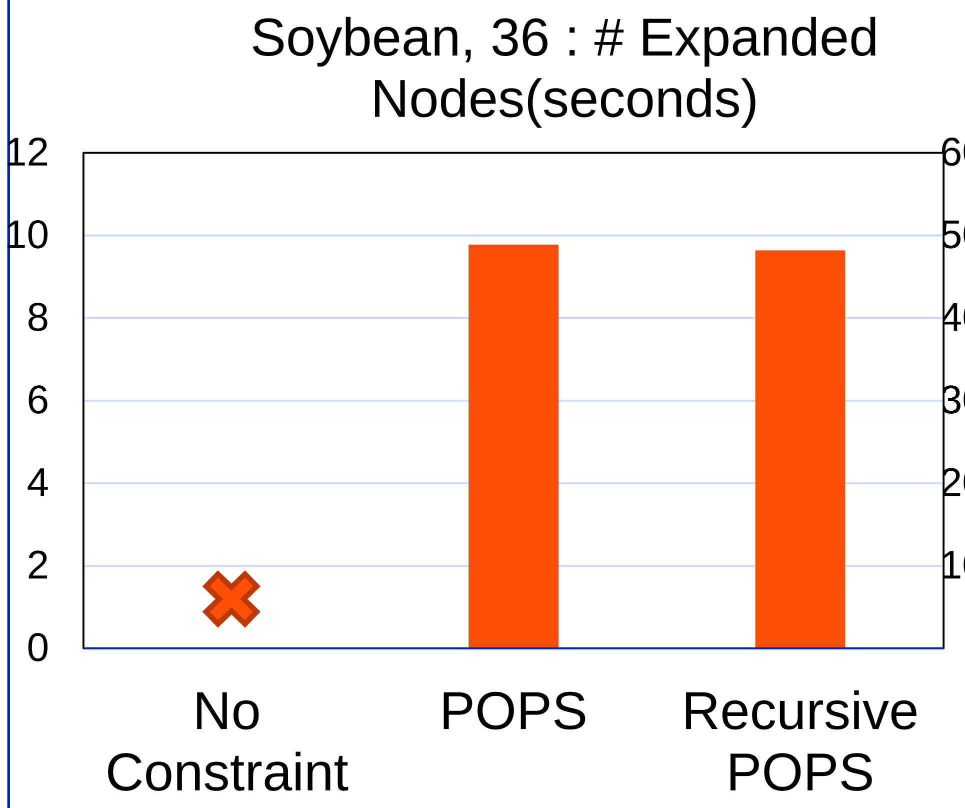
Evaluating POPS and recursive POPS constraints

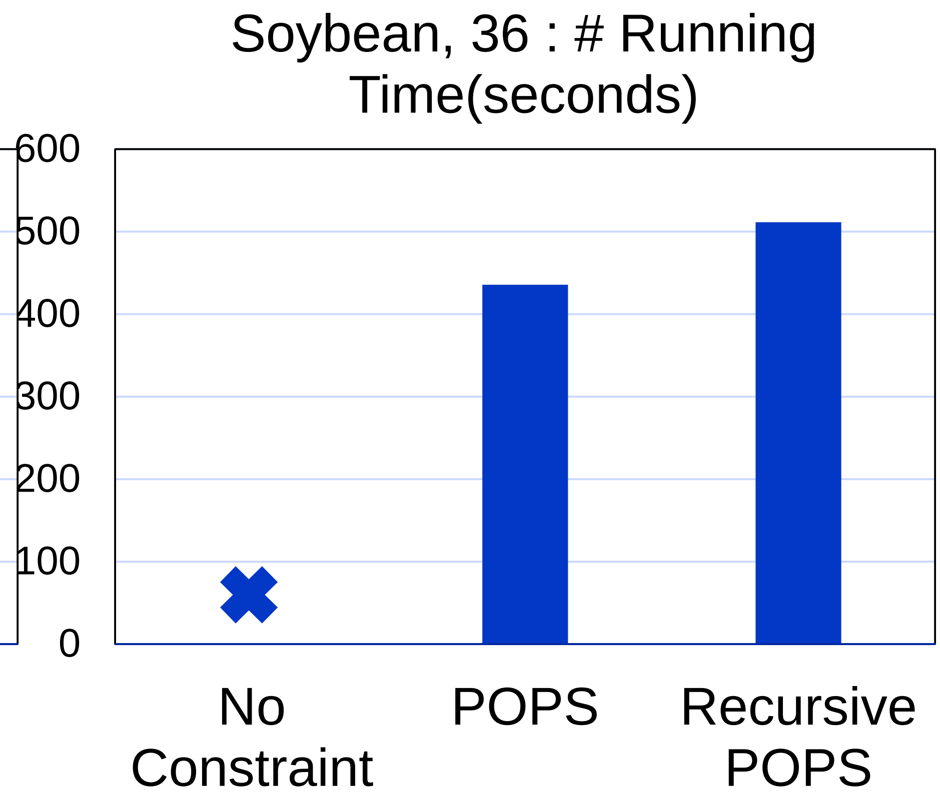




[Fan, Malone, Yuan, UAI-14]

Evaluating POPS and recursive POPS constraints





[Fan, Malone, Yuan, UAI-14]

Grouping in static k-cycle conflict heuristic

- Tightness of the heuristic highly depends on the grouping
- Characteristics of a good grouping
  - Reduce directed cycles between groups
  - Enforce as much acyclicity as possible

[Fan, Yuan, AAAI-15]

Existing grouping methods

- Create an undirected graph as skeleton
  - Parent grouping: connecting each variable to potentials parents in the best POPS
  - Family grouping: use Min-Max Parent Child (MMPC) [Tsarmardinos et al. 06]
- Use independence tests in MMPC to estimate edge weights
- Partition the skeleton into balanced subgraphs
  - by minimizing the total weights of the edges between the subgraphs

[Fan, Yuan, AAAI-15]

Advanced grouping

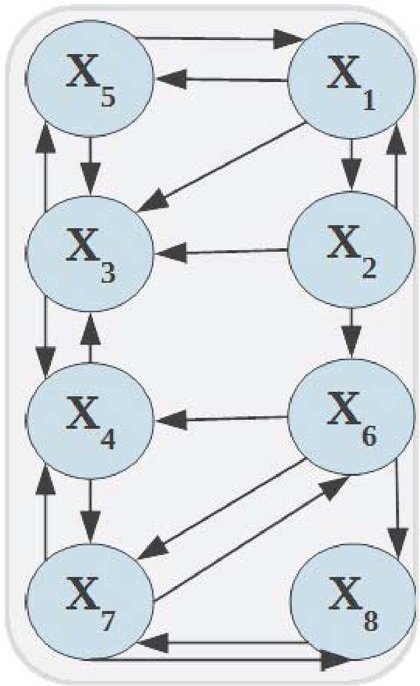
- **The potentially optimal parent sets (POPS) capture all possible relations between variables**

var.	POPS			
X_1	$\{X_2\}$	$\{X_5\}$		
X_2	$\{X_1\}$			
X_3	$\{X_1, X_5\}$	$\{X_1, X_2\}$	$\{X_2, X_4\}$	$\{X_1\}$
X_4	$\{X_3\}$	$\{X_6\}$	$\{X_7\}$	
X_5	$\{X_1, X_3\}$	$\{X_3\}$		
X_6	$\{X_2, X_7\}$	$\{X_7\}$		
X_7	$\{X_8\}$	$\{X_6, X_4\}$		
X_8	$\{X_6\}$	$\{X_7\}$		

- **Observation: Directed cycles in the heuristic originate from the POPS**

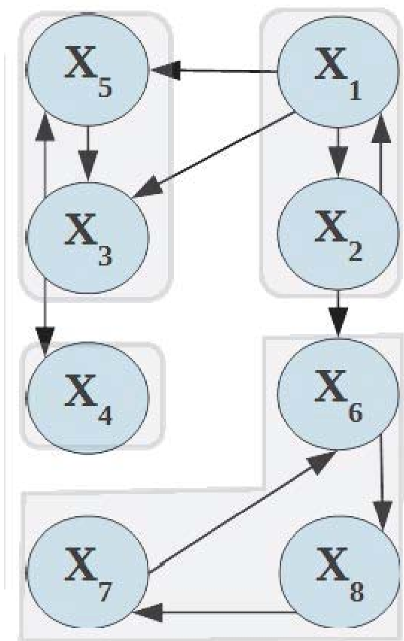
[Fan, Yuan, AAAI-15]

Parent relation graphs from all POPS

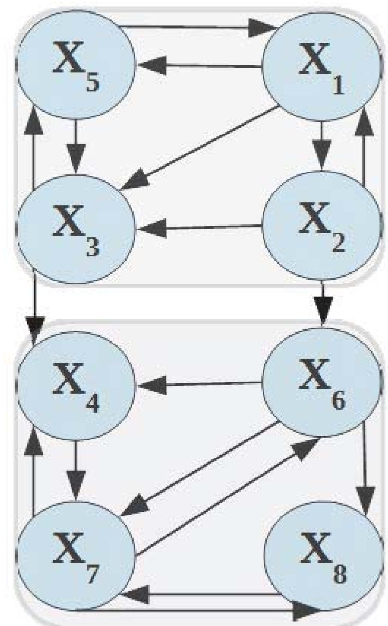


[Fan, Yuan, AAAI-15]

Parent relation graph from top-K POPS



K = 1



K = 2

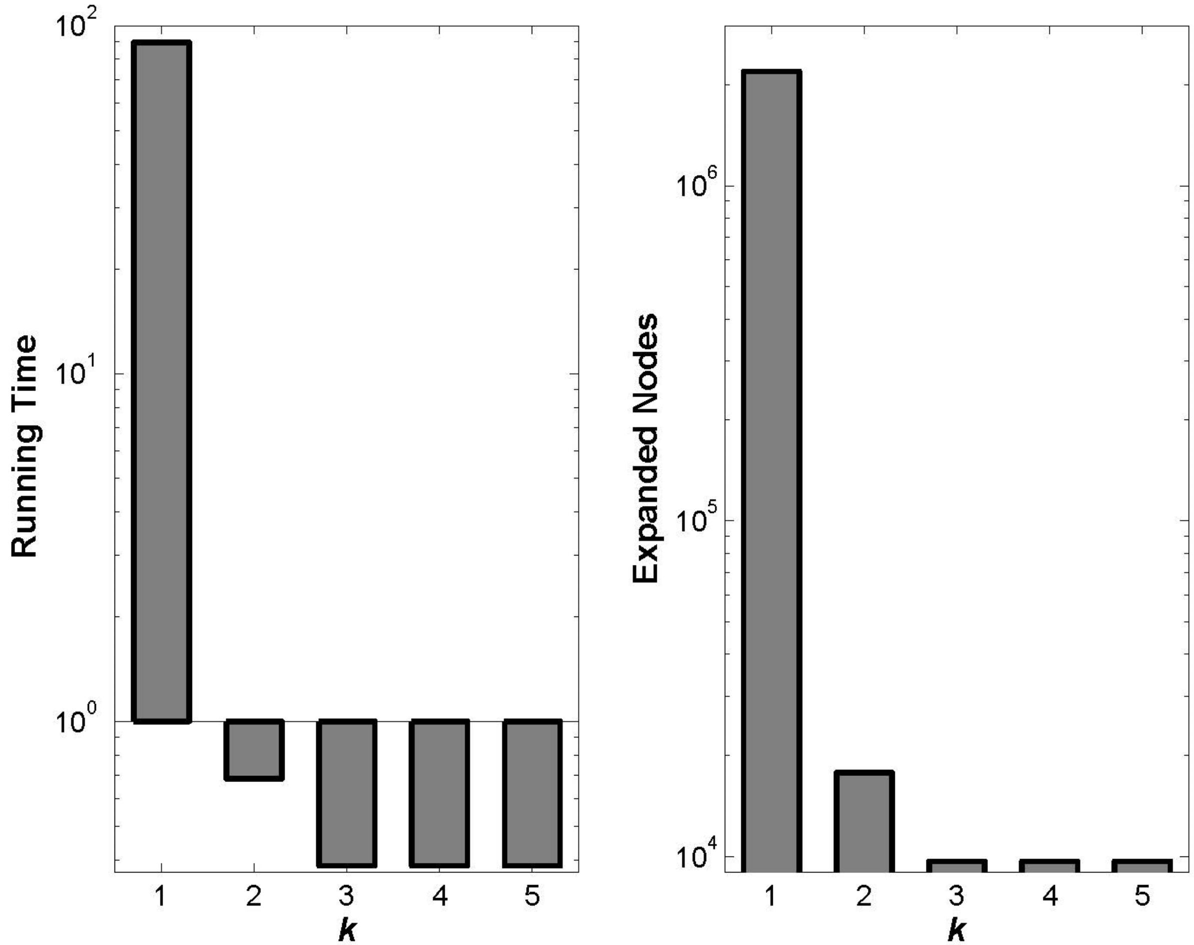
[Fan, Yuan, AAAI-15]

Component grouping

- γ: the size of the largest pattern database that can be created
- Use parent grouping if the largest SCC in top-1 graph is already larger than γ
- Otherwise, use component grouping
  - For K = 1 to $\max_i |POPS|_i$
    - » Use top-K POPS of each variable to create a parent relation graph
    - » If the graph has only one SCC or a too large SCC, return
    - » Divide the SCCs into two or more groups by using a Prim-like algorithm
  - Return feasible grouping of largest K

[Fan, Yuan, AAAI-15]

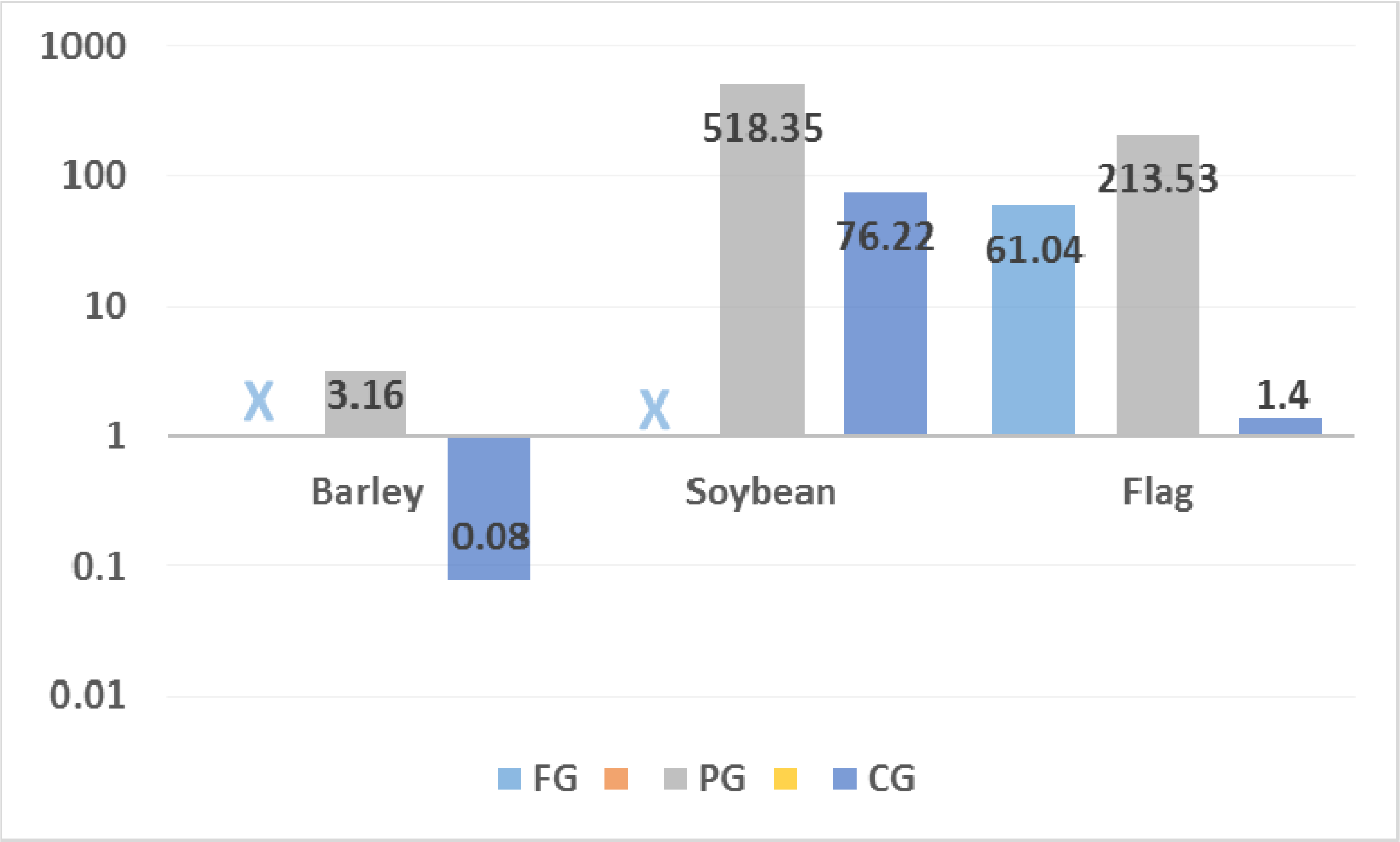
Parameter K



[Fan, Yuan, AAAI-15]

The running time and number of expanded nodes needed by A* to solve *Soybeans* with different K.

Comparing grouping methods



[Fan, Yuan, AAAI-15]

Summary

- **Formulation:**
  - **learning optimal Bayesian networks as a shortest path problem**
  - **Standard heuristic search algorithms applicable, e.g., A*, BFBnB**
  - **Design of upper/lower bounds critical for performance**
- **Extra information extracted from data enables**
  - **Creating ancestral graphs for decomposing the learning problem**
  - **Creating better grouping for the static k-cycle conflict heuristic**
- **Take home message: Methodology and data work better as a team!**
- **Open source software available from**
  - **http://urlearning.org**

Acknowledgements

- NSF CAREER grant, IIS-0953723
- NSF IIS grant, IIS-1219114
- PSC-CUNY Enhancement Award
- The Academy of Finland (COIN, 251170)

References

- Xiannian Fan, Changhe Yuan. An Improved Lower Bound for Bayesian Network Structure Learning. In Proceedings of the 29th AAAI Conference (AAAI-15). Austin, Texas. 2015.
- Xiannian Fan, Brandon Malone, Changhe Yuan. Finding Optimal Bayesian Networks Using Constraints Learned from Data. In Proceedings of the 30th Annual Conference on Uncertainty in Artificial Intelligence (UAI-14). Quebec City, Quebec. 2014.
- Xiannian Fan, Changhe Yuan, Brandon Malone. Tightening Bounds for Bayesian Network Structure Learning. In Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14). Quebec City, Quebec. 2014.
- Changhe Yuan, Brandon Malone. Learning Optimal Bayesian Networks: A Shortest Path Perspective. Journal of Artificial Intelligence Research (JAIR). 2013.
- Brandon Malone, Changhe Yuan. Evaluating Anytime Algorithms for Learning Optimal Bayesian Networks. In Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI-13). Seattle, Washington. 2013.
- Brandon Malone, Changhe Yuan. A Depth-first Branch and Bound Algorithm for Learning Optimal Bayesian Networks. IJCAI-13 Workshop on Graph Structures for Knowledge Representation and Reasoning (GKR'13). Beijing, China. 2013.
- Changhe Yuan, Brandon Maone. An Improved Admissible Heuristic for Learning Optimal Bayesian Networks. In Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI-12). Catalina Island, CA. 2012.
- Brandon Malone. Learning optimal Bayesian networks with heuristic search. PhD Dissertation. Department of Computer Science and Engineering, Mississippi State University. July, 2012.
- Brandon Malone, Changhe Yuan. A Parallel, Anytime, Bounded Error Algorithm for Exact Bayesian Network Structure Learning. In Proceedings of the Sixth European Workshop on Probabilistic Graphical Models (PGM-12). Granada, Spain. 2012.
- Changhe Yuan, Brandon Malone and Xiaojian Wu. Learning Optimal Bayesian Networks Using A* Search. 22nd International Joint Conference on Artificial Intelligence (IJCAI-11). Barcelona, Catalonia, Spain, July 2011.
- Brandon Malone, Changhe Yuan, Eric Hansen and Susan Bridges. Memory-Efficient Dynamic Programming for Learning Optimal Bayesian Networks, 25th AAAI Conference on Artificial Intelligence (AAAI-11). San Francisco, CA. August 2011.
- Brandon Malone, Changhe Yuan, Eric Hansen and Susan Bridges. Improving the Scalability of Optimal Bayesian Network Learning with Frontier Breadth-First Branch and Bound Search, 27th Conference on Uncertainty in Artificial Intelligence (UAI-11). Barcelona, Catalonia, Spain, July 2011.