# 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 $\infty$. 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).

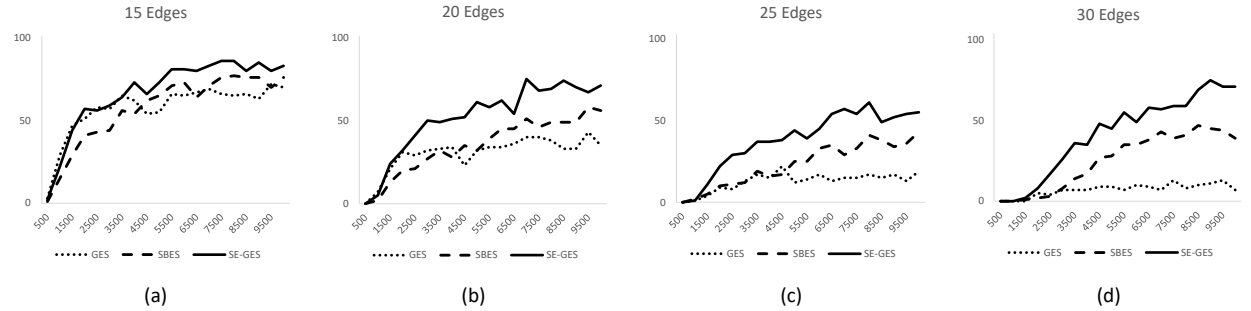


Figure 2: Percent of times the generative structure was identified for each algorithm as a function of the sample size for generative structures containing (a) 15 edges, (b) 20 edges, (c) 25 edges and (d) 30 edges.

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

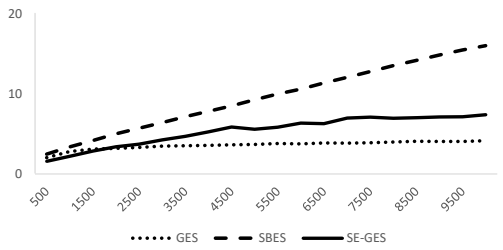


Figure 3: Average time in seconds to run each algorithm as a function of the sample size for the generative structures containing 30 edges.

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.