Statistically Efficient Greedy Equivalence Search

David Maxwell Chickering
Microsoft Research
Redmond, WA 98052
dmax@microsoft.com

Abstract

We establish the theoretical foundation for statistically efficient variants of the Greedy Equivalence Search algorithm. If each node in the generative structure has at most k parents, we show that in the limit of large data, we can recover that structure using greedy search with operator scores that condition on at most k variables. We present simple synthetic experiments that compare a backward-only variant of the new algorithm to GES using finite data, showing increasing benefit of the new algorithm as the complexity of the generative model increases.

1 INTRODUCTION

Greedy Equivalence Search (GES) is a score-based search algorithm that searches over the space of equivalence classes of Bayesian-network structures. The algorithm is appealing because (1) it explicitly (and greedily) searches for the highest-scoring model, and (2) in the large-sample limit, assuming the generative distribution is perfect with respect to a DAG model $\mathcal{G}$ defined over the observables, it is guaranteed to result with $\mathcal{G}$'s equivalence class; in other words, in the large-sample limit there are no local maxima in the search space and $\mathcal{G}$ is the global maximum. The GES algorithm consists of two simple phases: Forward Equivalence Search (FES) and Backward Equivalence Search (BES). In FES, we greedily add edges until we reach a local maximum, and in BES we greedily remove edges until we reach a local maximum.

There are two potential problems with the GES algorithm in practice. First, the branching factor of the search space can grow to be exponential in the number of nodes if the models reached by FES are complex. Chickering and Meek (2015) solved this problem by introducing *Selective Greedy Equivalence Search* (SGES), which is an implementation of GES that in the worst case completes in polynomial time, yet retains the large-sample correctness guarantees.

The second potential problem, which is addressed in the current paper, is that of *statistical* efficiency: we expect the volume of data needed to attain the large-sample guarantees of GES to grow with the number of variables used to score the search operators. This is particularly problematic in discrete domains, where the scoring functions are effectively estimating separate multinomial distributions for each configuration of the values of a node's parents; the number of these configurations grows exponentially with the number of parents. If GES reaches highly-connected models as it traverses the search space, it can prescribe calls to the scoring function that condition on almost every node in the domain.

In this paper, we show how to score BES search operators in highly-connected models using low-order calls to the scoring function. We show that if in the generative model no node has more than k parents, we get the large-sample guarantees of GES using calls to the scoring function that "condition on" at most k nodes, and are functions of at most $k + 2$ variables. We guarantee computational efficiency by combining this method with the search-space pruning of SGES, and we call the resulting algorithm *Statistically Efficient Greedy Equivalence Search* or *SE-GES* for short.

We do not explore variants of the *forward* phase of SE-GES that can be combined with our modified backward phase. For practical variants of SE-GES, we want to modify FES to reach more complex models than what would be prescribed by the 'normal' (i.e., not statistically efficient) scoring function. In our experiments, we avoid this issue by starting SE-GES with the fully-connected model.

Our paper is organized as follows. In Section 2 we dis-

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

cuss related work. In Section 3, we describe our notation. In Section 4, we provide a detailed description of both GES and SGES. In Section 5, we provide the details of our new SE-GES algorithm. We also present the main theorems that demonstrate the large-sample optimality of SE-GES, although we defer the proofs of these results to the supplemenary material. In Section 6, we present the results of a synthetic-data experiment that compares SE-GES to both GES and SGES. Finally, we conclude in Section 7.

2 RELATED WORK

The relationship between GES and SE-GES mirrors, to some extent, the relationship between the SGS algorithm and the PC algorithm (Spirtes et al., 2000), which are constraint-based approaches to learning Bayesian-network structures. Like GES, the SGS algorithm can require scores that are functions of a large number of variables, even if the generative distribution is sparse. Like SE-GES, the PC algorithm solves this problem by iteratively increasing a complexity bound.

Although SE-GES is based on scores, it traverses through dense regions of the search space by scoring low-order "independence facts". As a result, SE-GES could be considered a *hybrid* approach that uses both a score and an independence oracle. There are many other such hybrid approaches to learning that leverage the constraints from an independence test to reduce the complexity of a score-based search algorithm. Examples include the *Sparse Candidate* method of Friedman et al. (1999), the *max-min hill-climbing* algorithm Tsamardinos et al. (2006), and the *H2PC* algorithm of Gasse at al. (2014). Nandy et al. (2018) prove both classical and high-dimensional consistency for two hybrid variants of GES.

We call SE-GES *statistically efficient* without any formal treatment of sample complexity; our main result is that we can limit the size of the conditioning sets during search to be as small as possible. Other researchers have studied sample complexity of structure search more formally, including Kalisch and Buhlmann (2007), who prove uniform consistency for the PC algorithm, and Zuk et al. (2006) who provide asymptotic upper and lower bounds on the number of samples that are needed for the generative model to be globally optimal in the score-based setting.

3 NOTATION

We use upper-case letters (e.g., A) to denote variables, and we use bold-face letters to represent sets of variables (e.g., $\mathbf{A}$). We use calligraphic letters (e.g., $\mathcal{G}, \mathcal{E}$) to denote statistical models and graphs. A *Bayesian-network model* for a set of variables $\mathbf{U}$ is a pair $(\mathcal{G}, \boldsymbol{\theta})$. $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a directed acyclic graph—or *DAG* for short—consisting of nodes in one-to-one correspondence with the variables and directed edges that connect those nodes. $\boldsymbol{\theta}$ is a set of parameter values that specify all of the conditional probability distributions. The Bayesian network represents a joint distribution over $\mathbf{U}$ that factors according to the structure $\mathcal{G}$. The structure $\mathcal{G}$ of a Bayesian-network model represents the independence constraints that must hold in the distribution. The set of all independence constraints implied by the structure $\mathcal{G}$ can be characterized by the *Markov conditions*, which are the constraints that each variable is independent of its non-descendants given its parents. All other independence constraints follow from properties of independence. A distribution defined over the variables from $\mathcal{G}$ is *perfect with respect to* $\mathcal{G}$ if the set of independencies in the distribution is equal to the set of independencies implied by the structure $\mathcal{G}$.

Two DAGs $\mathcal{G}$ and $\mathcal{G}'$ are *equivalent*[1]—denoted $\mathcal{G} \approx \mathcal{G}'$—if the independence constraints in the two DAGs are identical. Because equivalence is reflexive, symmetric, and transitive, the relation defines a set of equivalence classes over network structures. An equivalence class of DAGs is an *independence map* (*IMAP*) of another equivalence class of DAGs if all independence constraints implied by the first class are also implied by the second class. For two DAGs $\mathcal{G}$ and $\mathcal{H}$, we use $\mathcal{G} \leq \mathcal{H}$ to denote that the equivalence class of $\mathcal{H}$ is an IMAP of the equivalence class of $\mathcal{G}$; we use $\mathcal{G} < \mathcal{H}$ when $\mathcal{G} \leq \mathcal{H}$ and the two equivalence classes are not the same. Verma and Pearl (1991) show that two DAGs are equivalent if and only if they have the same *skeleton* (i.e., the graph resulting from ignoring the directionality of the edges) and the same *v-structures* (i.e., pairs of edges $X \rightarrow Y$ and $Y \leftarrow Z$ where X and Z are not adjacent). As a result, we can use a *partially directed acyclic graph*—or *PDAG* for short—to represent an equivalence class of DAGs: for a PDAG $\mathcal{P}$, the equivalence class of DAGs is the set that has the same skeleton and the same v-structures as $\mathcal{P}$[2].

We use $\mathbf{NA}_{X,Y}^{\mathcal{P}}$ to denote, within PDAG $\mathcal{P}$, the set of nodes that are *neighbors* of X (i.e., connected with an undirected edge) and also adjacent to Y (i.e., without re-

[1] We make the standard conditional-distribution assumptions of multinomials for discrete variables and Gaussians for continuous variables so that if two DAGs have the same independence constraints, then they can also model the same set of distributions.

[2] The definitions for the skeleton and set of v-structures for a PDAG are the obvious extensions to these definitions for DAGs.

gard to whether the connecting edge is directed or undirected).

An edge in $\mathcal{G}$ is *compelled* if it exists in every DAG that is equivalent to $\mathcal{G}$. If an edge in $\mathcal{G}$ is not compelled, we say that it is *reversible*. A *completed* PDAG (*CPDAG*) $\mathcal{C}$ is a PDAG with two additional properties: (1) for every directed edge in $\mathcal{C}$, the corresponding edge in $\mathcal{G}$ is compelled and (2) for every undirected edge in $\mathcal{C}$ the corresponding edge in $\mathcal{G}$ is reversible. Unlike non-completed PDAGs, the CPDAG representation of an equivalence class is unique.

We use $\mathbf{Pa}_Y^{\mathcal{P}}$ and $\mathbf{Ch}_Y^{\mathcal{P}}$ to denote the *parents* and *children*, respectively, of node Y in $\mathcal{P}$. We use $\mathbf{CC}_{X,Y}^{\mathcal{P}}$ to denote the common children of X and Y in $\mathcal{P}$. For any pair of nodes X and Y in some DAG $\mathcal{H}$, we use $\mathbf{D}_{X,Y}^{\mathcal{H}}$ to denote the set of all descendants of the common children $\mathbf{CC}_{X,Y}^{\mathcal{H}}$. Importantly, $\mathbf{D}_{X,Y}^{\mathcal{H}}$ includes $\mathbf{CC}_{X,Y}^{\mathcal{H}}$ as degenerate descendants.

4 GREEDY EQUIVALENCE SEARCH AND SELECTIVE GREEDY EQUIVALENCE SEARCH

The GES algorithm is a two-phase greedy search through the space of DAG equivalence classes. The algorithm represents the states of the search using CPDAGs, performing transformation operators to these graphs to move in the space. Each operator corresponds to a DAG edge modification, and is scored using a DAG scoring function that we assume has three properties. First, we assume the scoring function is *score equivalent*, which means that it assigns the same score to equivalent DAGs. Second, we assume the scoring function is *locally consistent*, which means that, given enough data, (1) if the current state *is not* an IMAP of $\mathcal{G}$, the score prefers edge additions that remove incorrect independencies, and (2) if the current state *is* an IMAP of $\mathcal{G}$, the score prefers edge deletions that remove incorrect dependencies. Finally, we assume the scoring function Sc is *decomposable*, which means we can express it as a sum of node-specific scores:

$$Sc(\mathcal{G}, \mathbf{D}) = \sum_{i=1}^{n} Sc(X_i, \mathbf{Pa}_i^{\mathcal{G}}) \qquad (1)$$

Note that the data $\mathbf{D}$ is implicit in the right-hand side of Equation 1.

We use *node score* to refer to a node-specific score. We call the second argument of each node score the *conditioning set* for the node score. We use the size of the conditioning set as our measure of its complexity; we are implicitly assuming that the number of states for each variable is constant.

Operator: $Delete(X, Y, \mathbf{H})$ applied to $\mathcal{C}$

Preconditions: (1) $X - Y$ or $X \rightarrow Y \in \mathcal{C}$, (2) $\mathbf{H} \subseteq \mathbf{NA}_{Y,X}^{\mathcal{C}}$, (3) $\overline{\mathbf{H}} = \mathbf{NA}_{Y,X}^{\mathcal{C}} \setminus \mathbf{H}$ is a clique.

Scoring:
$Sc(Y, \{\mathbf{Pa}_Y^{\mathcal{C}} \cup \overline{\mathbf{H}}\} \setminus X) - Sc(Y, \{\mathbf{Pa}_Y^{\mathcal{C}} \cup \overline{\mathbf{H}}\} \cup X)$

Transformation:

Remove edge between X and Y
foreach $H \in \mathbf{H}$ **do**
  Replace $Y - H$ with $Y \rightarrow H$
  if $X - H$ **then** Replace with $X \rightarrow H$;
end
Convert to CPDAG

Figure 1: Preconditions, scoring, and transformation algorithm for a $Delete$ operator.

The first phase of the GES—called *forward equivalence search* or *FES*—starts with an empty (i.e., no-edge) CPDAG and greedily applies *GES insert* operators until no operator has a positive score; these operators correspond precisely to the union of all single-edge additions to all DAG members of the current (equivalence class) state. After FES reaches a local maximum, GES switches to the second phase—called *backward equivalence search* or *BES*—and greedily applies *GES delete* operators until no operator has a positive score; these operators correspond precisely to the union of all single-edge deletions from all DAG members of the current state.

Theorem 1 (Chickering, 2002) *Let $\mathcal{C}$ be the CPDAG that results from applying the GES algorithm to m records sampled from a distribution that is perfect with respect to DAG $\mathcal{G}$. Then in the limit of large m, $\mathcal{C} \approx \mathcal{G}$.*

In Figure 1, we provide the details of the $Delete$ operator that is used during the second phase of GES. After applying the edge modifications in the transformation, the resulting PDAG $\mathcal{P}$ is not necessarily completed and hence we may have to convert $\mathcal{P}$ into the corresponding CPDAG representation. As shown by Chickering (2002), this conversion can be accomplished easily by using the structure of $\mathcal{P}$ to extract a DAG that we then convert into a CPDAG by undirecting all reversible edges. The complexity of this procedure for a $\mathcal{P}$ with n nodes and e edges is $O(n \cdot e)$, and requires no calls to the scoring function.

Note that in Figure 1 the score for the delete operator—which we will call the *deletion score*—is the difference

Algorithm 1: SELECTIVE-GEN-OPS($\mathcal{C}, X, Y, k$)

Input : CPDAG $\mathcal{C}$ with adjacent X,Y and parent limit k
Output: $\mathbf{Ops} = \{\mathbf{H}_1, \ldots, \mathbf{H}_m\}$
$\mathbf{Ops} \longleftarrow \emptyset$
$\mathbf{S} \longleftarrow$ Generate maximal cliques $\mathbf{C}_1, ..., \mathbf{C}_m$ from $\mathbf{NA}^{\mathcal{C}}_{Y,X}$
foreach $\mathbf{C}_i \in \mathbf{S}$ **do**
  $\mathbf{H}_0 \longleftarrow \mathbf{NA}^{\mathcal{C}}_{Y,X} \setminus \mathbf{C}_i$
  foreach $\mathbf{C} \subseteq \mathbf{C}_i$ *with* $|\mathbf{C}| \leq k$ **do**
    $\mathbf{H} \longleftarrow \mathbf{H}_0 \cup \mathbf{C}$
7   **if** KEEPOPERATOR($\mathcal{C}, X, Y, \mathbf{H}, k$) **then**
      Add $\mathbf{H}$ to Ops
return $\mathbf{Ops}$

between the node scores for Y under two conditioning sets: with X *excluded* from the conditioning set and with X *included* in the conditioning set. Thus the number of variables in the second node-score function is exactly one more than the number of variables in the first. We say that the *order* of a deletion operator $Delete(X, Y, \mathbf{H})$ is the number of variables in $\{\mathbf{Pa}^{\mathcal{C}}_Y \cup \overline{\mathbf{H}}\} \setminus X$. In other words, the order of the deletion operator is the number of variables excluding X and Y that are needed to compute its score. This means that to score an order-k delete operator, we need to use a node score that is a function of $k + 2$ variables.

The role of FES in the large-sample limit is to identify a state $\mathcal{C}$ for which $\mathcal{G} \leq \mathcal{C}$; Theorem 1 holds if FES is replaced with any algorithm that results in an IMAP of $\mathcal{G}$. The implementation details of such an algorithm can be important in practice because what constitutes a "large" amount of data depends on the order of the deletion operators. In our experiments, we use the degenerate algorithm that simply returns the complete (i.e., no missing edges) graph. This algorithm is guaranteed to return an IMAP regardless of the number of rows in the data because it imposes no independence constraints, but it is not a realistic candidate to use for GES in practice.

Assuming a computationally efficient implementation of FES, Chickering and Meek (2015) show how to restrict the set of deletion operators during BES so that the resulting *selective* GES algorithm (SGES) runs in polynomial time. The algorithm used by Chickering and Meek (2015) to generate the restricted set of operators, SELECTIVE-GEN-OPS, is reproduced as Algorithm 1 with an additional test for the condition "KEEPOPERATOR" shown in red on Line 7 that we discuss later; the original algorithm works as if this function always returns true.

Algorithm 2: SE-GES($\mathbf{D}$)

Input : Data $\mathbf{D}$
Output: CPDAG $\mathcal{C}$
1 $\mathcal{C} \longleftarrow$ FINDIMAP
$\mathbf{M}^{-1} \longleftarrow$ UNDEFINED for all node pairs
$k \longleftarrow 0$
Repeat
5 $\mathbf{M}^k \longleftarrow$ UPDATESEPARATORS($\mathbf{M}^{k-1}, k$)
6 $\mathcal{C} \longleftarrow$ SE-BES($\mathcal{C}, k$)
7 **if** every node in $\mathcal{C}$ has $\leq k$ parents **then**
    return $\mathcal{C}$
  else
    $k \longleftarrow k + 1$

Algorithm 3: SE-BES($\mathcal{C}, k$)

Input : CPDAG $\mathcal{C}$, Bound k
Output: CPDAG $\mathcal{C}$
Repeat
  $\mathbf{Ops} \longleftarrow \emptyset$
  foreach Adjacent (X, Y) in $\mathcal{C}$ **do**
    $\mathbf{Ops} \longleftarrow \mathbf{Ops} \cup$ SELECTIVE-GEN-OPS($\mathcal{C}, X, Y, k$)
5 $Op \longleftarrow$ highest-scoring operator in $\mathbf{Ops}$
  if Op score is negative **then**
    return $\mathcal{C}$
  else
    $\mathcal{C} \longleftarrow$ Apply Op to $\mathcal{C}$

5 STATISTICALLY EFFICIENT GREEDY EQUIVALENCE SEARCH

In this section, we introduce *Statistically Efficient Greedy Equivalence Search*. In Algorithm 2 we provide pseudo-code for SE-GES, and in Algorithm 3 we provide pseudo-code for its main subroutine SE-BES. To simplify the presentation, we assume that the data $\mathbf{D}$ passed into SE-GES and the minimal separating sets $\mathbf{M}^k$ updated by SE-GES are global variables that are available to SE-BES and all of its subroutines.

SE-GES works as follows. First, in Line 1, we apply an algorithm whose goal is to identify an IMAP of the generative model. Assuming infinite data, FES is guaranteed to identify an IMAP, but because of statistical-efficiency concerns, we may decide to use alternative algorithms instead. Next, we progressively iterate through the values of a bound k on the order of the deletion operators, initially set to zero. For each value of k we identify, on Line 5, all order-k separators $\mathbf{M}^k$; the separators capture inferred conditional independence relationships and are discussed in detail in Section 5.1. Next, on Line 6, we call SE-BES.

SE-BES is the same as the selective variant of Chickering and Meek (2015) described in Section 4, except that when given a delete operator of order larger than k, it either (1) filters that operator from consideration or (2) it uses the order-k separators to evaluate an alternative deletion score. The decision of which operators to filter is based on whether the alternative deletion score is provably "correct" in the large-sample limit.

SE-BES implements the deletion-operator filter using the KEEPOPERATOR predicate in the implementation of SELECTIVE-GEN-OPS shown in Algorithm 1; we define this predicate rigorously in Section 5.2. SE-BES applies the alternative deletion score to deletion operators of order greater than k on Line 5 in Algorithm 3; we describe the details of how this scoring works in Section 5.3.

After SE-BES completes, SE-GES checks on Line 7 if the resulting CPDAG is consistent with the bound k, meaning that each node in any DAG model contained in the CPDAG has at most k parents. Because all DAGs in an equivalence class have the same maximum number of parents (see Chickering, 1995), this check can be done by extracting an arbitrary DAG from $\mathcal{C}$ and counting parents. If the CPDAG is consistent with the bound, SE-GES terminates with the CPDAG as its final state; otherwise, it increments k by one and loops back to Line 5.

As we discuss in Section 5.2.3, in the large-sample limit, SE-GES is guaranteed to return the generative structure. Furthermore, if each node has at most k parents in that structure, SE-GES will terminate after running SE-BES with bound k, and therefore the algorithm will complete using only node scores that are functions of $k+2$ or fewer variables.

5.1 Minimal Separating Sets

In this section, we describe the separators $\mathbf{M}^k$ used by SE-GES. Each time SE-GES reaches Line 5, it identifies an *order k minimal separating set* for each pair of nodes. Intuitively, an order k minimal separating set for X and Y—denoted $\mathbf{M}^k_{XY}$—is any set of size at most k that renders X and Y independent in the data $\mathbf{D}$ and for which no subset of $\mathbf{M}^k_{XY}$ also has this property; note that we leave the data $\mathbf{D}$ implicit in our notation for $\mathbf{M}^k_{XY}$. Using the scoring function as our indicator of independence, we can define $\mathbf{M}^k_{XY}$ to be any minimal set of size at most k for which

$$Sc(Y, \mathbf{M}^k_{XY}) - Sc(Y, X \cup \mathbf{M}^k_{XY}) > 0 \qquad (2)$$

Note that—just like the deletion-operator scores—all of the order-k minimal separating sets are defined by node scores containing at most $k+2$ variables. Also, like other independence-based search algorithms in the literature, we can include a "significance" threshold to use in place of 0 in Equation 2 above.

We can show that for any score-equivalent scoring function, the role of X and Y is symmetric in the definition of the minimal separating sets. Note that even if the scoring function acts as a perfect independence oracle for the generative distribution, the minimal separating set for a pair of nodes is not unique; it turns out that *any* minimal set is sufficient to guarantee that SE-GES is asymptotically optimal. This leaves open various implementations that break ties based on an approximate low-order score.

If no order-k separating set between X and Y exists, we say that $\mathbf{M}^k_{XY}$ is *undefined*. By convention, we define the size of any undefined minimal separating set to be ∞. Thus, the test for $|\mathbf{M}^k_{XY}| \leq j$ will hold for any order-k separating set that is defined and contains no more than j elements.

Similar to the PC algorithm of Spirtes et al. (2000), we can limit our search for the separating sets of a particular order using the previously-computed lower-order dependencies. In some sense, we can view UPDATESEPARATORS as a partial implementation of the PC algorithm that SE-GES uses as a subroutine; the difference is that the separators are not interpreted *directly* as missing edges in the CPDAG, but are rather used as an aid to score the deletion operators.

Assuming there are n nodes in the domain, to compute $\mathbf{M}^k_{XY}$ for a pair of variables X and Y, we in the worst case have to enumerate over all subsets of variables up to size k, which will require $O(n^k)$ evaluations of Equation 2; we require a worst-case $O(n^{2+k})$ evaluations to compute the sets for all pairs of nodes. Importantly, the node scores used in Equation 2 are functions of no more than $k+2$ variables.

5.2 k-Certified Delete Operators

In this section, we define the KEEPOPERATOR predicate used in Line 7 of Algorithm 1; we provide the pseudocode as Algorithm 4. This predicate automatically keeps those delete operators that can be scored using a normal order-k deletion score. In addition, it also keeps those delete operators that, assuming that the separators $\mathbf{M}^k$ are consistent with the independencies in the generative structure, provably result in an IMAP of the generative distribution; we call these additional operators *k-certified delete operators* because they can be "certified" using the independence facts implied by the order-k separators.

For the remainder of this section, we define what it means to be a k-certified delete operator.

Algorithm 4: KEEPOPERATOR($\mathcal{C}, X, Y, \mathbf{H}, k$)

Input : CPDAG $\mathcal{C}$, Operator $Delete(X, Y, \mathbf{H})$, Bound k
Output: TRUE to keep the operator
$\overline{\mathbf{H}} \longleftarrow \mathbf{NA}^{\mathcal{C}}_{Y,X} \setminus \mathbf{H}$
if $|\{\mathbf{Pa}^{\mathcal{C}}_Y \cup \overline{\mathbf{H}}\} \setminus X| \leq k$ OR $Delete(X, Y, \mathbf{H})$ is k-certified in $\mathcal{C}$ **then return** TRUE
else return FALSE

5.2.1 Maximum Parent Bound

Given a DAG model $\mathcal{H}$ that is an independence map of the generative structure $\mathcal{G}$, we can use the structure of $\mathcal{H}$ to bound the number of parents for any node in $\mathcal{G}$, as we show in the following proposition. We defer the proof to the supplement.

Proposition 1 *Let $\mathcal{G}$ and $\mathcal{H}$ be two DAGs with $\mathcal{G} \leq \mathcal{H}$. Let Y be any node that has k parents in $\mathcal{G}$. Then some node in $\{Y\} \cup \mathbf{Ch}^{\mathcal{H}}_Y$ has at least k parents in $\mathcal{H}$.*

Proposition 1 motivates the following upper bound $B^{\mathcal{H}}(Y)$ on the number of parents of Y in the generative structure $\mathcal{G}$:

$$B^{\mathcal{H}}(Y) = \max_{N \in \{Y\} \cup \mathbf{Ch}^{\mathcal{H}}_Y} |\mathbf{Pa}^{\mathcal{H}}_N|$$

Note that Proposition 1 applies to DAG models, whereas we are using equivalence classes of DAG models in SE-GES. We can leverage Proposition 1 for our implementation of SE-GES given the following lemma:

Lemma 1 *If $\mathcal{H} \approx \mathcal{H}'$, then for every node Y, $B^{\mathcal{H}}(Y) = B^{\mathcal{H}'}(Y)$.*

From Lemma 1, we see that the bound $B^{\mathcal{H}}(Y)$ is the same for every DAG in an equivalence class, and thus we can compute the upper bound for every node by extracting an arbitrary member of the equivalence class and counting parent sets. As a result, for any CPDAG $\mathcal{C}$, we will use $B^{\mathcal{C}}(Y)$ to denote this upper bound for all the DAGs contained in $\mathcal{C}$.

5.2.2 k-Certified Children

We can use the parent bound to guarantee in some situations—and in the large-sample limit—that a node C *must* be a common child of two nodes X and Y. To this end, we have the following definition:

Definition 1 *C is a k-*certified common child of X and Y *in $\mathcal{C}$ if the following conditions hold: (1) $|\mathbf{M}^k_{XY}| \leq k$, (2) $C \notin \mathbf{M}^k_{XY}$, (3) $|\mathbf{M}^k_{XC}| > k$, (4) $|\mathbf{M}^k_{YC}| > k$, and (5) $B^{\mathcal{C}}(C) \leq k$*

To understand this definition, it is useful to think of the d-separation constraints that must hold in any generative DAG model $\mathcal{G}$. Condition (1) implies that X and Y are not adjacent. Given the parent bound (5), conditions (3) and (4) imply that C must be adjacent to X and Y, respectively. Adding condition (2) implies that C cannot be a parent of either X or Y. More formally, we have the following result.

Theorem 2 *If the separator sets $\mathbf{M}^k$ are consistent with the independencies in a distribution that is perfect with respect to $\mathcal{G}$, then any k-certified common child of X and Y is a common child of X and Y in $\mathcal{G}$.*

5.2.3 Main Result

With the definition of the max-parent bound and k-certified children, we can now define a k-certified delete operator.

Definition 2 *A delete operator $Delete(X, Y, \mathbf{H})$ is a k-*certified delete operator in $\mathcal{C}$ *if the following conditions hold:*

1. $|\mathbf{M}^k_{XY}| \leq k$
2. *Every node in $\mathbf{C} = \{\mathbf{CC}^{\mathcal{C}}_{X,Y} \cup \mathbf{H}\}$ is a k-certified common child of X and Y in $\mathcal{C}$*
3. *With $\overline{\mathbf{H}} = \mathbf{NA}^{\mathcal{C}}_{Y,X} \setminus \mathbf{H}$, for every node $E \in \{\mathbf{Pa}^{\mathcal{C}}_Y \cup \overline{\mathbf{H}}\} \setminus \mathbf{M}^k_{XY}$ either*
   - *(a) $\mathbf{M}^k_{EX} \leq k$ and every semi-directed path from $\mathbf{C}$ to $\mathbf{M}^k_{EX}$ passes through a node in $\overline{\mathbf{H}} \cup X \cup Y$*
   - *(b) $\mathbf{M}^k_{EY} \leq k$ and every semi-directed path from $\mathbf{C}$ to $\mathbf{M}^k_{EY}$ passes through a node in $\overline{\mathbf{H}} \cup X \cup Y$*

The following two theorems codify the significance of the above definition.

Theorem 3 *If the separator sets $\mathbf{M}^k$ are consistent with the independencies in a distribution that is perfect with respect to $\mathcal{G}$, then applying any k-certified delete operator to $\mathcal{C}$ results in an IMAP of $\mathcal{G}$.*

Theorem 4 *If the separator sets $\mathbf{M}^k$ are consistent with the independencies in a distribution that is perfect with respect to $\mathcal{G}$, where each node in $\mathcal{G}$ has at most k parents, then for any CPDAG $\mathcal{C}$with $\mathcal{G} < \mathcal{C}$, there exists a k-certified delete operator in $\mathcal{C}$.*

Combining these two results with the observation that algorithm SELECTIVE-GEN-OPS only eliminates delete operators that already fail requirement (2) in the definition of k-certification, we see that in the large-sample limit, if we define the predicate KEEPOPERATOR to return TRUE precisely for the k-certified delete operators, SE-BES will reach the equivalence class of the generative distribution by repeatedly applying k-certified delete operators.

We defer the proofs of Theorem 3 and Theorem 4 to the supplement, but now explain the intuition behind the specific conditions in Definition 2, under the assumptions (1) we can use the separating sets $\mathbf{M}^k$ as an order-k independence oracle and (2) the current CPDAG $\mathcal{C}$ is an IMAP of the generative model $\mathcal{G}$. Condition 1 of Definition 2 simply tests that X and Y are not adjacent in $\mathcal{G}$.

We show in the supplement that if X and Y are not adjacent in $\mathcal{G}$, then applying $Delete(X, Y, \mathbf{H})$ to $\mathcal{C}$ results in an IMAP of $\mathcal{G}$ if no non-$\mathbf{M}^k_{XY}$ "extra" node E from the conditioning set $\mathbf{S}$ of the delete (i.e., $\mathbf{S} = \{\mathbf{Pa}^{\mathcal{C}}_Y \cup \overline{\mathbf{H}}\} \setminus X$) is contained in $\mathbf{D}^{\mathcal{G}}_{X,Y}$. Condition 3 of Definition 2 provides a test that can rule out $E \in \mathbf{D}^{\mathcal{G}}_{X,Y}$ that requires only order-k separators, but the test is only correct in the case where the post-delete common children of X and Y in the CPDAG are common children of X and Y in $\mathcal{G}$; condition 2 of Definition 2 guarantees this property.

What allows condition 3 to work is the following technical result: if $E \in \mathbf{D}^{\mathcal{G}}_{X,Y}$, then E cannot be separated from either X or Y without conditioning on some node in $\mathbf{D}^{\mathcal{H}}_{X,Y}$, where $\mathcal{H}$ is any DAG member of the post-delete equivalence class. Furthermore, we can identify the nodes in $\mathbf{D}^{\mathcal{H}}_{X,Y}$ precisely as those nodes reachable by a semi-directed path that does not reach any node in $\overline{\mathbf{H}} \cup X \cup Y$.

The existence result of Theorem 4 leverages both (1) the result of Chickering and Meek (2015) that we can always find an edge to delete where the induced subgraph "below" the edge is correct[3], and (2) we can always identify the necessary order-k separators in Definition 2 given a parent bound k from $\mathcal{G}$.

5.3 Scoring k-Certified Operators

We now consider how to score high-order operators that are k certified; namely, how we implement Line 5 in Algorithm 3 for those operators of order greater than k.

If we interpret the minimal separating sets as outputs from an independence oracle, then the k-certified operators can be understood as an unordered set of candidate deletions that are all "correct" in the sense that they result in an IMAP of the generative distribution. Under this interpretation, we can apply *any* of the candidate deletions and will end up with the generative structure once this algorithm returns no operators. We see that in the large sample limit, this approach will guide SE-BES out of the "dense" region of the search space using an independence oracle in much the same way that the PC algorithm works; the main difference being that the SE-BES delete operators necessarily result in non-empty equivalence classes.

But there are a number of advantages to using score-based search algorithms. Perhaps the most important is that in many real-world applications, we have finite data and a score that we have designed for the purpose of maximization. By discretizing the score to "independent" and "not independent", we lose the granularity of the score and end up solving a constraint-satisfaction problem instead of a maximization problem.

To leverage the scoring function, we can score delete operators using the separator set $\mathbf{M}_{XY}$ directly: $Sc(Y, \mathbf{M}_{XY}) - Sc(Y, X \cup \mathbf{M}_{XY})$. By definition, this score will be greater than zero, but it allows us to prioritize deletions based on the magnitude of the "independence score"; this score can be understood as an approximate lower bound under the large-sample convergence of the scoring function to the Bayesian Information Criterion (Schwarz, 1978).

6 EXPERIMENTS

In this section, we present the results of a synthetic experiment that demonstrates that a particular implementation of SE-GES—which starts its search from the complete model—can identify the structure of the generative distribution more often than GES, and that this benefit increases with the complexity of the generative structure. We conducted our experiment using a small number of variables with the goal of demonstrating both that (1) GES can fail to reach sparse generative structures due to the need to explore dense regions of the search space, and (2) SE-GES can successfully traverse *out of* dense regions of the search space during the backward phase by leveraging its low-order scoring function. Importantly, we are not endorsing the complete-model variant of SE-GES for large domains, but rather hope to understand the behavior of the backward phase of the algorithm in the case when a (more practical) variant of FES reaches states with similarly-sized clusters of nodes of increasingly dense dependence structure.

For our experiment, we generated gold-standard net-

[3]Corollary 3 of Chickering and Meek (2015).

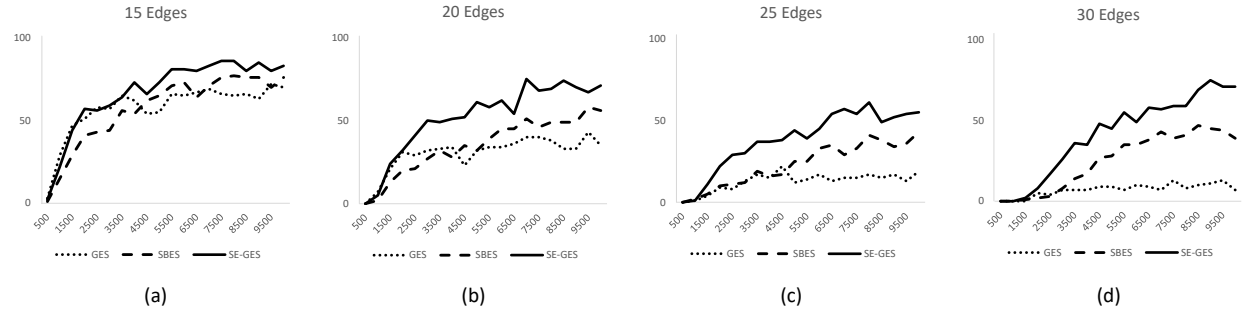

Figure 2: Percent of times the generative structure was identified for each algorithm as a function of the sample size for generative structures containing (a) 15 edges, (b) 20 edges, (c) 25 edges and (d) 30 edges.

works randomly, sampled data from those networks, and then ran competing structure-search algorithms using the sampled data. We evaluated the algorithms based on their ability to recover exactly the structure (i.e., the equivalence class) of the gold-standard network. Every gold-standard network contained 12 nodes, and every node had 3 discrete states. We generated random gold-standard structures with an increasing number of edges, but with the constraint that each node had at most 3 parents.

To implement the cap on the number of parents, we started with a random "maximally dense" network structure that was created as follows. First, we took a random permutation of the nodes, and for each node in order, we added a random 3 parents from the predecessors in the permutation. Thus, every node has exactly 3 parents except for the first three in the permutation (which have 0, 1 and 2 parents, respectively).

Given a maximally dense network and a desired number of edges for a gold-standard network, we simply randomly deleted edges until we had the given number of edges remaining. To specify the distribution for the gold-standard network, we sampled the parameters of each conditional distribution from a uniform Dirichlet distribution.

In our experiment, we varied the edge count in the gold-standard networks from 15 to 30 in steps of 5, and for each edge count, we varied the data size from 500 to 10000 in steps of 500. For each edge count and data size, we generated 100 random gold-standard networks, sampled the given number of rows from that network, and then ran the structure-search algorithms. For each algorithm, we recorded the percent of times that algorithm identified the generative structure.

We compared SE-GES to both GES and SGES. We used the Bayesian Information Criterion to score search states for all algorithms; this criterion satisfies the required properties described in Section 4. For both SE-GES and SGES, we used the "complete model" implementation of FES, so that both of these algorithms started with the no-missing-edges graph. We stopped both SE-GES and SGES after we hit the (known) parent limit of 3. For each parent limit, when SE-GES reached a local minimum, we continued from that point on with SGES. For both SE-GES and SGES, we kept track of the final model reached after each parent limit, and ran GES from the *best* one after stopping.

In Figure 2, we show the percent of times—out of the 100 random instances—that the final model reached by each algorithm was equivalent to the gold-standard network. As expected, for each of the 4 levels of gold-standard complexity, we see that all of the algorithms perform better with higher sample sizes. The figure demonstrates that GES has an increasingly difficult time identifying the gold-standard network as the complexity of that network increases, and that of the three algorithms, SE-GES is the most robust to increasing complexity of the gold-standard model. In Figure 3, we show the average time for each algorithm to complete as a function of the number of rows, for the gold-standard network consisting of 30 edges.

7 CONCLUSION

We have provided theoretical building blocks for a class of greedy structure-search algorithms that require only low-order scores while retaining the large-sample guarantees of GES. The benefits of SE-GES (and SGES) are manifest when both (1) the generative distribution is sparse and (2) GES needs to reach a dense IMAP during forward search. In our experiments, we tried to simulate both of these conditions by creating generative distributions with increasing number of edges while maintaining

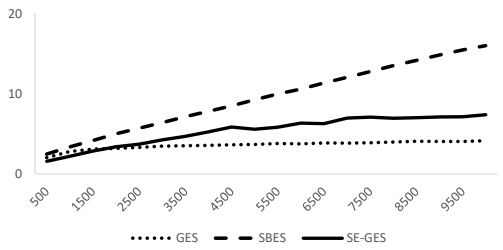

Figure 3: Average time in seconds to run each algorithm as a function of the sample size for the generative structures containing 30 edges.

a maximum-parent constraint.

In real-world scenarios, there remain many open questions about how to best leverage the low-order scoring of SE-GES. We expect, for example, that there should be much better ways to run FES than starting from the complete graph; we believe there is opportunity in using heuristic forward-search algorithms that use low-order scores when adding edges.

There are also many alternatives to scoring the k-certified delete operators. In this paper and in the experiments, we considered using the separator sets as if they were the parent sets in a "normal" delete operator. But an operator is k-certified only if a number of independence/dependence facts hold simultaneously. In particular, we require two dependencies to hold for each k-certified common child, and we require at least one independence to hold for each "extra" parent. If the scoring function only has weak evidence for any one of these facts, we might want to lower the priority the corresponding operator.

Acknowledgments

We thank Chris Meek for the valuable discussions on this work.

References

[1] David Maxwell Chickering. A transformational characterization of Bayesian network structures. In S. Hanks and P. Besnard, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence,* Montreal, QC, pages 87–98. Morgan Kaufmann, August 1995.

[2] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, November 2002.

[3] David Maxwell Chickering and Christopher Meek. Selective greedy equivalence search: Finding optimal Bayesian networks using a polynomial number of score evaluations. In *Proceedings of the Thirty First Conference on Uncertainty in Artificial Intelligence,* Amsterdam, Netherlands, 2015.

[4] Nir Friedman, Iftach Nachman, and Dana Peer. Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence,* Stockholm, Sweden. Morgan Kaufmann, 1999.

[5] Maxime Gasse, Alex Aussem, and Haytham Elghazel. A hybrid algorithm for Bayesian network structure learning with application to multi-label learning. *Expert Systems with Applications*, 2014.

[6] Markus Kalisch and Peter Buhlmann. Estimating high-dimensional directed acyclic graphs with the PC algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.

[7] Preetam Nandy, Alain Hauser, and Marloes H. Maathuis. High-dimensional consistency in score-based and hybrid structure learning. *Annals of Statistics*, 46(6A):3151–3183, 2018.

[8] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[9] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search (second edition)*. The MIT Press, Cambridge, Massachussets, 2000.

[10] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 2006.

[11] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In M. Henrion, R. Shachter, L. Kanal, and J. Lemmer, editors, *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, 1991.

[12] Or Zuk, Shiri Margel, and Eytan Domany. On the number of samples needed to learn the correct structure of a Bayesian network. In *Proceedings of the Twenty Second Conference on Uncertainty in Artificial Intelligence,* Canbridge, MA, 2006.

Supplement to: Statistically Efficient Greedy Equivalence Search

Abstract

This supplement contains the proofs for the main results of our paper: Theorem 2, Theorem 3 and Theorem 4.

8 INTRODUCTION

In this supplement, we provide the proofs to the results from our paper *Statistically Efficient Greedy Equivalence Search*. The supplement is organized as follows. In Section 9, we present additional notation and definitions needed for the proofs. In Section 10, we provide a number of preliminary results regarding active paths in DAG models, most of which have been proved elsewhere. In Section 11, we prove Proposition 1 and Lemma 1 from the main paper. In Section 12, we prove Theorem 2 from the main paper. In Section 13, we prove results about the implications that active paths in a DAG model have on any IMAP of that DAG model. In Section 14, we explore statistically efficient deletions in DAG models. In Section 15, we show how to extend many of the DAG-model results from Section 14 to CPDAGs. Finally, in Section 16, we prove Theorem 3 and Theorem 4 from the main paper.

For the entirety of this supplement, we assume that all statistical models are defined over the same set of variables. To simplify the presentation of all the results for DAGs below, unless $\mathcal{G}$ and $\mathcal{H}$ are defined explicitly in the statement of a result, the results all assume that $\mathcal{G}$ and $\mathcal{H}$ are DAG models and that $\mathcal{G} \leq \mathcal{H}$.

9 ADDITIONAL NOTATION

A *node sequence* is an ordered sequence $(N_1, \ldots, N_k)$ of nodes in a DAG. We typically use the symbol π to denote a node sequence; we use $\pi(N_1, N_k)$ to represent a node sequence when we want to emphasize that the first and last points are N_1 and N_k, respectively. A node sequence is a *path* in $\mathcal{G}$ if every pair of consecutive nodes is connected by an edge in $\mathcal{G}$. Note that our definition of a path is not a function of the direction of the edges. The *edges of a path in* $\mathcal{G}$ are the ordered intermediate edges in $\mathcal{G}$. Often the graph is clear from context, and we refer simply to *the edges of* π. We often represent paths in a graph by specifying the edges in the path, as in $\pi = X \to Y \to Z$; this is shorthand for specifying the node sequence $\pi = (X, Y, Z)$ and the intermediate edges $X \to Y$ and $Y \to Z$ from $\mathcal{G}$. A path π is a *directed path in* $\mathcal{G}$ if all of the edges of π in $\mathcal{G}$ along π are directed in the same way.

A *fragment* of a node sequence π is a **contiguous** subset of the node sequence. For example, if $\pi = (A, B, C, D)$, then (A, B, C) and (B, C) are both fragments of π. If π is a path in $\mathcal{G}$, we call the path corresponding to a fragment of π a *path fragment* in $\mathcal{G}$. For example, for path $\pi = A \to B \to C \to D$ in some graph $\mathcal{G}$, the paths $A \to B \to C$ and $B \to C$ are both path fragments of π in $\mathcal{G}$.

When a path has converging arrows at some node W (i.e., the path contains $\to W \leftarrow$), we call the node W a *collider* along the path; we call all other nodes in the path—including the endpoints—the *non-colliders*. For a path π in $\mathcal{G}$, we use $\mathbf{INC}_{\pi}^{\mathcal{G}}$ to represent the *intermediate* non-colliders in that path. We use $\mathbb{E}(\pi)$ to denote the set containing the two endpoints of π.

A *path detour in* $\mathcal{G}$ of a node sequence $\pi(X, Y)$ is a path between X and Y in $\mathcal{G}$ that contains a (not necessarily strict) subsequence of the node sequence. Unlike a path fragment, the sequence of nodes in the subset need not be contiguous. For example, suppose there is a path π in $\mathcal{G}$. A path detour of π in $\mathcal{H}$ is a path that traverses a subset of the nodes in π, in the same order, but may skip over some of the intermediate nodes.

A path $\pi(X, Y)$ is a $\mathbf{S}$-*active path in* $\mathcal{G}$ if neither X nor Y belongs to $\mathbf{S}$, and for every intermediate node W along the path, either (1) W is a collider and $W \in \mathbf{S}$ or (2) W is not a collider and $W \notin \mathbf{S}$ [1].

[1] An equivalent (and perhaps more prevalent) definition of an active path states that for condition (1), either W or a descendant of W in $\mathcal{G}$ is in $\mathbf{S}$. For those readers familiar with the cel-

A $\mathbf{S}$-active path $\pi(X, Y)$ in $\mathcal{G}$ is *compact* if its endpoints X and Y do not appear as intermediate points along $\pi(X, Y)$, and no node $S \in \mathbf{S}$ occurs more than once.

The independence constraints implied by a DAG structure are characterized by the *d-separation* criterion. Two nodes A and B are said to be d-separated in a DAG $\mathcal{G}$ given a set of nodes $\mathbf{S}$ if and only if there is no active path in $\mathcal{G}$ between A and B given $\mathbf{S}$. We sometimes use the notation $\mathbf{X} \perp\!\!\!\perp_{\mathcal{G}} \mathbf{Y} | \mathbf{S}$ to denote that in $\mathcal{G}$, all nodes in $\mathbf{X}$ are d-separated from all nodes in $\mathbf{Y}$ given $\mathbf{S}$; conversely, we use $\mathbf{X} \not\perp\!\!\!\perp_{\mathcal{G}} \mathbf{Y} | \mathbf{S}$ to denote the fact that there exists a $\mathbf{S}$-active path between some node in $\mathbf{X}$ and some node in $\mathbf{Y}$.

The direction of each *terminal* edge in an active path—that is, the first and last edge encountered in a traversal from one end of the path to the other—is important for determining whether we can concatenate two active paths together to make a third active path. We say that a path $\pi(A, B)$ is *into* A if the terminal edge incident to A is oriented toward A (i.e., $A \leftarrow$). Similarly, the path is into B if the terminal edge incident to B is oriented toward B. If a path is not into an endpoint A, we say that the path is *out of* A.

We say a node X is $\mathcal{G}$*-lowest* from set $\mathbf{X}$ if no directed path exists in $\mathcal{G}$ from X to some other node in $\mathbf{X}$.

For any PDAG $\mathcal{P}$ and subset of nodes $\mathbf{V}$, we use $\mathcal{P}[\mathbf{V}]$ to denote the subgraph of $\mathcal{P}$ induced by $\mathbf{V}$; that is, $\mathcal{P}[\mathbf{V}]$ has as nodes the set $\mathbf{V}$ and has as edges all those from $\mathcal{P}$ that connect nodes in $\mathbf{V}$.

A set of nodes $\mathbf{T}$ *holds a tree in* $\mathcal{G}$ if there is some node $R \in \mathbf{T}$ for which there is a directed path in $\mathcal{G}$ from R to every other node in $\mathbf{T}$, where each such directed path passes entirely through the nodes in $\mathbf{T}$. In other words, the induced subgraph $\mathcal{G}[\mathbf{T}]$ contains a directed tree in its edges among all nodes in $\mathbf{T}$. We call the top-most node of the set $\mathbf{T}$ the *root* of the tree

Given DAGs $\mathcal{G}$ and $\mathcal{H}$ for which $\mathcal{G} < \mathcal{H}$, we say that an edge e from $\mathcal{H}$ is *deletable in* $\mathcal{H}$ *with respect to* $\mathcal{G}$ if, for the DAG $\mathcal{H}'$ that results after removing e from $\mathcal{H}$, we have $\mathcal{G} \leq \mathcal{H}'$. We will say that an edge is *deletable in* $\mathcal{H}$ or simply *deletable* if $\mathcal{G}$ or both DAGs, respectively, are clear from context.

10 PRELIMINARY DAG RESULTS

In this section, we include a number of preliminary results regarding Bayesian networks, many of which have been proven elsewhere.

Lemma 8. (Chickering, 2002) *Let $\pi(A, B)$ be a $\mathbf{S}$-active path between A and B, and let $\pi(B, C)$ be a $\mathbf{S}$-active path between B and C. If either path is out of B, then the concatenation of $\pi(A, B)$ and $\pi(B, C)$ is a $\mathbf{S}$-active path between A and C.*

Proposition 8. *Any $\emptyset$-active path $\pi(X, Y)$ in $\mathcal{G}$ that is out of X is a directed path from X to Y in $\mathcal{G}$.*

Proof: Because π is out of X, we know the first edge in π is directed away from X. Because the path is $\emptyset$-active, it cannot contain a collider, and thus all subsequent edges in the path must also be directed away from X. □

Proposition 9. *If X and Y are not adjacent to each other in $\mathcal{G}$, then $X \perp\!\!\!\perp_{\mathcal{G}} Y | \mathbf{Pa}_X^{\mathcal{G}} \cup \mathbf{Pa}_Y^{\mathcal{G}}$.*

Proposition 10. *(**Chickering, 2002**) If there is an edge between X and Y in $\mathcal{G}$, then there is an edge between X and Y in $\mathcal{H}$.*

From Proposition 10, the following result is immediate:

Proposition 11. *If π is a path in $\mathcal{G}$ then π is a path in $\mathcal{H}$.*

Lemma 9. *(**Chickering, 2002**) If $\mathcal{G}$ contains the collider $X \rightarrow Z \leftarrow Y$, then either $\mathcal{H}$ contains the same collider or X and Y are adjacent in $\mathcal{H}$.*

The following two results follows immediately from the definition of an active path.

Proposition 12. *Every $\emptyset$-active path $\pi(X, Y)$ in $\mathcal{G}$ either (1) is a directed path from one endpoint to the other or (2) contains an intermediate node R and is the concatenation of a directed path from R to X and a directed path from R to Y.*

Proposition 13. *If π is a $\mathbf{S}$-active path in $\mathcal{G}$, then every path fragment π' is $\mathbf{S} \setminus \mathbb{E}(\pi')$-active in $\mathcal{G}$.*

Note that in the statement of Proposition 13, the endpoints of the fragment are explicitly excluded from the conditioning set in order to satisfy this requirement of an active path.

Proposition 14. *If π is a $\mathbf{S}$-active path in $\mathcal{G}$, then for any $\{N_i, N_j\} \subseteq \mathbf{INC}_\pi^{\mathcal{G}} \cup \mathbb{E}(\pi)$, there is a $\mathbf{S}$-active fragment $\pi'(N_i, N_j)$ in $\mathcal{G}$.*

Proof: Every node in $\mathbf{INC}_\pi^{\mathcal{G}} \cup \mathbb{E}(\pi)$ is reached by π and consequently a fragment between every pair in the set must exist. Because none of the nodes in $\mathbf{INC}_\pi^{\mathcal{G}} \cup \mathbb{E}(\pi)$ belong to $\mathbf{S}$, the result follows from Proposition 13. □

Lemma 10. (Chickering and Meek, 2015) *An edge $X \rightarrow Y$ is deletable in $\mathcal{H}$ with respect to $\mathcal{G}$ if and only if $Y \perp\!\!\!\perp_{\mathcal{G}} X | \mathbf{Pa}_Y^{\mathcal{H}} \setminus X$.*

The final results in this section regard holding a tree in a DAG model.

Proposition 15. *If $\mathcal{G}$ contains a directed path π, then the set of nodes in the path holds a tree in $\mathcal{G}$.*

Proposition 16. *$\mathbf{T}$ holds a tree in $\mathcal{G}$ if and only if $\mathcal{G}[\mathbf{T}]$ has a $\emptyset$-active path between every pair of nodes.*

ebrated "Bayes ball" algorithm of Shachter (1998) for testing d-separation, the definition in this paper is simply a valid path that the ball can take between two nodes.

Proof: Assume $\mathcal{G}[\mathbf{T}]$ has a $\emptyset$-active path between every pair of nodes, and let $R \in \mathbf{T}$ be any node in that induced subgraph that has no parents. By assumption, there is a $\emptyset$-active path between R and all other nodes in $\mathbf{T}$. Because R has no parents, every such path must be out of R, and we conclude from Proposition 8 that there is a directed path from R to all other nodes in $\mathbf{T}$, and thus $\mathbf{T}$ holds a tree in $\mathcal{G}[\mathbf{T}]$; because every edge in $\mathcal{G}[\mathbf{T}]$ exists in $\mathcal{G}$, $\mathbf{T}$ must hold a tree in $\mathcal{G}$ as well.

Assume $\mathbf{T}$ holds a tree in $\mathcal{G}$. By definition, $\mathbf{T}$ must hold a tree in $\mathcal{G}[\mathbf{T}]$ as well. Let R denote the root of $\mathbf{T}$. Consider any pair $\{T_i, T_j\} \in \mathbf{T}$. If $T_i = R$ or $T_j = R$, the directed path from R to the other node constitutes an $\emptyset$-active path in $\mathcal{G}[\mathbf{A}]$; otherwise, from Lemma 8, we can concatenate (1) the directed path from R to T_i and (2) the directed path from R to T_j to create an $\emptyset$-active path in $\mathcal{G}[\mathbf{T}]$. □

Lemma 11. *If $\mathbf{T}$ holds a tree in $\mathcal{G}$, then $\mathbf{T}$ holds a tree in $\mathcal{H}$.*

Proof: From Proposition 16, we know that for every pair of nodes $\{T_i, T_j\} \subseteq \mathbf{T}$, there is a $\emptyset$-active path between T_i and T_j in $\mathcal{G}$ that passes entirely through the set $\mathbf{T}$. From Lemma 13, we know that for every such active path in $\mathcal{G}$, there is a $\emptyset$-active detour between T_i and T_j in $\mathcal{H}$, and thus by Proposition 16 the lemma follows. □

11 PROOF OF PROPOSITION 1 AND LEMMA 1

In this section, we prove Proposition 1 and Lemma 1 from the main paper.

Proposition 1 *Let $\mathcal{G}$ and $\mathcal{H}$ be two DAGs with $\mathcal{G} \leq \mathcal{H}$. Let Y be any node that has k parents in $\mathcal{G}$. Then some node in $\{Y\} \cup \mathbf{Ch}_Y^{\mathcal{H}}$ has at least k parents in $\mathcal{H}$.*

Proof: If $\mathbf{Pa}_Y^{\mathcal{H}} \supseteq \mathbf{Pa}_Y^{\mathcal{G}}$ then the proposition follows immediately. Otherwise, consider the $\mathcal{H}$-lowest $L \in \mathbf{Pa}_Y^{\mathcal{G}} \backslash \mathbf{Pa}_Y^{\mathcal{H}}$. From Proposition 10 we know L must be a child of Y in $\mathcal{H}$ and from Lemma 9 we know L must be adjacent to every other member of $\mathbf{Pa}_Y^{\mathcal{G}}$ in $\mathcal{H}$; because L is lowest, its parents in $\mathcal{H}$ must contain Y and every other element of $\mathbf{Pa}_Y^{\mathcal{G}}$, which means $|\mathbf{Pa}_L^{\mathcal{H}}| \geq k$. □

For any DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, we say an edge $X \rightarrow Y \in \mathbf{E}$ is *covered* in $\mathcal{G}$ if X and Y have identical parents, with the exception that X is not a parent of itself. That is, $X \rightarrow Y$ is covered in $\mathcal{G}$ if $\mathbf{Pa}_Y^{\mathcal{G}} = \mathbf{Pa}_X^{\mathcal{G}} \cup X$. The significance of covered edges is evident from the following result:

Lemma 12. (Chickering, 1995) *Let $\mathcal{G}$ be any DAG model, and let $\mathcal{G}'$ be the result of reversing the edge $X \rightarrow Y$ in $\mathcal{G}$. Then $\mathcal{G}'$ is a DAG that is equivalent to $\mathcal{G}$ if and only if $X \rightarrow Y$ is covered in $\mathcal{G}$.*

Lemma 1 *If $\mathcal{H} \approx \mathcal{H}'$, then for every node Y, $B^{\mathcal{H}}(Y) = B^{\mathcal{H}'}(Y)$.*

Proof: From Lemma 12, we can establish the result by showing that it holds for any covered edge reversal. From the definition of a covered edge, if we reverse a covered edge $A \rightarrow B$, then the number of parents of A after the reversal will be equal to the number of parents of B before the reversal, and the number of parents of B after the reversal will be equal to the number of parents of A before the reversal. In other words, A and B "swap" the number of parents. Given this, we see that any covered edge reversal that does not change the elements in the set $\{Y\} \cup \mathbf{Ch}_Y^{\mathcal{H}}$ cannot change the bound $B^{\mathcal{H}}(Y)$ either. Consequently, we must have $B = Y$ and A is some parent of Y. But after the reversal, A will be a child of Y and have the same number of parents that Y did before the reversal, and thus the lemma follows. □

12 PROOF OF THEOREM 2

In this section, we prove Theorem 2 from the main paper, which we re-state here:

Theorem 2 *If the separator sets $\mathbf{M}^k$ are consistent with the independencies in a distribution that is perfect with respect to $\mathcal{G}$, then any k-certified common child of X and Y is a common child of X and Y in $\mathcal{G}$.*

Proof: Let C be any k-certified common child. First we show that C cannot be an ancestor of either X or Y in $\mathcal{G}$. To simplify notation, let $\mathbf{T} = \mathbf{M}_{XY}^k$.

Because $|\mathbf{M}_{XC}^k| > k$ and $|\mathbf{M}_{YC}^k| > k$ (properties 3 and 4 in the definition of a k-certified common child), we know that there must be a $\mathbf{T}$-active path both between X and C and between Y and C. These two paths must meet at C as a collider, else their concatenation would constitute a $\mathbf{T}$-active path between X and Y. Suppose we are wrong, and that C is an ancestor of X (or Y). Then if the directed path from C to X (Y) is not blocked by a node in $\mathbf{T}$, it would constitute a $\mathbf{T}$-active path from C to X (Y) that is out-of C, and thus could be concatenated to the $\mathbf{T}$-active path from Y (X) to C to create a $\mathbf{T}$-active path between X and Y. But if the directed path from C to X (Y) *is* blocked by a node in $\mathbf{T}$, then we can use the *highest* such $T \in \mathbf{T}$ to identify a $\mathbf{T}$-active segment

$$C \rightarrow \ldots \rightarrow T \leftarrow \ldots \leftarrow C$$

between C and itself—that is out-of C on both ends—that we can insert between the $X - C$ path and the $Y - C$ path to once again identify a $\mathbf{T}$-active path between X and Y.

Given that C is not an ancestor of X or Y—or equivalently, neither X or Y is a descendant of C—we know that either C is a common child of X and Y, or C is independent of at least one of them given $\mathbf{Pa}_C^{\mathcal{G}}$; we know $|\mathbf{Pa}_C^{\mathcal{G}}| \leq k$ from property 5, and thus properties 3 and 4 rule out either independence, and we conclude C must be a common child of X and Y. □

13 CONSTRAINTS BETWEEN ACTIVE PATHS IN $\mathcal{G}$ AND THE STRUCTURE OF $\mathcal{H}$

In this section, we show how the existence of active paths in a DAG model $\mathcal{G}$ constrain the structure of any IMAP $\mathcal{H}$. For the majority of these results, we posit the existence of a $\{\mathbf{Pa}_Y^{\mathcal{H}} \setminus X\}$-active path in $\mathcal{G}$ between X and Y—which means that any edge $X \to Y$ in $\mathcal{H}$ would not be deletable per Lemma 10—and derive consequences about $\mathcal{H}$.

Lemma 13. *For any $\mathbf{S}$-active path π in $\mathcal{G}$, there is a $\mathbf{S}$-active detour of π in $\mathcal{H}$.*

Proof: Assume $\mathbb{E}(\pi) = \{X, Y\}$. We prove the lemma by induction on the number of nodes k from $\mathbf{INC}_\pi^{\mathcal{G}}$.

For the basis, we consider $k = 0$, and thus the path edges of π in $\mathcal{G}$ either (1) consists of a single edge between X and Y, in which case the lemma follows by Proposition 10, or (2) consists of a collider $X \to S \leftarrow Y$ for some $S \in \mathbf{S}$, in which case the lemma follows by Lemma 9.

For the induction step, assuming $k > 0$, we know there must be some intermediate node along the path that does not belong to $\mathbf{S}$; let H be the *highest* such node in $\mathcal{G}$. From Proposition 13, we know that there is a $\mathbf{S}$-active path in $\mathcal{G}$ both between X and H and between Y and H; from the induction hypothesis, both of these paths have a $\mathbf{S}$-active detour in $\mathcal{H}$. If either of these two detours are out-of H, we know from Lemma 8 that we can concatenate them together to construct the desired detour between X and Y in $\mathcal{H}$. Otherwise, these two paths meet as a collider:

$$X - \ldots - L \to H \leftarrow R - \ldots - Y$$

Because both paths from X and Y are $\mathbf{S}$ active in $\mathcal{H}$, we know that neither of the two in-coming nodes L and R can be in $\mathbf{S}$. Both of these nodes are higher than H, yielding a contradiction. □

Lemma 14. *For any $\mathbf{Pa}_Y^{\mathcal{H}}$-active path π between Y and some other node in $\mathcal{G}$, every node in $\mathbf{INC}_\pi^{\mathcal{G}}$ is a descendant of Y in $\mathcal{H}$.*

Proof: $\mathcal{H}$ asserts that Y is independent of every non-descendant given its parents, and because $\mathcal{G} \leq \mathcal{H}$, these independences must also hold in $\mathcal{G}$. Because in $\mathcal{G}$, there is a $\mathbf{Pa}_Y^{\mathcal{H}}$-active path between Y and every node in $\mathbf{INC}_\pi^{\mathcal{G}}$ the lemma follows. □

Proposition 17. *Let $X \to Y$ be any edge in $\mathcal{H}$, and let $\mathbf{S} = \mathbf{Pa}_Y^{\mathcal{H}} \setminus X$ be the parents of Y excluding X. For any compact $\mathbf{S}$-active path π between X and Y in $\mathcal{G}$, every node in $\mathbf{INC}_\pi^{\mathcal{G}}$ is a descendant of both X and Y in $\mathcal{H}$.*

Proof: From Proposition 14, we know that for every $N \in \mathbf{INC}_\pi^{\mathcal{G}}$, there is a $\mathbf{S}$-active fragment $\pi(Y, N)$ of π in $\mathcal{G}$. Because π is compact, we know that X does not occur anywhere along the fragment, and thus $\pi(Y, N)$ is $\mathbf{Pa}_Y^{\mathcal{H}}$ active in $\mathcal{G}$ as well. Thus from Lemma 14 we conclude that N must be a descendant of Y in $\mathcal{H}$. Because $\mathcal{H}$ contains the edge $X \to Y$, every descendant of Y is also a descendant of X. □

Lemma 15. *Let $X \to Y$ be any edge in $\mathcal{H}$, and let $\mathbf{S} = \mathbf{Pa}_Y^{\mathcal{H}} \setminus X$ be the parents of Y excluding X. If there exists a compact $\mathbf{S}$-active path π between X and Y in $\mathcal{G}$, then for every $\mathbf{S}$-active path fragment π' of π in $\mathcal{G}$, there exists a $\emptyset$-active detour of π' in $\mathcal{H}$.*

Proof: Suppose the lemma is wrong. From Lemma 13, we know there exists a $\mathbf{S}$-active detour π'' of π' in $\mathcal{H}$. If π'' is not $\emptyset$ active, then it must contain some segment $L \to S \leftarrow R$, where $S \in \mathbf{S}$. Because π'' is a detour of a fragment of π, it contains a subset of the nodes in π, and thus because neither endpoint of π'' can belong to $\mathbf{S}$, at least one of $\{L, R\}$ must belong to $\mathbf{INC}_\pi^{\mathcal{H}}$. But from Proposition 17 all of these nodes are descendants of Y in $\mathcal{H}$, and thus because S is a parent of Y in $\mathcal{H}$, $\mathcal{H}$ must contain a cycle. □

Lemma 16. *Let $X \to Y$ be any edge in $\mathcal{H}$, and let $\mathbf{S} = \mathbf{Pa}_Y^{\mathcal{H}} \setminus \{X\}$ be the parents of Y excluding X. For any compact $\mathbf{S}$-active path π between X and Y in $\mathcal{G}$, if $\mathbf{INC}_\pi^{\mathcal{G}} \neq \emptyset$, then the $\mathcal{H}$-highest element of $\mathbf{INC}_\pi^{\mathcal{G}}$ is a common child of X and Y in $\mathcal{H}$.*

Proof: Let N_H be the $\mathcal{H}$-highest element of $\mathbf{INC}_\pi^{\mathcal{G}}$. From Lemma 15, $\mathcal{H}$ must contain a $\emptyset$-active fragment of π between both (1) X and N_H and (2) Y and N_H. From Proposition 17, we know both X and Y are ancestors of every node in $\mathbf{INC}_\pi^{\mathcal{G}}$, and because N_H is the $\mathcal{H}$-highest element of $\mathbf{INC}_\pi^{\mathcal{G}}$, N_H is an ancestor of every *other* node in $\mathbf{INC}_\pi^{\mathcal{G}}$. This means that neither of the $\emptyset$-active fragments can contain any node from $\mathbf{INC}_\pi^{\mathcal{G}}$ other than N_H, lest the fragment would contain a collider, and thus these two fragments must consist of a single edge. Because both X and Y are ancestors of N_H, the lemma follows. □

Lemma 17. *Let $\pi(X, Y)$ be any directed path in $\mathcal{G}$. If $\mathcal{G} \leq \mathcal{H}$, then either (1) every node in $\pi(X, Y)$ is a descendant of X in $\mathcal{H}$, or (2) there exists a node in $\pi(X, Y)$ that is a parent of X in $\mathcal{H}$.*

Proof: Induction on the length of the path. If the path is of length one, we know from Proposition 10 that X and Y must be adjacent in $\mathcal{H}$, in which case either X is a parent of Y—and hence every node in the path is a descendant of X in $\mathcal{H}$—or Y is a parent of X in $\mathcal{H}$.

Suppose the lemma is correct for all paths of length $k - 1$, and consider a length-k path $\pi(X, Y)$ in $\mathcal{G}$, and let Z be the next-to-last node in the path. Because there is a directed path of length $k - 1$ from X to Z in $\mathcal{G}$, we know by the induction hypothesis that either (1) all nodes in the fragment $\pi(X, Z)$ are descendants of X in $\mathcal{H}$ or (2) there exists a node in the fragment $\pi(X, Z)$ that is a parent of X in $\mathcal{H}$. Because $\pi(X, Z)$ is a fragment of $\pi(X, Y)$, if

(2) holds for $\pi(X, Z)$ then (2) holds for $\pi(X, Y)$ and the lemma follows. For the remainder of the proof, we assume that all nodes in $\pi(X, Z)$ are descendants of X in $\mathcal{H}$.

From Lemma 13, we know that there must be a $\emptyset$-active detour π' of $\pi(X, Y)$ in $\mathcal{H}$. From Proposition 12, we know that π' in $\mathcal{H}$ must either be a directed path from X to Y—in which case the lemma follows because Y is a descendant of X—or there is some node R other than X along the path for which π' is the concatenation of a directed path from R to X and a directed path from R to Y. Because all intermediate nodes of $\pi(X, Y)$ are descendants of X in $\mathcal{H}$, R cannot be any intermediate node, else $\mathcal{H}$ would contain a cycle. Thus we conclude that $R = Y$ and the directed path contains no intermediate nodes, and hence Y is a parent of X in $\mathcal{H}$. □

Lemma 18. *Let $X \to Y$ be any edge in $\mathcal{H}$, and let $C \in \mathbf{CC}^{\mathcal{G}}_{X,Y} \cap \mathbf{CC}^{\mathcal{H}}_{X,Y}$ be any common child of X and Y in both $\mathcal{G}$ and $\mathcal{H}$. Let π be any directed path in $\mathcal{G}$ that starts at C. If π reaches any node that does not belong to $\mathbf{D}_{\mathcal{H}}(X, Y)$, then the first such node is an element of $\mathbf{Pa}^{\mathcal{H}}_{Y}$.*

Proof: Because $C \in \mathbf{D}_{\mathcal{H}}(X, Y)$, we know π starts with a fragment $\pi_F(C, L)$ containing at least one edge where all nodes except for the last one, L, are in $\mathbf{D}_{\mathcal{H}}(X, Y)$. Let $\mathbf{T}$ be the set of nodes reached by this segment.

From Proposition 15, we know that $\mathbf{T}$ holds a tree in $\mathcal{G}$ (with root C), and from Lemma 11 $\mathbf{T}$ must also hold a tree in $\mathcal{H}$. L must be the root of the tree in $\mathcal{H}$, else by virtue of L being a descendant of a node in $\mathbf{D}_{\mathcal{H}}(X, Y)$, it would also have to belong to this set.

Because C is a common child of X and Y in $\mathcal{G}$, we can prepend the edge $X \to C$ or $Y \to C$ to π to create a longer directed path, and thus from Proposition 15 we know that both $\mathbf{T} \cup X$ and $\mathbf{T} \cup Y$ hold a tree in $\mathcal{G}$, and from Lemma 11 they both hold a tree in $\mathcal{H}$ as well.

First we consider the set $\mathbf{T} \cup Y$. Because all nodes in $\mathbf{T}$ except for L are descendants of Y in $\mathcal{H}$, we know that the root of the $\mathbf{T} \cup Y$ tree in $\mathcal{H}$ must be either Y or L. If L is the root, it must be a parent of Y because, again, all other nodes are descendants of Y, and the lemma follows.

We complete the proof by showing that it is impossible for Y to be the root of the $\mathbf{T} \cup Y$ tree in $\mathcal{H}$.

Assume we are wrong, and that Y is the root of the $\mathbf{T} \cup Y$ tree in $\mathcal{H}$. Because L is the root of the $\mathbf{T}$ tree in $\mathcal{H}$, it must be the case that Y is a parent of L in $\mathcal{H}$.

Now we consider the set $\mathbf{T} \cup X$. As in the case of Y, the root of this tree in $\mathcal{H}$ must either be X or L. Because $X \to Y$ is in $\mathcal{H}$ and we just argued that $Y \to L$ is in $\mathcal{H}$, L cannot be the root of the tree, lest $\mathcal{H}$ contains a cycle. Because L is the root of the $\mathbf{T}$ tree in $\mathcal{H}$, it must be the case that X is a parent of L in $\mathcal{H}$. But this means L is a common child of X and Y in $\mathcal{H}$ and thus belongs to $\mathbf{D}_{\mathcal{H}}(X, Y)$, yielding a contradiction, and we conclude that indeed Y cannot be the root of the $\mathbf{T} \cup Y$ tree. □

Proposition 18. *If $X \perp\!\!\!\perp_{\mathcal{G}} Y | \mathbf{T}$, then $\mathbf{T} \cap \mathbf{D}_{\mathcal{G}}(X, Y) = \emptyset$.*

Proof: Suppose there is some highest $T \in \mathbf{T}$ that is a descendant of a common child C of X and Y. We can construct a $\mathbf{T}$-active path between X and Y that consists of the two directed paths from X and Y to T through their common child:

$$X \to C \to \ldots \to T \leftarrow \ldots \leftarrow C \leftarrow Y$$

Any intermediate node that blocks any portion of this path would constitute a higher node in $\mathbf{T}$. □

14 STATISTICALLY EFFICIENT DELETION IN DAG MODELS

In this section, we show how to use low-order independence facts to prove deletability in DAG models.

Theorem 8. *Let $X \to Y$ be any edge in $\mathcal{H}$ for which $X \perp\!\!\!\perp_{\mathcal{G}} Y | \mathbf{T}$ and $\mathbf{CC}^{\mathcal{H}}_{X,Y} \subseteq \mathbf{CC}^{\mathcal{G}}_{X,Y}$. Then $X \to Y$ is deletable in $\mathcal{H}$ if for every $E \in \mathbf{Pa}^{\mathcal{H}}_{Y} \setminus \mathbf{T}$, there exists a set $\mathbf{Q}$ for which (1) $\mathbf{Q} \cap \mathbf{D}_{\mathcal{H}}(X, Y) = \emptyset$ and (2) either $E \perp\!\!\!\perp_{\mathcal{G}} X | \mathbf{Q}$ or $E \perp\!\!\!\perp_{\mathcal{G}} Y | \mathbf{Q}$ (or both).*

Proof: Let $\mathbf{S} = \mathbf{Pa}^{\mathcal{H}}_{Y} \setminus \{X\}$. If the lemma is wrong, then $X \to Y$ is not deletable in $\mathcal{H}$, and thus from Lemma 10 there must exist a $\mathbf{S}$-active path π between X and Y in $\mathcal{G}$.

We first argue that $\mathbf{INC}^{\mathcal{G}}_{\pi}$ must contain at least one element. If this set were empty, then either π consists of a single edge between X and Y or π consists of a single collider $X \to S \leftarrow Y$ for some $S \in \mathbf{S}$. The first case is ruled out by the existence of $\mathbf{T}$ that renders X and Y independent. For the second case, we know $S \notin \mathbf{T}$, or else we would have a $\mathbf{T}$-active path between X and Y in $\mathcal{G}$; but this means that $S \in \mathbf{Pa}^{\mathcal{H}}_{Y} \setminus \mathbf{T}$ (i.e., it is one of the "E" nodes), and given that X and Y are both parents of S, no set $\mathbf{Q}$ can exist that renders it independent from X or Y.

Given that $\mathbf{INC}^{\mathcal{G}}_{\pi}$ is non-empty, we conclude from Lemma 16 that π must reach some node $C \in \mathbf{CC}^{\mathcal{H}}_{X,Y}$, and from the conditions of the theorem C is in $\mathbf{CC}^{\mathcal{G}}_{X,Y}$ as well. Because C is not a collider along π in $\mathcal{G}$, it cannot be in the conditioning set, and thus both of the path fragments $\pi(C, X)$ and $\pi(C, Y)$ are $\mathbf{S}$ active in $\mathcal{G}$, and at least one of them must be out-of C in $\mathcal{G}$. Without loss of generality, assume the $\mathbf{S}$-active fragment $\pi(C, X)$ is out-of C in $\mathcal{G}$.

From the definition of an active path, the (out-of) fragment $\pi(C, X)$ must consist of a directed path from C to either X or some node $S \in \mathbf{S}$. If the directed path reaches X, we know from Proposition 18 that it cannot be blocked by any node in $\mathbf{T}$ (any such node would belong to $\mathbf{D}_{\mathcal{G}}(X, Y)$ by virtue of being a descendant of $C \in \mathbf{CC}^{\mathcal{G}}_{X,Y}$), and thus we

could prepend this path with the edge $Y \to C$ to identify a $\mathbf{T}$-active path between X and Y, a contradiction.

We conclude that the fragment $\pi(C, X)$ must start with a directed path starting with C and reach some node $S \in \mathbf{S}$. Because S is a parent of Y in $\mathcal{H}$, we know $S \notin \mathbf{D}_{\mathcal{H}}(X, Y)$, and thus from Lemma 18, the directed path must hit some *first* such S. If we prepend this directed path with $X \to C$ or $Y \to C$, we have identified a directed path between both X and Y to S for which all the intermediate nodes are in $\mathbf{D}_{\mathcal{H}}(X, Y)$. This contradicts the existence of a $\mathbf{Q}$ with $\mathbf{Q} \cap \mathbf{D}_{\mathcal{H}}(X, Y) = \emptyset$ for which either $S \perp\!\!\!\perp_{\mathcal{G}} X | \mathbf{Q}$ or $S \perp\!\!\!\perp_{\mathcal{G}} Y | \mathbf{Q}$, and we conclude that no $\mathbf{S}$-active path between X and Y can exist. □

Corollary 8. (Chickering and Meek, 2015) *If there exists a deletable edge in $\mathcal{H}$, then there exists a deletable edge $X \to Y$ in $\mathcal{H}$ for which $\mathbf{D}_{\mathcal{H}}(X, Y) = \mathbf{D}_{\mathcal{G}}(X, Y)$.*

Recall that in the main paper we use $\mathbf{M}^k_{XY}$ to denote a minimal separating set of size at most k between X and Y in $\mathcal{G}$, and that if no such set exists, we say that this set is *undefined*. For the remainder of this section, take k to be the number of variables in the domain and use $\mathbf{M}_{XY}$ to represent separating sets of any size.

Proposition 19. *$\mathbf{M}_{XY}$ is undefined if and only if X and Y are adjacent in $\mathcal{G}$.*

Proof: If X and Y are adjacent, no set can separate them. If they are not adjacent, they are either d-separated by the parents of Y or by the parents of X (or both). □

Lemma 19. *Every node in $\mathbf{M}_{XY}$ is an ancestor of either X or Y (or both) in $\mathcal{G}$.*

Proof: Suppose the lemma is wrong. Then $\mathbf{M}_{XY}$ must be defined and contain at least one node that is not an ancestor of either X or Y; let L be any $\mathcal{G}$-lowest such node, and let $\mathbf{R} = \mathbf{M}_{XY} \setminus \{L\}$ be the remaining nodes from the minimal separating set. Because $\mathbf{M}_{XY}$ is minimal, there must be some $\mathbf{R}$-active path between X and Y that reaches L as an intermediate node. One of the subpaths between L and either X or Y must be out-of L, or else L would be a collider along the path. Without loss of generality, assume that the sub-path between L and Y is out-of L. This sub-path cannot be $\emptyset$-active, or else the path is a directed path from L to Y. But if the sub-path is *not* $\emptyset$-active, there must be a directed path from L to some node $R \in \mathbf{R}$. R cannot be an ancestor of X or Y, or else L would be also, but if R is not an ancestor of either X or Y, then L is not the lowest node with this property, yielding a contradiction. □

Proposition 20. *If in $\mathcal{G}$, nodes X and Y are not adjacent and both have no more than k parents, then $\mathbf{M}_{XY} \leq k$.*

Proof: Follows because, given that X and Y are not adjacent in $\mathcal{G}$ we have either $X \perp\!\!\!\perp_{\mathcal{G}} Y | \mathbf{Pa}^{\mathcal{G}}_X$ or $X \perp\!\!\!\perp_{\mathcal{G}} Y | \mathbf{Pa}^{\mathcal{G}}_Y$ (or both), and thus given the parent bound there exists a separating set of size $\leq k$. □

Theorem 9. *If there exists a deletable edge in $\mathcal{H}$, then there exists a deletable edge $X \to Y$ in $\mathcal{H}$ for which (1) $\mathbf{CC}^{\mathcal{H}}_{X,Y} \subseteq \mathbf{CC}^{\mathcal{G}}_{X,Y}$ and (2) for every $E \in \mathbf{Pa}^{\mathcal{H}}_Y \setminus \mathbf{M}_{XY}$, either (a) $\mathbf{M}_{EX}$ is defined and $\mathbf{M}_{EX} \cap \mathbf{D}_{\mathcal{H}}(X, Y) = \emptyset$ or (b) $\mathbf{M}_{EY}$ is defined and $\mathbf{M}_{EY} \cap \mathbf{D}_{\mathcal{H}}(X, Y) = \emptyset$.*

Proof: From Corollary 8, we know there exists a deletable edge $X \to Y$ for which $\mathbf{D}_{\mathcal{H}}(X, Y) = \mathbf{D}_{\mathcal{G}}(X, Y)$; for the remainder of the proof we assume $X \to Y$ has this property.

Because $\mathbf{CC}^{\mathcal{G}}_{X,Y} \subseteq \mathbf{D}_{\mathcal{G}}(X, Y)$ and $\mathbf{CC}^{\mathcal{H}}_{X,Y} \subseteq \mathbf{D}_{\mathcal{H}}(X, Y)$, property (1) follows immediately.

Consider any $E \in \mathbf{Pa}^{\mathcal{H}}_Y \setminus \mathbf{M}_{XY}$. E cannot be adjacent to both X and Y in $\mathcal{G}$, else either (1) it would be a common child, in which case $X \to Y$ would not be deletable, or (2) it would have to belong to $\mathbf{M}_{XY}$ to ensure $X \perp\!\!\!\perp_{\mathcal{G}} Y | \mathbf{M}_{XY}$. Without loss of generality, assume E is not adjacent to X in $\mathcal{G}$, and let Q be any element of $\mathbf{M}_{EX}$. We establish the theorem by showing that $Q \notin \mathbf{D}_{\mathcal{H}}(X, Y)$.

From Lemma 19, we know that Q must be an ancestor of either E or X (or both) in $\mathcal{G}$. If Q is an ancestor of X in $\mathcal{G}$, it cannot be in $\mathbf{D}_{\mathcal{H}}(X, Y)$ because, given that $\mathbf{D}_{\mathcal{H}}(X, Y) = \mathbf{D}_{\mathcal{G}}(X, Y)$, that would imply that Q is also a descendant of X in $\mathcal{G}$, which implies $\mathcal{G}$ contains a cycle. If Q is an ancestor of E in $\mathcal{G}$, it cannot be in $\mathbf{D}_{\mathcal{H}}(X, Y)$ because, given that $\mathbf{D}_{\mathcal{H}}(X, Y) = \mathbf{D}_{\mathcal{G}}(X, Y)$, that would imply that E is also in $\mathbf{D}_{\mathcal{H}}(X, Y)$, which implies—because E is a parent of Y in $\mathcal{H}$—that $\mathcal{H}$ contains a cycle. □

We now define the DAG equivalent of a k-certified delete operator. In particular, we define a k-deletable edge in a DAG $\mathcal{H}$—which is completely parallel to the CPDAG variant—as follows.

Definition 8. *An edge $X \to Y$ is a* k-deletable edge *in $\mathcal{H}$ if the following conditions hold:*

1. $|\mathbf{M}_{XY}| \leq k$
2. *Every node in $\mathbf{CC}^{\mathcal{H}}_{X,Y}$ is a k-verifiable common child of X and Y*
3. *For every $E \in \mathbf{Pa}^{\mathcal{H}}_Y \setminus \mathbf{M}_{XY}$, either (a) $|\mathbf{M}_{EX}| \leq k$ and $\mathbf{M}_{EX} \cap \mathbf{D}_{\mathcal{H}}(X, Y) = \emptyset$ or (b) $|\mathbf{M}_{EY}| \leq k$ and $\mathbf{M}_{EY} \cap \mathbf{D}_{\mathcal{H}}(X, Y) = \emptyset$.*

The following is the DAG variant of Theorem 3 from the main paper.

Theorem 10. *Any k-deletable edge in $\mathcal{H}$ is deletable in $\mathcal{H}$.*

Proof: Let $X \to Y$ be any k-deletable edge, and let $\mathbf{S} = \mathbf{Pa}^{\mathcal{H}}_Y \setminus \{X\}$. If the lemma is wrong, then $X \to Y$ is not deletable in $\mathcal{H}$, and thus there must exist a $\mathbf{S}$-active path π between X and Y in $\mathcal{G}$.

From property 1 from Definition 8, we know there exists a set $\mathbf{T} = \mathbf{M}_{XY}$ that d-separates X and Y in $\mathcal{G}$.

We first argue that $\mathbf{INC}_{\pi}^{\mathcal{G}}$ must contain at least one element. If this set were empty, then either π consists of a single edge between X and Y or π consists of a single collider $X \rightarrow S \leftarrow Y$ for some $S \in \mathbf{S}$. The first case is ruled out by the existence of set $\mathbf{T}$ that renders X and Y independent. For the second case, we know $S \notin \mathbf{T}$, or else we would have a $\mathbf{T}$-active path between X and Y in $\mathcal{G}$; but this means that $S \in \mathbf{Pa}_Y^{\mathcal{H}} \setminus \mathbf{T}$ (i.e., it is one of the "E" nodes), and given that X and Y are both parents of S, no set $\mathbf{Q}$ can exist that renders it independent from X or Y.

Given that $\mathbf{INC}_{\pi}^{\mathcal{G}}$ is non-empty, we conclude from Lemma 16 that π must reach some node $C \in \mathbf{CC}_{X,Y}^{\mathcal{H}}$, and because all such nodes are k-verifiable common children, we know from Theorem 2 that C is in $\mathbf{CC}_{X,Y}^{\mathcal{G}}$ as well. Because C is not a collider along π in $\mathcal{G}$, it cannot be in the conditioning set, and thus both of the path fragments $\pi(C, X)$ and $\pi(C, Y)$ are $\mathbf{S}$ active in $\mathcal{G}$, and at least one of them must be out-of C in $\mathcal{G}$. Without loss of generality, assume the $\mathbf{S}$-active fragment $\pi(C, X)$ is out-of C in $\mathcal{G}$.

From the definition of an active path, the (out-of) fragment $\pi(C, X)$ must consist of a directed path from C to either X or some node $S \in \mathbf{S}$. If the directed path reaches X, we know from Proposition 18 that it cannot be blocked by any node in $\mathbf{T}$ (any such node would belong to $\mathbf{D}_{\mathcal{G}}(X, Y)$ by virtue of being a descendant of $C \in \mathbf{CC}_{X,Y}^{\mathcal{G}}$), and thus we could prepend this path with the edge $Y \rightarrow C$ to identify a $\mathbf{T}$-active path between X and Y, a contradiction.

We conclude that the fragment $\pi(C, X)$ must start with a directed path starting with C and reach some node $E \in \mathbf{S} \setminus \mathbf{T}$. Because E is a parent of Y in $\mathcal{H}$, we know $E \notin \mathbf{D}_{\mathcal{H}}(X, Y)$, and thus from Lemma 18, the directed path must hit some *first* such E. If we prepend this directed path with $X \rightarrow C$ or $Y \rightarrow C$, we have identified a directed path between both X and Y to E for which all the intermediate nodes are in $\mathbf{D}_{\mathcal{H}}(X, Y)$. This contradicts the third property of a k-deletable edge. □

The following is the DAG variant of Theorem 4 from the main paper.

Theorem 11. *If each node in $\mathcal{G}$ has at most k parents, and if there exists a deletable edge in $\mathcal{H}$, then there exists a k-deletable edge in $\mathcal{H}$.*

Proof: From Theorem 9, we know there exists a j-deletable edge in $\mathcal{H}$, where j is the maximum size of a defined separating set in the statement of that theorem. From Proposition 20, and the fact that no node in $\mathcal{G}$ has more than k parents, we conclude $j \leq k$. □

15 COMPLETED PDAG RESULTS

In this section, we show that the results about deletability in DAG models can be extended to deletion operators in CPDAGs.

A described in detail by Chickering (2002), when we use equivalence classes of DAG models as search states, the search operators are defined and scored by edge additions and deletions to the DAGs that belong to the equivalence class; we gain efficiency in the representation by grouping together all DAGs that result in the same equivalence class.

More formally, for an equivalence class $\mathcal{C}$ and deletion operator $O = Delete(X, Y, \mathbf{H})$, we say a DAG model $\mathcal{H}$ is a *representative consistent extension for O in $\mathcal{C}$* if (1) $\mathcal{H}$ is a consistent extension of $\mathcal{C}$ and (2) deleting $X \rightarrow Y$ from $\mathcal{H}$ results in a DAG model that is a consistent extension of the equivalence class resulting from applying O to $\mathcal{C}$. We use $\mathcal{R}(\mathcal{C}, O)$ to denote the set of all representative consistent extensions for O in $\mathcal{C}$.

Chickering (2002) shows that the set of DAG models in $\mathcal{R}(\mathcal{C}, O)$ is characterized by the presence of the edge $X \rightarrow Y$ and the set $\mathbf{H}$ defined from the operator O. In particular, we have the following lemma[2].

Lemma 20. (Chickering 2002) *Let $\mathcal{C}$ be any CPDAG, let $\mathcal{H}$ be any consistent extension of $\mathcal{C}$ and let $O = Delete(X, Y, \mathbf{H})$ be any valid edge-deletion operator. Then $\mathcal{H} \in \mathcal{R}(\mathcal{C}, O)$ if any only if $\mathcal{H}$ contains the edge $X \rightarrow Y$ and $\mathbf{CC}_{X,Y}^{\mathcal{H}} = \mathbf{CC}_{X,Y}^{\mathcal{C}} \cup \mathbf{H}$.*

Corollary 9. *Let $\mathcal{C}$ be any completed PDAG, let $O = Delete(X, Y, \mathbf{H})$ be any valid edge-deletion operator, and let $\overline{\mathbf{H}} = \mathbf{NA}_{Y,X} \setminus \mathbf{H}$. Then for every $\mathcal{H} \in \mathcal{R}(\mathcal{C}, O)$, $\overline{\mathbf{H}} \subseteq \mathbf{Pa}_Y^{\mathcal{H}}$.*

Proof: Suppose not, and let Z be any node in $\overline{\mathbf{H}}$ that is not a parent of Y in $\mathcal{H}$. Because Z is in $\mathbf{NA}_{Y,X}$ it must be a child of Y in $\mathcal{H}$, and because $X \rightarrow Y$ is in $\mathcal{H}$, the edge between X and Z must be directed as $X \rightarrow Z$, lest $\mathcal{H}$ contains a cycle. But this means Z is a common child of X and Y in $\mathcal{H}$, which from Lemma 20 means $Z \in \mathbf{H}$, a contradiction. □

Lemma 21. *Let $\mathcal{H}$ be any DAG containing a deletable edge $X \rightarrow Y$, and let $\mathcal{C}$ be the equivalence class containing $\mathcal{H}$. Then for $\mathbf{H} = \mathbf{CC}_{X,Y}^{\mathcal{H}} \setminus \mathbf{CC}_{X,Y}^{\mathcal{C}}$, the operator $Delete(X, Y, \mathbf{H})$ is a valid edge-deletion operator for $\mathcal{C}$.*

Proof: From the preconditions of a valid delete operator, we need to show (1) $\mathbf{H} \subseteq \mathbf{NA}_{Y,X}$ and (2) $\mathbf{NA}_{Y,X} \setminus \mathbf{H}$ is a clique.

Property (1) follows by the definition of $\mathbf{H}$ by the following argument. Because all nodes in this set are common children of X and Y in $\mathcal{H}$, they must all be adjacent to both X and Y in $\mathcal{C}$. We have explicitly excluded any $\mathcal{C}$ children of

[2]The lemma is stated slightly differently in Chickering (2002). Instead of defining the set of common children of X and Y in $\mathcal{H}$ explicitly as we do here, the lemma in Chickering (2002) only specifies that the common children in $\mathcal{H}$ that are connected to Y by reversible edges must be $\mathbf{H}$; the common children that are connected to Y by compelled edges are precisely the common children in $\mathcal{C}$.

Y, and the set cannot contain any $\mathcal{C}$ parents or Y lest these would have been parents (i.e., not children) of Y in $\mathcal{H}$; thus they must all be neighbors of Y that are adjacent to X in $\mathcal{C}$.

For Property (2), assume the lemma is wrong and there exists two nodes A and B from $\mathbf{NA}_{Y,X} \setminus \mathbf{H}$ that are not adjacent. In $\mathcal{H}$, we know that A and B cannot both be parents of Y, lest $A \rightarrow Y \leftarrow B$ would be a v-structure in $\mathcal{H}$ and thus the edges would be directed in $\mathcal{C}$, contradicting the fact that they are both neighbors in $\mathcal{C}$. Without loss of generality, assume that A is a child of Y in $\mathcal{H}$. Because A is adjacent to X and $\mathcal{H}$ contains the edge $X \rightarrow Y$, A must be a common child of X and Y in $\mathcal{H}$, lest $\mathcal{H}$ contains a cycle. But because $A \notin \mathbf{CC}^{\mathcal{C}}_{X,Y}$, this means it must be in $\mathbf{H}$, a contradiction. □

Lemma 22. (Chickering, 2002a) *If $X-Y$ is an undirected edge in a completed PDAG $\mathcal{C}$, then $\mathbf{Pa}^{\mathcal{C}}_X = \mathbf{Pa}^{\mathcal{C}}_Y$.*

Proposition 21. *Let $\mathcal{C}$ be any completed PDAG. If $\mathcal{C}$ contains a semi-directed path π from X to Y, then $\mathcal{C}$ contains a detour of π consisting of (1) an undirected path of length zero or more edges from X to Z, and (2) a directed path of zero or more edges from Z to Y.*

Proof: As we traverse the edges of π from X to Y, if we ever reach a segment $A \rightarrow B - C$, we know from Lemma 22 that A is a parent of C, and we can replace the segment with $A \rightarrow C$. Repeating this replacement until we reach Y establishes the result. □

Lemma 23. (Chickering, 1995) *Let $\{X, Y, Z\}$ be any three nodes that form a clique of size three in PDAG $\mathcal{P}$. If any two of the edges in the clique are undirected, then the third edge is undirected as well.*

For a completed PDAG $\mathcal{C}$, we use $SD^{\mathcal{C}}(\mathbf{C}, \mathbf{B})$ to denote the set of all nodes reachable in $\mathcal{C}$ from a node in $\mathbf{C}$ via a semi-directed path that does not pass through any node in set $\mathbf{B}$ of "blocking" nodes.

Theorem 12. *Let $\mathcal{C}$ be any CPDAG, let $O = Delete(X, Y, \mathbf{H})$ be any valid edge-deletion operator, let $\overline{\mathbf{H}} = \mathbf{NA}_{Y,X} \setminus \mathbf{H}$, let $\mathbf{C} = \mathbf{CC}^{\mathcal{C}}_{X,Y} \cup \mathbf{H}$, and let $\mathbf{B} = \overline{\mathbf{H}} \cup X \cup Y$. Then for every $\mathcal{H} \in \mathcal{R}(\mathcal{C}, O)$, $\mathbf{D}_{\mathcal{H}}(X, Y) = SD^{\mathcal{C}}(\mathbf{C}, \mathbf{B})$.*

Proof: We break the proof into two parts. For Part I, we show that $SD^{\mathcal{C}}(\mathbf{C}, \mathbf{B}) \subseteq \mathbf{D}_{\mathcal{H}}(X, Y)$, and for Part II, we show $\mathbf{D}_{\mathcal{H}}(X, Y) \subseteq SD^{\mathcal{C}}(\mathbf{C}, \mathbf{B})$.

Part I: We demonstrate that $SD^{\mathcal{C}}(\mathbf{C}, \mathbf{B}) \subseteq \mathbf{D}_{\mathcal{H}}(X, Y)$ by considering only those nodes in $SD^{\mathcal{C}}(\mathbf{C}, \mathbf{B})$ that are reachable by an *undirected* path; once this is established, Part I of the lemma follows because, from Proposition 21, any node reachable by a semi-directed path can be reached by a detour of that path that first traverses undirected edges only, then follows directed edges that must also exist in $\mathcal{H}$.

Let $\mathbf{S}$ be the nodes in $SD^{\mathcal{C}}(\mathbf{C}, \mathbf{B})$ that are reachable by some undirected path in $\mathcal{C}$ that does not pass through any node in $\overline{\mathbf{H}} \cup X \cup Y$, and let S be any node in $\mathbf{S}$. We prove Part I by showing $S \in \mathbf{D}_{\mathcal{H}}(X, Y)$ using an induction on the length of the *shortest* undirected path from some $C \in \mathbf{C}$ to S in CPDAG $\mathcal{C}$.

For the basis, we consider paths both of length zero and of length one. For length zero, $S \in \mathbf{CC}^{\mathcal{C}}_{X,Y} \cup \mathbf{H}$ and thus by Lemma 20 $S \in \mathbf{CC}^{\mathcal{H}}_{X,Y} \subseteq \mathbf{D}_{\mathcal{H}}(X, Y)$.

For length one, we consider the undirected edge $C - S$ in $\mathcal{C}$. If the edge is directed as $C \rightarrow S$ in $\mathcal{H}$, the lemma follows immediately. Otherwise, the edge must be directed as $C \leftarrow S$ in $\mathcal{H}$, in which case S must be adjacent to both X and Y in both $\mathcal{C}$ and $\mathcal{H}$, lest $\mathcal{H}$ contains a v-structure not in $\mathcal{C}$. If S is a child of Y in $\mathcal{H}$, then because we know (from Lemma 20) that $\mathcal{H}$ contains the edge $X \rightarrow Y$, S must also be a child of X (lest $\mathcal{H}$ contains a cycle) and thus $S \in \mathbf{CC}^{\mathcal{H}}_{X,Y} \subseteq \mathbf{D}_{\mathcal{H}}(X, Y)$. If S is a parent of Y in $\mathcal{H}$, then we have a directed path $S \rightarrow Y \rightarrow C$ in $\mathcal{H}$. Because the edge between C and S is undirected in $\mathcal{H}$, we know that either (1) both of the edges $S \rightarrow Y$ and $Y \rightarrow C$ are directed in $\mathcal{C}$, which means $C \leftarrow S$ would also have to be directed in $\mathcal{C}$, contradicting the fact that $C-S$ is undirected, or (2) neither of the edges $S \rightarrow Y$ nor $Y \rightarrow C$ are directed in $\mathcal{C}$, which means that $S \in \overline{\mathbf{H}}$, another contradiction. Thus S cannot be a parent of Y and the lemma follows.

For the induction step, assume the lemma holds for any shortest path of length k, and we consider a shortest path of length $k+1$ between C and S. Because $k > 1$, we know there is some previous pair of nodes A and B with $A - B - S$ being the last two edges in the shortest path. Because the path from C to S is shortest, the sub-path from C to B must also be a shortest path, and we conclude from the induction hypothesis that $B \in \mathbf{D}_{\mathcal{H}}(X, Y)$. Thus, if the edge between B and S is directed in $\mathcal{H}$ as $B \rightarrow S$, the lemma follows immediately. For the remainder of the proof, we assume the edge between B and S is directed in $\mathcal{H}$ as $B \leftarrow S$.

We now consider the *first* edge $C - F$ along the shortest path to S. From the proof of the length-one induction hypothesis, we know that either $F \in \mathbf{C}$—in which case the undirected path from C to S is not the shortest path from a node in $\mathbf{C}$ to S—or the edge is directed as $C \rightarrow F$. If the first edge is directed as $C \rightarrow F$ and the last edge is directed as $B \leftarrow S$, then there must be some intermediate collider $L \rightarrow I \leftarrow R$ along the path. Because both edges are undirected in $\mathcal{C}$, we know L and R must be adjacent, and via Lemma 23 the edge $L - R$ is undirected in $\mathcal{C}$, which means we can create a shorter undirected path by replacing the segment $L - I - R$ in the path by $L - R$. With this contradiction, we conclude that Part I of the lemma follows.

Part II: We now demonstrate that $\mathbf{D}_{\mathcal{H}}(X, Y) \subseteq$

$SD^{\mathcal{C}}(X, Y)$. Consider any $S \in \mathbf{D}_{\mathcal{H}}(X, Y)$, and let π denote any directed path from $C \in \mathbf{CC}^{\mathcal{H}}_{X,Y}$ to S in $\mathcal{H}$. Because $\mathcal{H}$ is identical to a consistent extension of $\mathcal{C}$, except that the edge $X \rightarrow Y$ has been deleted, we know that corresponding to the directed path in $\mathcal{H}$ must be a semi-directed path in $\mathcal{C}$. It remains to be shown that this semi-directed path does not pass through any node in $\mathbf{E} = \overline{\mathbf{H}} \cup X \cup Y$; but given Corollary 9 and the fact that both X and Y are ancestors in $\mathcal{H}$ of every node in π, this follows immediately given that $\mathcal{H}$ is acyclic. □

The following two corollaries follow from the equivalence of the definitions of (1) a k-deletable edge in DAG $\mathcal{H}$ and (2) a k-certified delete operator O in $\mathcal{C}$ for which $\mathcal{H} \in \mathcal{R}(\mathcal{C}, O)$. Both of these definitions have three properties: property 1 is identical in the two definitions, property 2 is equivalent as a result of Lemma 20, and property 3 is equivalent from Theorem 12.

Corollary 10. *If delete operator $O = Delete(X, Y, \mathbf{H})$ in $\mathcal{C}$ is k certified, then the edge $X \rightarrow Y$ is k deletable in every $\mathcal{H} \in \mathcal{R}(\mathcal{C}, O)$.*

Corollary 11. *Given a delete operator $O = Delete(X, Y, \mathbf{H})$ in $\mathcal{C}$, if there exists an edge $X \rightarrow Y$ that is k deletable in some $\mathcal{H} \in \mathcal{R}(\mathcal{C}, O)$, then O is k certified.*

16 PROOF OF THEOREM 3 AND THEOREM 4 FROM MAIN PAPER

Theorem 3 *If the separator sets $\mathbf{M}^k$ are consistent with the independencies in a distribution that is perfect with respect to $\mathcal{G}$, then applying any k-certified delete operator to $\mathcal{C}$ results in an IMAP of $\mathcal{G}$.*

Proof: Let $O = Delete(X, Y, \mathbf{H})$ be any k-certified delete operator, and let $\mathcal{H}$ be any element of $\mathcal{R}(\mathcal{C}, O)$. From Corollary 10, we know that $X \rightarrow Y$ is k-deletable in $\mathcal{H}$, and from Theorem 10, $X \rightarrow Y$ is deletable in $\mathcal{H}$. From the definition of a deletable edge, the resulting DAG $\mathcal{H}'$ must be an IMAP of $\mathcal{G}$. From the definition of $\mathcal{R}(\mathcal{C}, O)$, $\mathcal{H}'$ is a member of the CPDAG resulting from applying O to $\mathcal{C}$, and thus the theorem follows. □

Theorem 4 *If the separator sets $\mathbf{M}^k$ are consistent with the independencies in a distribution that is perfect with respect to $\mathcal{G}$, where each node in $\mathcal{G}$ has at most k parents, then for any CPDAG $\mathcal{C}$with $\mathcal{G} < \mathcal{C}$, there exists a k-certified delete operator in $\mathcal{C}$.*

Proof: From Theorem 1, we know that as long as $\mathcal{G} < \mathcal{C}$, there must exist some valid delete operator $O = Delete(X, Y, \mathbf{H})$. This means that for any $\mathcal{H} \in \mathcal{R}(\mathcal{C}, O)$, the edge $X \rightarrow Y$ is deletable in $\mathcal{H}$, and from Theorem 11, we conclude that there must also exist some k-deletable edge $Z \rightarrow W$ in $\mathcal{H}$. From Lemma 21, we know there must be a corresponding valid delete operator $O' = Delete(Z, W, \mathbf{H}')$ for $\mathcal{C}$. From Corollary 11, we conclude that O' is a k-certified delete operator in $\mathcal{C}$. □

References

[1] David Maxwell Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, February 2002.

[2] Ross Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In G. Cooper and S. Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence,* Madison, Wisconsin, pages 480–487. Morgan Kaufmann, August 1998.