On the Number of Conditional Independence Tests in Constraint-based Causal Discovery

Marc Franquesa Monés*
CFIS, UPC
Broad Institute

Jiaqi Zhang*
LIDS, MIT
Broad Institute

Caroline Uhler
LIDS, MIT
Broad Institute

Abstract

Learning causal relations from observational data is a fundamental problem with wide-ranging applications across many fields. Constraint-based methods infer the underlying causal structure by performing conditional independence tests. However, existing algorithms such as the prominent `PC` algorithm need to perform a large number of independence tests, which in the worst case is exponential in the maximum degree of the causal graph. Despite extensive research, it remains unclear if there exist algorithms with better complexity without additional assumptions. Here, we establish an algorithm that achieves a better complexity of $p^{\mathcal{O}(s)}$ tests, where p is the number of nodes in the graph and s denotes the maximum undirected clique size of the underlying essential graph. Complementing this result, we prove that any constraint-based algorithm must perform at least $2^{\Omega(s)}$ conditional independence tests, establishing that our proposed algorithm achieves exponent-optimality up to a logarithmic factor in terms of the number of conditional independence tests needed. Finally, we validate our theoretical findings through simulations, on semi-synthetic gene-expression data, and real-world data, demonstrating the efficiency of our algorithm compared to existing methods in terms of number of conditional independence tests needed.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

1 INTRODUCTION

Inferring causal relationships between variables is a fundamental challenge across many scientific domains (Wright, 1921; Haavelmo, 1943; Duncan, 1975). Directed acyclic graphs (DAGs) are a popular framework for representing causal structures, where nodes correspond to random variables and directed edges denote direct causal effects. The objective of causal structure discovery is to identify these edges and their orientations from data on the nodes. With observational data and no additional assumptions, it is known that the underlying causal DAG is only identifiable up to its Markov equivalence class (Verma and Pearl, 1990), which can be represented by a partially directed graph known as the *essential graph*.

Constraint-based causal structure discovery methods use conditional independence (CI) tests to infer the underlying essential graph, the most prominent example being the `PC` algorithm (Spirtes et al., 2001): (1) it begins with a fully connected undirected graph and iteratively removes edges between variables if they are found to be conditionally independent given some conditioning set; (2) edge orientations are inferred based on cases where variables that are conditionally independent become dependent when conditioned on a common child—corresponding to v-structures in the underlying DAG; (3) possibly additional orientations are inferred using a set of logical rules known as the Meek rules (Meek, 1995). Building on this approach, many constraint-based algorithms have been introduced to address more general and diverse settings, e.g., including in the presence of latent variables (Verma and Pearl, 1990; Spirtes et al., 1989, 2001; Colombo et al., 2012; Colombo and Maathuis, 2014; Sondhi and Shojaie, 2019).

Despite significant progress in constraint-based al-

*Equal contribution. M.F.M. contributed to this work while visiting the Institute for Data, Systems, and Society at MIT.

gorithms, it has been observed that their efficiency, both in terms of computational speed and correctness, tend to degrade when scaling to larger or denser graphs (Spirtes et al., 2001; Kalisch and Bühlman, 2007). The main bottleneck of the PC algorithm (and other algorithms of similar nature) arises from the adjacency search (step (1) described above), which requires a large number of CI tests, exponential in the maximum degree of the underlying DAG in the worst case (Spirtes et al., 2001; Claassen et al., 2013). Performing a large number of CI tests not only exacerbates the computational burden but may also impact accuracy, since in the finite sample regime guaranteeing the correctness of each CI test requires strong assumptions (Uhler et al., 2013; Teh et al., 2024; Mazaheri et al., 2025). This motivates the development of causal structure discovery algorithms that minimize the number of CI tests that need to be performed.

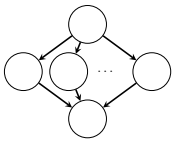


Figure 1: A graph with p nodes, where $d = p - 2$, and $s = 2$.

In this work, we present an algorithm, called *Greedy Ancestral Search* (GAS), that is exponent-optimal up to a logarithmic factor in terms of the number of CI tests performed. GAS requires at most $p^{\mathcal{O}(s)}$ CI tests,[1] where p is the number of nodes in the DAG and s denotes the size of the maximum undirected clique in the underlying essential graph. The best upper bound for previous constraint-based algorithms (Richardson, 1996; Spirtes et al., 2001; Claassen et al., 2013; Colombo and Maathuis, 2014) was $p^{\mathcal{O}(d)}$, where d is the graph's maximum degree. Considering the degree of a node in the maximum undirected clique of the essential graph, note that $s \leq d+1$. In fact, s is often much smaller than d; see, for example, the DAG in Figure 1; thus, $p^{\mathcal{O}(s)}$ is often much smaller than $p^{\mathcal{O}(d)}$. Complementing this upper bound, we prove that any constraint-based algorithm must perform $2^{\Omega(s)}$ CI tests utilizing a technique introduced in Zhang et al. (2024), thereby establishing that GAS achieves exponent-optimality up to a logarithmic factor in terms of the number of CI tests needed. Since d can be of the size $\Omega(p \cdot s)$ (e.g., as in the graph in Figure 1), the previous constraint-based algorithms do not achieve exponent-optimality up to a logarithmic factor. Finally, we empirically benchmark our algorithm against established methods on both synthetic and real-world data, assessing both computational efficiency and accuracy.

To provide some insight on how the reduction in number of CI tests is achieved, consider the PC algorithm. The adjacency search in step (1) of the PC algorithm is where the CI tests are performed. To reduce the number of CI tests compared to the PC algorithm, GAS integrates steps (1) and (2) of the PC algorithm; namely, GAS focuses on using CI tests to learn *ancestral relationships*, which can then be used to perform CI tests to uncover *adjacencies* in a more targeted way. In particular, note that if all ancestral relationships were known, then a single CI test would be sufficient to determine the presence/absence of an edge.[2] As a consequence, the main difficulty of causal structure discovery is to learn the ancestral relations, which motivates our approach of concentrating on CI tests that provide ancestral information. Recent work (Shiragur et al., 2024) has shown that specific ancestral relationships can be learned using a polynomial number of CI tests. We show that, with appropriate modifications, this approach can be extended to learn all the adjacencies with substantially fewer CI tests than other methods.

1.1 Related Works

Learning causal structures from observational data has been extensively studied, with current methods primarily falling into three categories (Kitson et al., 2023): constraint-based approaches (Verma and Pearl, 1990; Spirtes et al., 2001; Kalisch and Bühlman, 2007), which utilize CI tests as constraints to learn the DAG; score-based approaches (Chickering, 2002; Geiger and Heckerman, 2002; Brenner and Sontag, 2013), which search for DAGs that maximize a score function evaluating how well the DAG represents the data; and hybrid approaches (Schulte et al., 2010; Alonso-Barba et al., 2013; Nandy et al., 2018; Solus et al., 2021), which combine both.

For constraint-based algorithms, the focus of this work, a major challenge is their CI test complexity. Algorithms such as PC have a CI test complexity that is

[1] Here, we use the asymptotic notation $\mathcal{O}$ (c.f., (Arora and Barak, 2009)) in the following way: a function $f : \mathcal{G} \mapsto \mathbb{R}$ is $p^{\mathcal{O}(s)}$ if and only if there exists a fixed constant $c \in \mathbb{R}$, such that for any DAG $\mathcal{G}$, it holds that $f(\mathcal{G}) \leq p^{c \cdot s}$, where p and s are the number of nodes and the maximum undirected clique size of $\mathcal{G}$, respectively. We define $2^{\Omega(s)}$ and $p^{\mathcal{O}(d)}$ in a similar way.

[2] All ancestral relationships being known is equivalent to being given a permutation $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_p)$ of the nodes that is *consistent* with the DAG (i.e., if there is a directed edge $\pi_i \to \pi_j \in E$, then it has to hold that $i < j$). In this case a single CI test is sufficient to determine the presence/absence of an edge, namely $\pi_i \to \pi_j$ for $i < j$ if and only if X_{π_i} is not independent of X_{π_j} given X_S where $S = \{\pi_1, \pi_2, \ldots, \pi_{j-1}\} \setminus \{\pi_i\}$.

exponential in the graph's maximum degree (Spirtes et al., 2001; Claassen et al., 2013). While improvements exist that depend on other parameters like the maximum in-degree (Mokhtarian et al., 2025) or the maximum separating set size (Sondhi and Shojaie, 2019), they face similar scalability issues or require additional assumptions.

To address this complexity, several research directions have emerged. Some methods leverage minimal conditional (in)dependencies to obtain ancestral structures (Claassen et al., 2013; Magliacane et al., 2016). Other recent work shifts the focus from learning the entire graph to characterizing which parts of it can be learned under a constrained number of CI tests (Shiragur et al., 2024) or with small conditioning sets (Kocaoglu, 2023). Alternative formulations, such as testing a hypothesized DAG, have also been explored (Zhang et al., 2024). While valuable, these approaches either change the problem's scope or do not aim to recover the complete essential graph. In this work, by contrast, we recover the full essential graph with fewer number of CI tests.

While discrete-variable Bayesian network learning is known to be NP-hard (Chickering et al., 2004), causal discovery with CI oracles constitutes a distinct problem class that is not NP-hard under bounded degree assumptions (Claassen et al., 2013). Under the causal sufficiency assumption, our derived bounds guarantee that the essential graph can be recovered with a polynomial number of tests provided the size of the undirected cliques is bounded. This condition covers the case of bounded degree. Furthermore, for the general case of unbounded clique sizes, our lower bound suggests that constraint-based discovery with CI oracles is not in NP, since verifying a specific Markov equivalence class requires an exponential number of queries, but remains within EXP as the number of all possible DAGs is bounded by $2^{poly(p)}$.

Beyond computational efficiency, the correctness of any causal discovery algorithm depends on statistical assumptions that connect the observed data to the underlying DAG. For classic algorithms like `PC`, correctness in the infinite-sample limit is guaranteed by the Markov and faithfulness assumptions (Lauritzen, 1996; Spirtes et al., 2001). Recent work provides a framework for characterizing the precise correctness conditions for any given constraint-based algorithm, enabling a formal comparison of the required assumptions (Sadeghi, 2017; Teh et al., 2024). A related line of work explores how additional or redundant CI tests can be used to detect and correct errors, thereby increasing robustness (Faller and Janzing, 2025). These approaches highlight a fundamental goal: to precisely characterize the necessary and sufficient CI information required to recover the correct causal structure. In this work, we contribute to both fronts: we determine the number of CI tests required to recover the essential graph, and we show that our method requires strictly weaker correctness assumptions than the standard combination of Markov and faithfulness. Our empirical results further show that our algorithm achieves higher accuracy than existing methods in denser graphs as well as on synthetic gene expression data.

1.2 Organization

We begin by reviewing essential background and notation in Section 2. Our main theoretical results are summarized in Section 3. The proposed algorithm, along with its correctness guarantees and complexity, is presented in Section 4. In Section 5, we establish a lower bound in terms of number of CI tests, proving the optimality of our algorithm. Empirical validation is provided in Section 6. We conclude with a brief discussion in Section 7.

2 PRELIMINARIES

2.1 Graph Definitions

Let $\mathcal{G}$ be a graph consisting of a finite set of p nodes V and a set of edges E. For any two nodes $u, v \in V$, we write $u \sim v$ if they are adjacent and $u \not\sim v$ otherwise. Edges can be undirected $u - v$ or directed $u \to v$, $u \leftarrow v$. For any subset of nodes $W \subseteq V$, we denote its complement by $\bar{W} = V \setminus W$ and the subgraph it induces by $G[W]$. A *path* in $\mathcal{G}$ is a sequence of nodes such that consecutive nodes are adjacent. A path that starts and ends at the same node is a *cycle*. A cycle is *directed* if all its edges are directed and follow a consistent orientation. A *clique* is a subset of nodes where every two nodes are adjacent. In an *undirected clique* the nodes are connected by undirected edges. A graph containing directed and/or undirected edges is *partially directed*. A partially directed graph is a *chain graph* if it has no directed cycle. The connected components of a chain graph formed by removing all directed edges are known as *chain components*. An undirected graph is *chordal* if every cycle of length four or more has a chord, that is, an edge connecting two non-consecutive nodes in the cycle. A fully directed chain graph is called a *directed acyclic graph* (DAG). The *skeleton* of a graph, denoted by $\mathrm{sk}(\mathcal{G})$, is the undirected graph that results when all directed edges are replaced by undirected edges.

In directed graphs, for any node $v \in V$ we denote its sets of *parents*, *ancestors*, *children*, and *de-*

scendants in $\mathcal{G}$ by $\texttt{pa}(v)$, $\texttt{anc}(v)$, $\texttt{ch}(v)$ and $\texttt{des}(v)$, respectively. We use square brackets for inclusive sets: e.g., $\texttt{anc}[v] = \texttt{anc}(v) \cup \{v\}$. Following Shiragur et al. (2024), a subset $S \subseteq V$ is called a *prefix node set* if for all $v \in S$, $\texttt{anc}[v] \subseteq S$; in other words, a prefix node set is a subset of nodes that is closed under ancestral relations. For any subset $W \subseteq V$, we define the set of *source nodes* within W as $\texttt{src}(W) = \{v \in W \mid \texttt{anc}(v) \cap W = \varnothing\}$. When necessary to distinguish the underlying graph, we employ subscript notation, for example, $\texttt{pa}_{\mathcal{G}}$, $\texttt{src}_{\mathcal{G}}$. A node v is called a *collider* on a path if its adjacent nodes u and w on the path satisfies $u \to v \leftarrow w$. If these parent nodes u and w are not adjacent, this configuration is called a *v-structure*.

2.2 Markov Equivalence Classes

DAGs are commonly used in causality (Lauritzen, 1982; Pearl, 1985, 2009) where nodes represent random variables and edges represent direct causal relationships. A distribution $\mathbb{P}$ is *Markov* relative to a DAG $\mathcal{G}$, if it factorizes as $\mathbb{P}(v_1, \ldots, v_p) = \prod_{i=1}^{p} \mathbb{P}(v_i \mid \texttt{pa}(v_i))$. This factorization implies a set of conditional independence relations, which can be determined from $\mathcal{G}$ using *d-separation* (Pearl, 1988). For disjoint node sets $A, B, C \subset V$, A and B are *d-separated* by C if all paths connecting A and B are *inactive* given C. A path is *inactive* (or *blocked*) given C if it contains at least one node v satisfying one of the following conditions: (i) v is a non-collider on the path and $v \in C$, or (ii) v is a collider on the path and $\texttt{des}[v] \cap C = \varnothing$. Otherwise, the path is *active* given C. We denote conditional independence in the distribution $\mathbb{P}$ by $A \perp B \mid C$, and d-separation in the DAG $\mathcal{G}$ by $A \perp_{\mathcal{G}} B \mid C$. For singleton sets, for example $A = \{a\}$, we omit the braces for simplicity, $a \perp B \mid C$. Additionally, we write $A \perp B \mid C$ for potentially overlapping sets to denote $A \perp B \mid C \setminus (A \cup B)$.

Multiple DAGs can represent the same conditional independence information (Verma and Pearl, 1990). The set of all DAGs encoding the same independencies forms a *Markov equivalence class* (MEC), denoted $[\mathcal{G}]$. Markov equivalent DAGs are characterized by having the same skeleton and the same v-structures, which enables a unique representation of an MEC by its *essential graph*, $\mathcal{E}(\mathcal{G})$ (Andersson et al., 1997). The essential graph $\mathcal{E}(\mathcal{G})$ is a chain graph that shares the skeleton of the DAGs in $[\mathcal{G}]$ and contains a directed edge if and only if the orientation of the edge is the same in every DAG belonging to the MEC. These invariant edge orientations stem from v-structures and orientations given by the *Meek rules* (Meek, 1995; Andersson et al., 1997).

If a distribution $\mathbb{P}$ is Markov with respect to a DAG $\mathcal{G}$, then d-separation implies conditional independence; i.e., $A \perp_{\mathcal{G}} B \mid C \implies A \perp B \mid C$. Under the *faithfulness* assumption, the reverse implication holds. When $\mathbb{P}$ is both Markov and faithful relative to $\mathcal{G}$, we say that $\mathbb{P}$ *respects* $\mathcal{G}$, signifying: $A \perp B \mid C \iff A \perp_{\mathcal{G}} B \mid C$.

3 MAIN RESULTS

Under the Markov and faithfulness assumptions,[3] we provide an algorithm, called *Greedy Ancestral Search* (GAS), that achieves optimality in terms of the number of CI tests. Our algorithm is given in Section 4.2, with the following guarantees.

Theorem 1. *Given infinite samples from a distribution respecting a DAG $\mathcal{G}$, GAS (i.e., Algorithm 1) outputs $\mathcal{E}(\mathcal{G})$ using $p^{\mathcal{O}(s)}$ CI tests. Here, s is the size of the maximum undirected clique in $\mathcal{E}(\mathcal{G})$.*

This result shows that the number of CI tests required by GAS is upper bounded exponentially in the maximum undirected clique size of the essential graph. This is in contrast to current algorithms, which, without additional assumptions, require in the worst case the number of CI tests to be at least an exponential function of the maximum degree (Spirtes et al., 2001; Claassen et al., 2013). Complementing this result, we show that any algorithm requires at least $2^{\Omega(s)}$ CI tests to distinguish the ground-truth $[\mathcal{G}]$ versus other MECs.

Theorem 2. *Given infinite samples from a distribution respecting a DAG $\mathcal{G}$, any algorithm requires at least $2^{\Omega(s)}$ CI tests to verify $\mathcal{G} \in [\mathcal{G}]$. Specifically, for any collection of fewer than $2^s - s - 1$ CI tests, there exists an MEC $[\mathcal{H}] \neq [\mathcal{G}]$ such that both classes are indistinguishable based on these CI relations.*

Since any constraint-based causal discovery algorithm must distinguish the true MEC from all others, it follows that, without additional assumptions, it must perform at least $2^{\Omega(s)}$ CI tests. Together, Theorems 1 and 2 show that the complexity of constraint-based causal discovery, in terms of the number of CI tests, is exponential in s, and that our algorithm achieves this bound.

Additionally, we show that our algorithm can output the correct MEC when faithfulness is violated. As a consequence, its correctness condition is weaker than Markov and faithfulness, and hence it can output the correct essential graph in more settings. There exist other algorithms that also provably require weaker

[3] We remark here that we do not need to assume causal sufficiency for Theorem 1 to hold. Causal sufficiency needs to be assumed when one requires that $\mathcal{G}$ captures the complete causal explanation (Meek, 1997).

assumptions than Markov and faithfulness (Ramsey et al., 2006; Raskutti and Uhler, 2018; Solus et al., 2021), their complexity in terms of CI testing could be prohibitive. The proof of the following result can be found in Appendix D.

Proposition 3. *There exists a joint distribution that is Markov but not faithful to a DAG $\mathcal{G}$, for which Algorithm 1, when given observational data sampled from it, correctly outputs the ground-truth $\mathcal{E}(\mathcal{G})$.*

In the remainder of this section, we provide a concise overview of the methods used to establish our main theorems.

3.1 Outline Of Methods

Upper bound for GAS. Compared to other constraint-based algorithm (e.g., PC), the reason our algorithm performs fewer CI tests is its use of ancestral information during the adjacency search, which is learned by keeping track of both conditional independence *and* dependence statements. This ancestral information is inferred through two types of CI tests (given in Definitions 4 and 5 below), which primarily serve to identify edge orientations arising from v-structures and Meek rule 1. As we show in Section 4.2, these edge orientations are sufficient to recover all ancestral information in the underlying essential graph. GAS tracks this ancestral information through an iterative procedure that expands a prefix node set $S \subset V$. This process, which builds on Shiragur et al. (2024), serves two key roles: it enables learning the remaining ancestral relations in the essential graph, and it simplifies CI testing for adjacencies by restricting the search space of the conditioning sets.

Next, to show that the number of CI tests is bounded by s, we prove that the iterative expansion procedure terminates with a partition of the nodes V into disjoint chordal components $S_1, S_2, \ldots, S_m$, where each partial union $S_1 \cup S_2 \cup \cdots \cup S_i$ for $i = 1, \ldots, m$ forms a prefix node set. It follows from Dirac (1961) that any minimal separator within a chordal component S_i forms a clique. This property implies that the conditioning sets used for CI tests within S_i need not exceed the size of the largest clique in the skeleton, which can serve as a stopping criterion when searching for adjacencies within S_i. Further details are provided in Section 4.

Lower bound on any constraint-based algorithm. The argument is based on the largest undirected clique of size s in the essential graph. This clique contains $2^s - s - 1$ distinct subsets of size at most $s - 2$, each representing a potential conditioning set. If an algorithm performs fewer CI tests, there must exist at least one subset that has not been used as the conditioning set. For any such unused set, we show how to construct a distinct MEC that is consistent with all performed CI tests. A formal proof detailing this construction is provided in Section 5.

4 UPPER BOUND

In this section, we present our main algorithm and show that it outputs the correct essential graph using at most $p^{\mathcal{O}(s)}$ CI tests. Proofs omitted from this section are provided in Appendix B.

4.1 Learning A Prefix Node Set

From the orientation rules of PC (steps (2) and (3) described in Section 1), we see that the directional information in the essential graph are derived from v-structures and Meek rules. While four Meek rules exist for orienting edges, only rule 1 introduces ancestral information not already captured by transitivity (see Figure 6). Therefore, the ancestral relationships in an essential graph can be completely determined by its v-structures and the application of Meek rule 1. Leveraging this observation, we developed two sets of CI tests to specifically identify the ancestral relations established by v-structures (Definition 4) and Meek rule 1 (Definition 5).

Definition 4 (Set D_S^m)**.** For all $w \in \bar{S}$, let $w \in D_S^m$ if and only if $u \perp v \mid S \cup W$ and $u \not\perp v \mid S \cup W \cup \{w\}$ for some $u \in V$, $v \in \bar{S}$ and $W \subset \bar{S}$ with $|W| = m$.

These CI tests identify a downstream node w by its properties as a collider, or a descendant of a collider. Specifically, conditioning on w creates a dependency between two previously independent nodes, u and v. Definition 4 generalizes the concept of *minimal* conditional dependence (Claassen and Heskes, 2011; Magliacane et al., 2016), by incorporating a prefix node set.

Next, we define tests to identify nodes whose incident edges are oriented based on Meek rules.

Definition 5 (Set F_S^m)**.** For all $w \in \bar{S}$, let $w \in F_S^m$ if and only if $u \not\perp w \mid S$ and $u \perp w \mid S \cup W$ for some $u \in S$ and $W \subset \bar{S}$ with $|W| = m$.

This definition checks for a node w that is initially dependent on a node u in the prefix node set S, but becomes independent once we condition on $W \subseteq \bar{S}$. Graphically, this means that all active paths between u and w pass through W. This implies w has an ancestor in $\bar{S}$, indicating that it can be excluded when expanding the prefix node set S.

With these definitions, we can expand a prefix node

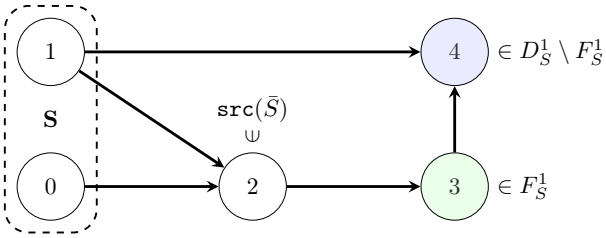


Figure 2: Example graph illustrating the classification of nodes into sets D_S^1 and F_S^1 given the prefix node set $S = \{0, 1\}$.

set $S \subset V$ (which may be empty) into a new, larger set $S' \subseteq V$.

Theorem 6. *Let $S \subset V$ be a prefix node set. For any integer ℓ, the set $S' = V \setminus (D_S^0 \cup D_S^1 \cup F_S^1 \cup \cdots \cup D_S^\ell \cup F_S^\ell)$ can be obtained with $\mathcal{O}(p^{\ell+3})$ number of CI tests. In addition, S' is a prefix node set satisfying $(S \cup \texttt{src}(\bar{S})) \subseteq S'$, that is, it contains all the remaining source nodes in $\bar{S}$.*

The proof of Theorem 6 relies on the following lemmas. These lemmas serve two purposes: first, they establish the properties of the sets previously defined, and second, they guarantee that S' strictly increases the size of the prefix node set S.

Lemma 7. *Let S be a prefix node set. If $w \in D_S^m$, then $\texttt{des}[w] \subseteq D_S^m$. Furthermore, we have $D_S^m \cap \texttt{src}(\bar{S}) = \varnothing$.*

The preceding lemma shows that D_S^m contains all descendants of its members. While descendants of nodes in F_S^m are not necessarily in F_S^m, they must be in $D_S^m \cup F_S^m$, as shown in the following lemma; see also Figure 2.

Lemma 8. *Let S be a prefix node set. If $w \in F_S^m \setminus \left(F_S^1 \cup \cdots \cup F_S^{m-1}\right)$, then $\texttt{des}[w] \subseteq F_S^m \cup D_S^m$. Furthermore, $F_S^m \cap \texttt{src}(\bar{S}) = \varnothing$.*

Proof of Theorem 6. We first show that S' satisfies $\texttt{src}(\bar{S}) \subset S'$. By Lemmas 7 and 8 we have $(D_S^0 \cup D_S^1 \cup F_S^1 \cup \cdots \cup D_S^\ell \cup F_S^\ell) \cap \texttt{src}(S) = \varnothing$. Since $\bar{S}' = D_S^0 \cup D_S^1 \cup F_S^1 \cup \cdots \cup D_S^\ell \cup F_S^\ell$, it must hold that $\texttt{src}(\bar{S}) \subset S'$.

Next we show that S' is a prefix node set. For this we only need to show that for all $x \in S'$ and $y \in \texttt{anc}(x)$, it holds that $y \in S'$. For contradiction, assume $x \in S'$ but $y \notin S'$. We must have $y \in D_S^i$ or $y \in F_S^j$ for some $i, j \leq m$. If $y \in D_S^i$, then by Lemma 7 we have $x \in D_S^i$, a contradiction. Otherwise, let j be the lowest number such that $y \in F_S^j$; then by Lemma 8 we have $x \in F_S^j \cup D_S^j$, a contradiction. Therefore S' must be a prefix node set.

We now bound the number of CI tests needed to obtain S'. Towards this, we derive the complexity of D_S^m and F_S^m. Constructing D_S^m requires checking, for all $w \in \bar{S}$, whether there exist $u \in V$, $v \in \bar{S}$, and $W \subset \bar{S}$ with $|W| = m$ such that $u \perp v \mid S \cup W$ and $u \not\perp v \mid S \cup W \cup \{w\}$. The required number of conditional independence tests is upper bounded by $2 \times \binom{|\bar{S}|}{1}\binom{|V|-1}{1}\binom{|\bar{S}|-1}{1}\binom{|\bar{S}|-2}{m}$, which is in $\mathcal{O}(p^{m+3})$. Therefore, computing D_S^m takes $\mathcal{O}(p^{m+3})$ CI tests; similarly, F_S^m takes $\mathcal{O}(p^{m+2})$ CI tests. Since we compute up to $m = \ell$, the total number of CI tests required is $\mathcal{O}(p^{\ell+3})$. □

4.2 Learning The Essential Graph Via GAS

Algorithm 1 Greedy Ancestral Search (GAS)

Input: CI queries from a distribution $\mathbb{P}$ that respects a DAG $\mathcal{G}$
Output: The essential graph of $\mathcal{G}$

```
S ← empty ordered list;
S ← ∅;
E ← complete undirected graph on V;
▷ Prefix node set expansion ◁
while S ≠ V do
    V' ← V \ S;
    ℓ ← 0;
    while there exists a clique in E[V'] with size ≥ ℓ do
        ▷ Edge removal ◁
        for v ∈ V, w ∈ V' s.t. v − w in E do
            Remove v − w in E if ∃W ⊂ V' with |W| = ℓ s.t. w ⊥ v | S ∪ W;
        ▷ Ordering learned from v-structures ◁
        Compute D_S^ℓ;
        V' ← V' \ D_S^ℓ;
        if ℓ > 0 then
            ▷ Ordering learned from Meek rules ◁
            Compute F_S^ℓ;
            V' ← V' \ F_S^ℓ;
        ℓ ← ℓ + 1;
    S ← S ∪ V';
    Add V' to the end of S;
for v ∈ S_i, w ∈ S_j with i < j do
    If v − w in E, replace with v → w;
return E
```

Our algorithm for learning essential graphs, Greedy Ancestral Search (GAS), is presented in Algorithm 1. It extends the prefix node set method of CCPG (Shiragur et al., 2024). Unlike CCPG, which only identifies a coarse, partition-level graph, GAS recovers the complete essential graph. In particular, we generalize their Definitions 4.1–4.3 by introducing the parameter m; the

original definitions are recovered as special cases from our Definitions 4 and 5 by choosing $m = 0$ or $m = 1$. In the following, we prove that this generalization is sufficient to identify all edges in the essential graph, and we establish a stopping criterion for Algorithm 1, allowing us to bound the number of CI tests needed.

`GAS` initializes with a complete undirected graph $\mathcal{E}$ on the node set V. It proceeds iteratively, constructing a sequence of expanding prefix node sets $\varnothing \subset \cdots \subset S \subset \cdots \subset V$ by greedily adding elements according to Theorem 6, while simultaneously removing edges from the working graph. The number of CI tests used in this construction is bounded by an exponential function of ℓ, which we show using our stopping criterion—to be no greater than the size of the maximum undirected clique in the essential graph. A detailed, step-by-step example in Appendix C illustrates the execution of Algorithm 1 on a 5-node graph, showing how the prefix node set is expanded, how edges are removed, and how the final edge orientations are determined.

The sets D_S^m and F_S^m are calculated using the CI test results from the edge removal step (Line 11). The computation of D_S^m, for example, only requires some targeted additional tests to identify descendants of v-structures (see Algorithm 3). The specific implementation is detailed in Appendix E.

We now sketch the proof of Theorem 1, which establishes the correctness and guarantees for Algorithm 1. The full formal proof can be found in Appendix B.3.

Correctness. To show that the algorithm correctly outputs the essential graph $\mathcal{E}(\mathcal{G})$, we formally characterize the nodes contained within the computed sets D_S^m and F_S^m (Lemmas 17 and 18). This characterization demonstrates that these sets accurately capture the necessary information for the algorithm to learn the partial ordering defined by directed edges $\mathcal{E}(\mathcal{G})$. This ordering is reflected in the partition $S_1, \ldots, S_m$ identified by Algorithm 1, where directed edges in $\mathcal{E}(\mathcal{G})$ exist only between nodes belonging to different components S_i and S_j.

Number of CI tests. The key is to bound the maximum order ℓ up to which the sets D_S^m and F_S^m must be computed. In Appendix B.3, we prove the following result: if $w \in D_S^m$ and $w \notin D_S^0 \cup \cdots \cup D_S^{m-1}$, then there must exist a clique of size m in $\mathtt{anc}(w) \setminus S$. This implies that the maximum required order is bounded by the maximum clique in any of the components S_i. This leads to the overall complexity bound of $p^{\mathcal{O}(s)}$ where s is the size of the maximum undirected clique in $\mathcal{E}(\mathcal{G})$.

5 LOWER BOUND

The following lemma from Zhang et al. (2024) specifies the exact CI tests on which two DAGs will disagree if they differ only by a single edge, provided that edge is undirected in the essential graph of the larger DAG.

Lemma 9 (Lemma 7 in Zhang et al. (2024))**.** *Let $\mathcal{G}$ and $\mathcal{H}$ be two DAGs such that $\mathcal{H}$ differs from $\mathcal{G}$ by missing one edge $u \to v$, which is undirected in $\mathcal{E}(\mathcal{G})$. Denote the maximum undirected clique in $\mathcal{G}$ containing u, v by S. For disjoint sets $A, B, C \subset [p]$, the statement of whether A and B are d-separated by C differs between $\mathcal{G}$ and $\mathcal{H}$ only if*

$$(\mathtt{pa}_{\mathcal{G}}(v) \cap \mathtt{ch}_{\mathcal{G}}(u)) \cap S \subseteq C \cap S \subseteq (\mathtt{pa}_{\mathcal{G}}(v) \setminus \{u\}) \cap S. \quad (1)$$

Proof of Theorem 2. First, note that by the binomial theorem we have $2^s = \sum_{i=0}^{s} \binom{s}{i}$. This number is exactly how many ways we can choose an unordered subset of any size from a fixed set of s elements. Therefore $2^s - \binom{s}{s-1} - \binom{s}{s} = 2^s - s - 1$ is exactly all the ways we can choose a subset of size less or equal to $s - 2$ from s nodes.

Let S be the largest undirected clique in $\mathcal{E}(\mathcal{G})$. If less than $2^s - s - 1$ tests have been done, there must be a subset $W \subseteq S$ of $s - 2$ nodes or less, such that no test $A \perp B \mid C$ has been performed where $C \cap S = W$. Since S is an undirected clique in $\mathcal{E}(\mathcal{G})$, its nodes can be ordered arbitrarily to obtain a valid DAG in $[\mathcal{G}]$. Let $\mathcal{F} \in [\mathcal{G}]$ be a DAG such that in the topological ordering of S all nodes in W are placed consecutively after the first node. Let u denote this first node in the topological ordering, and let v denote the node after all nodes in W. Let $\mathcal{H}$ be the DAG obtained by removing the edge $u \to v$ from $\mathcal{F}$. For disjoint sets A, B, C testing if A and B are independent given C will disagree between $\mathcal{F}$ and $\mathcal{H}$ only if $C \cap S = W$. This is because using Lemma 9 we have

$$\begin{aligned} W &= (\mathtt{pa}_{\mathcal{F}}(v) \cap \mathtt{ch}_{\mathcal{F}}(u)) \cap S \subseteq C \cap S \\ &\subseteq (\mathtt{pa}_{\mathcal{F}}(v) \setminus \{u\}) \cap S = W. \end{aligned} \quad (2)$$

Since no CI tests have been performed that satisfy $C \cap S = W$, all the performed tests agree with $[\mathcal{G}]$ and $[\mathcal{H}]$. Therefore, more tests must be performed in order to distinguish the MEC classes $[\mathcal{G}]$ and $[\mathcal{H}]$. □

We remark that our lower bound in Theorem 2 is strictly stronger than the one provided in Zhang et al. (2024). Our result provides an *any-case* guarantee: for any collection of fewer than $2^{\Omega(s)}$ CI tests, we show an indistinguishable MEC is guaranteed to exist. In contrast, Zhang et al. (2024) establish a *worst-case* result by constructing a specific MEC and proving that no algorithm can distinguish it with fewer than $2^{\Omega(s)}$ tests.

6 EXPERIMENTS

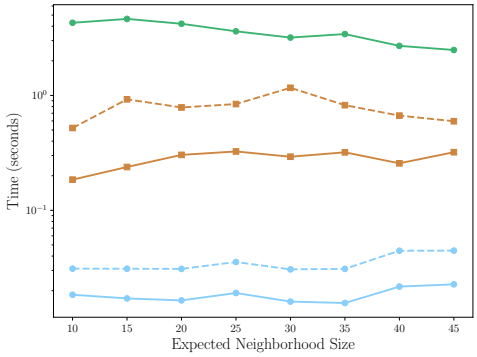
In this section, we empirically evaluate our proposed algorithm, Greedy Ancestral Search (GAS), and its variant, GAS+, which incorporates $\mathcal{O}(p^2)$ additional CI tests. We observe that these additional CI tests improve performance in practice (details can be found in Appendix F). We also benchmark performance against established causal discovery algorithms on synthetic data from linear Gaussian structural equation models (Wright, 1921) on random Erdős-Rényi graphs (Erdős and Réni, 1959), semi-synthetic gene expression data obtained using the SERGIO simulator (Dibaeinia and Sinha, 2020), and a real-world dataset. Implementation details and data descriptions are provided in Appendices E and G, respectively. Appendix H presents additional experiments, including results on scale-free graphs, networks with thousands of nodes, and comparisons of the number of CI tests performed by different algorithms. All code for these experiments is available at https://github.com/uhlerlab/greedy-ancestral-search.

6.1 Linear Gaussian Synthetic Data

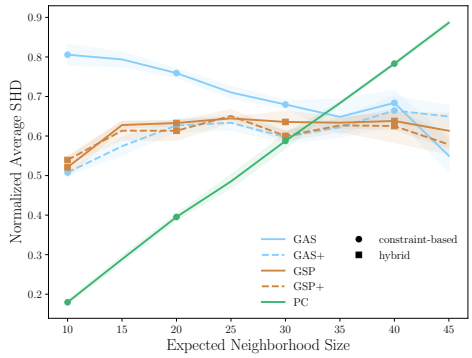
We compare both implementations of Algorithm 1, GAS and GAS+, against other constraint-based and hybrid algorithms that rely on CI tests. We include the popular constraint-based method PC (Spirtes et al., 2001), using the order-independent variant from Colombo and Maathuis (2014). We additionally include GSP with both depth 4 and depth ∞ (Solus et al., 2021). A comparison with algorithms FCI and GRaSP$_2$ (Spirtes et al., 2001; Lam et al., 2022), along with additional metrics and a similar experiment scaled to 2500 nodes, can be found in Appendix H.

To ensure fair comparisons, evaluations with algorithms originating from different softwares are presented separately. Within each evaluation set, GAS and GAS+ used the same CI tester as the respective baseline algorithms.

The experimental results are presented in Figure 3. Regarding computational speed, a consistent finding is that both variants of our algorithm, GAS and GAS+, are significantly faster than the other methods. Analyzing the prediction accuracy shown in Figure 3b, GAS+ achieves accuracy comparable to the GSP algorithms. The PC algorithm demonstrates high accuracy in sparser graphs, but its performance degrades by a large margin in denser graphs. In contrast, GAS performs fewer tests, resulting in the highest accuracy in denser graphs but a lower accuracy in sparse settings.



(a)



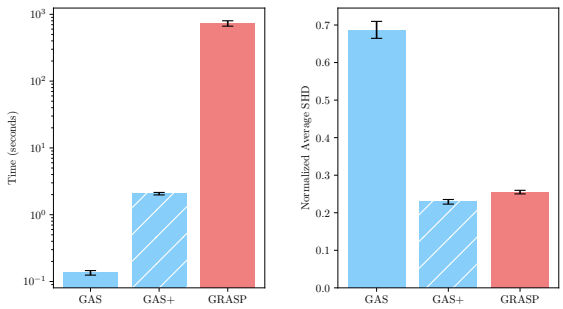
(b)

Figure 3: Comparison across Erdős-Rényi graphs on 50 nodes and increasing expected neighborhood size. Results are averaged over 3 runs. (a) Execution time (seconds) presented on a logarithmic scale. (b) Accuracy comparison measured by Structural Hamming Distance (SHD) between predicted and ground-truth graphs normalized by the number of possible edges. Shaded regions show the standard deviation.

6.2 SERGIO

For our semi-synthetic evaluation, we simulated gene expression data using the SERGIO model (Dibaeinia and Sinha, 2020). This enables precise comparison of inferred structures against the ground truth DAG. As in Dibaeinia and Sinha (2020), we utilized their DS1 graph, which samples 2700 cells each with the expression of 100 genes. Further details on this experimental setup are provided in Appendix G.

Figure 4 presents the comparative results on this semi-synthetic data. The high-degree structure of this dataset makes standard algorithms like PC and FCI computationally infeasible due to prohibitive runtimes (further discussed in Appendix G.2). We therefore benchmark our proposed methods, GAS and GAS+, against



(a) (b)

Figure 4: Comparison on semi-synthetic data generated using SERGIO. Results are averaged over 5 runs. (a) Execution time (seconds) presented on a logarithmic scale. (b) Accuracy comparison measured by Structural Hamming Distance (SHD) between predicted and ground-truth graphs normalized by the number of possible edges.

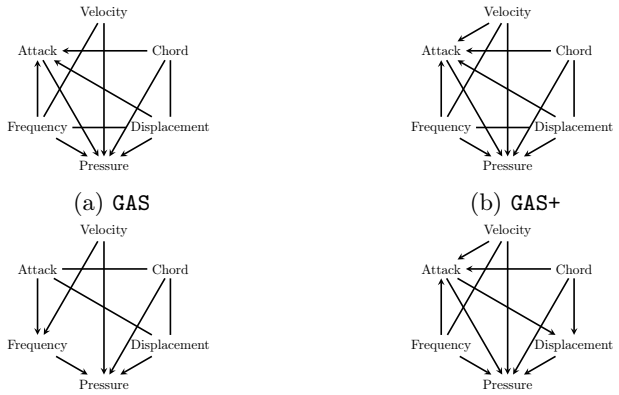
GRaSP$_2$ (Lam et al., 2022).[4] The findings indicate that while GAS is the fastest algorithm overall, GAS+ achieves the highest accuracy in recovering the underlying causal structure while being much faster than GRaSP$_2$.

6.3 Real-World Example

We consider the 6-variable Airfoil dataset (Asuncion and Newman, 2007); see Lam et al. (2022). Lacking a complete ground-truth DAG, we assess consistency on the known causal relations for this system: (1) *velocity*, *chord* and *attack* are experimental variables and thus exogenous, while (2) *pressure* is the measured outcome, expected to be an effect of the other variables.

Figure 5 shows the partially directed graphs learned by different algorithms on the Airfoil dataset. While all algorithms correctly identify *pressure* as a downstream variable, GAS is the most efficient, requiring the fewest CI tests and running twice as fast as the score-based GRaSP$_2$ algorithm. However, we note that only PC correctly identifies *attack* as an exogenous variable. For GAS and GAS+, this discrepancy stems from their reliance on conditional dependencies to infer ancestry. Specifically, the data indicates that *frequency* and *chord* are marginally independent but become dependent when conditioned on *attack*. If we assume the distribution respects an underlying DAG, this pattern implies that *attack* is a collider (or a descendant of one) and thus non-exogenous. While the CI test results may be correct, this implication can be broken by latent confounders.

[4] GSP's performance was found to be less competitive compared to the methods presented here.



(a) GAS (b) GAS+

(c) PC (d) GRaSP$_2$

Figure 5: Partially directed graphs learned in the Airfoil dataset. GAS, GAS+, and PC performed 46, 50, and 104 distinct CI tests, respectively.

7 DISCUSSION

In this work, we established tight bounds on the number of conditional independence tests required for causal structure discovery using constraint-based algorithms. We propose an exponent-optimal algorithm achieving this bound up to a logarithmic factor, requiring less CI tests than existing algorithms. Our empirical evaluations demonstrate that the proposed algorithm is significantly faster than established baseline methods and achieves highly competitive accuracy, particularly with denser graphs.

Limitations and future work. This work is motivated by the goal of improving the efficiency of constraint-based algorithms, both in terms of computational speed and correctness. We characterize efficiency by the number of CI tests performed, which directly impacts computational speed. However, as discussed in Section 1.1, the number of CI tests is related with, but not equivalent to, the exact correctness conditions required by an algorithm. While we have shown that our algorithm can succeed under assumptions weaker than the standard Markov and faithfulness conditions, its precise correctness guarantees—and how they compare to those of existing algorithms—remain unknown. Moreover, understanding how finite-sample effects influence these correctness conditions is an important direction for future work, critical to understanding the potential and limitations of constraint-based causal discovery. Finally, our algorithm's reliance on both conditional independencies and dependencies raises practical considerations. Future work should explore whether the algorithm's performance on finite, real-world data can be optimized by using different p-value thresholds or distinct statistical tests for each constraint.

Acknowledgements

We thank Kiran Shiragur for initial discussions that inspired this work. We also thank the anonymous reviewers for their helpful feedback. M.F.M. was partially supported by the Eric and Wendy Schmidt Center at the Broad Institute, Fundació Privada Mir-Puig, and a MOBINT-MIF grant. J.Z. was partially supported by an Apple AI/ML PhD Fellowship and the Eric and Wendy Schmidt Center at the Broad Institute. C.U. was partially supported by NCCIH/NIH (1DP2AT012345), ONR (N00014-24-1-2687), and the United States Department of Energy (DE-SC0023187).

References

Alonso-Barba, J. I., Gámez, J. A., Puerta, J. M., et al. (2013). Scaling up the greedy equivalence search algorithm by constraining the search space of equivalence classes. *International Journal of Approximate Reasoning*, 54(4):429–451.

Andersson, S. A., Madigan, D., and Perlman, M. D. (1997). A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505–541.

Arora, S. and Barak, B. (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Asuncion, A. and Newman, D. (2007). UCI Machine Learning Repository.

Brenner, E. and Sontag, D. (2013). Sparsityboost: A new scoring function for learning Bayesian network structure. *arXiv preprint arXiv:1309.6820*.

Bron, C. and Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577.

Brooks, T. F., Pope, D. S., and Marcolini, M. A. (1989). Airfoil Self-Noise. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5VW2C.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(Nov):507–554.

Chickering, D. M., Heckerman, D., and Meek, C. (2004). Large-sample learning of bayesian networks is np-hard. *Journal of Machine Learning Research*, 5(Oct):1287–1330.

Claassen, T. and Heskes, T. (2011). A logical characterization of constraint-based causal discovery. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, page 135–144, Arlington, Virginia, USA. AUAI Press.

Claassen, T., Mooij, J. M., and Heskes, T. (2013). Learning sparse causal models is not NP-hard. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 172–181, Arlington, Virginia, USA. AUAI Press.

Colombo, D. and Maathuis, M. H. (2014). Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research*, 15(1):3741–3782.

Colombo, D., Maathuis, M. H., Kalisch, M., and Richardson, T. S. (2012). Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321.

Dibaeinia, P. and Sinha, S. (2020). SERGIO: a single-cell expression simulator guided by gene regulatory networks. *Cell Systems*, 11(3):252–271.

Dirac, G. A. (1961). On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25(1-2):71–76.

Duncan, O. D. (1975). *Introduction to structural equation models*. Academic Press.

Erdős, P. and Réni, A. (1959). On random graphs. I. *Publicationes Mathematicae Debrecen*, 6(290-297):18.

Faller, P. M. and Janzing, D. (2025). On different notions of redundancy in conditional-independence-based discovery of graphical models. *arXiv preprint arXiv:2502.08531*.

Geiger, D. and Heckerman, D. (2002). Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *The Annals of Statistics*, 30(5):1412–1440.

Haavelmo, T. (1943). The statistical implications of a system of simultaneous equations. *Econometrica, Journal of the Econometric Society*, pages 1–12.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Kalisch, M. and Bühlman, P. (2007). Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8(3).

Kitson, N. K., Constantinou, A. C., Guo, Z., Liu, Y., and Chobtham, K. (2023). A survey of Bayesian network structure learning. *Artificial Intelligence Review*, 56(8):8721–8814.

Kocaoglu, M. (2023). Characterization and learning of causal graphs with small conditioning sets. *Advances in Neural Information Processing Systems*, 36:74140–74179.

Lam, W.-Y., Andrews, B., and Ramsey, J. (2022). Greedy relaxations of the sparsest permutation algorithm. In *Uncertainty in Artificial Intelligence*, pages 1052–1062. PMLR.

Lauritzen, S. L. (1982). *Lectures on Contingency Tables*. University of Aalborg Press, Aalborg, Denmark, 2nd edition.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford University Press.

Magliacane, S., Claassen, T., and Mooij, J. M. (2016). Ancestral causal inference. *Advances in Neural Information Processing Systems*, 29.

Mazaheri, B., Zhang, J., and Uhler, C. (2025). Faithfulness and intervention-only causal discovery. In *ICML 2025 Workshop on Scaling Up Intervention Models*.

Meek, C. (1995). Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, page 403–410, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Meek, C. (1997). *Graphical Models: Selecting causal and statistical models*. PhD thesis, Carnegie Mellon University.

Mokhtarian, E., Elahi, S., Akbari, S., and Kiyavash, N. (2025). Recursive causal discovery. *Journal of Machine Learning Research*, 26(61):1–65.

Nandy, P., Hauser, A., and Maathuis, M. H. (2018). High-dimensional consistency in score-based and hybrid structure learning. *The Annals of Statistics*, 46(6A):3151–3183.

Pearl, J. (1985). Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 329–334.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc.

Pearl, J. (2009). *Causality*. Cambridge University Press.

Ramsey, J., Spirtes, P., and Zhang, J. (2006). Adjacency-faithfulness and conservative causal inference. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, page 401–408, Arlington, Virginia, USA. AUAI Press.

Raskutti, G. and Uhler, C. (2018). Learning directed acyclic graph models based on sparsest permutations. *Stat*, 7(1):e183.

Richardson, T. (1996). A discovery algorithm for directed cyclic graphs. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, UAI'96, page 454–461, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Sadeghi, K. (2017). Faithfulness of probability distributions and graphs. *Journal of Machine Learning Research*, 18(148):1–29.

Schulte, O., Frigo, G., Greiner, R., and Khosravi, H. (2010). The IMAP hybrid method for learning Gaussian Bayes nets. In *Advances in Artificial Intelligence: 23rd Canadian Conference on Artificial Intelligence, Canadian AI 2010, Ottawa, Canada, May 31–June 2, 2010. Proceedings 23*, pages 123–134. Springer.

Shiragur, K., Zhang, J., and Uhler, C. (2024). Causal discovery with fewer conditional independence tests. In *Forty-first International Conference on Machine Learning*.

Solus, L., Wang, Y., and Uhler, C. (2021). Consistency guarantees for greedy permutation-based causal inference algorithms. *Biometrika*, 108(4):795–814.

Sondhi, A. and Shojaie, A. (2019). The reduced PC-algorithm: Improved causal structure learning in large random networks. *Journal of Machine Learning Research*, 20(164):1–31.

Spirtes, P., Glymour, C., and Scheines, R. (1989). Causality from probability.

Spirtes, P., Glymour, C., and Scheines, R. (2001). *Causation, Prediction, and Search*. The MIT Press.

Squires, C. (2018). *causaldag: creation, manipulation, and learning of causal models*.

Teh, K. Z., Sadeghi, K., and Soo, T. (2024). A general framework for constraint-based causal learning. *arXiv preprint arXiv:2408.07575*.

Uhler, C., Raskutti, G., Bühlmann, P., and Yu, B. (2013). Geometry of the faithfulness assumption in causal inference. *The Annals of Statistics*, pages 436–463.

Verma, T. and Pearl, J. (1990). Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '90, page 255–270, USA. Elsevier Science Inc.

Wright, S. (1921). Correlation and causation. *Journal of agricultural research*, 20(7):557.

Zhang, J., Shiragur, K., and Uhler, C. (2024). Membership testing in Markov equivalence classes via independence queries. In *International Conference on Artificial Intelligence and Statistics*, pages 3925–3933. PMLR.

Zheng, Y., Huang, B., Chen, W., Ramsey, J., Gong, M., Cai, R., Shimizu, S., Spirtes, P., and Zhang, K. (2024). Causal-learn: Causal discovery in Python. *Journal of Machine Learning Research*, 25(60):1–8.

CHECKLIST

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable]
   Yes. We do a thorough overview in Section 2.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable]
   Yes. The analysis is focused on the number of conditional independence tests required.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable]
   Yes. Anonymized source code can be found in the supplementary materials.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable]
   Yes.
   (b) Complete proofs of all theoretical results. [Yes/No/Not Applicable]
   Yes. Proofs not in the main manuscript can be found in the supplementary materials.
   (c) Clear explanations of any assumptions. [Yes/No/Not Applicable]
   Yes. The faithfulness assumption is stated and explained.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable]
   Yes. Code and instructions can be found in the supplementary materials.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable]
   Not applicable.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes/No/Not Applicable]
   Yes. All measures, statistics and error bars are well defined.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes/No/Not Applicable]
   Yes. In the supplementary materials.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes/No/Not Applicable]
   Yes.
   (b) The license information of the assets, if applicable. [Yes/No/Not Applicable]
   Yes.
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable]
   Yes.
   (d) Information about consent from data providers/curators. [Yes/No/Not Applicable]
   Not applicable.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable]
   Not applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Yes/No/Not Applicable]
   Not applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable]
   Not applicable.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable]
   Not applicable.

On the Number of Conditional Independence Tests in Constraint-based Causal Discovery: Supplementary Materials

A MEEK RULES

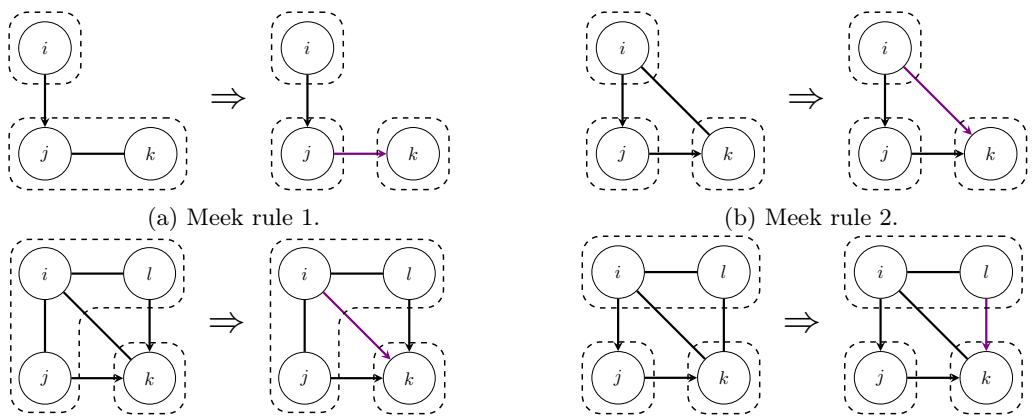
Proposition 10 (Meek rules, Meek (1995))**.** *We can infer all directed edges in $\mathcal{E}(\mathcal{G})$ using the following four rules:*

R1. If $i \to j \sim k$ and $i \not\sim k$, then $j \to k$.

R2. If $i \to j \to k$ and $i \sim k$, then $i \to k$.

R3. If $i \sim j, i \sim k, i \sim l, j \to k, l \to k$ and $j \not\sim l$, then $i \to k$.

R4. If $i \sim j, i \sim k, i \sim l, j \to k, i \to j$ and $j \not\sim l$, then $l \to k$.



(a) Meek rule 1. (b) Meek rule 2.

(c) Meek rule 3. (d) Meek rule 4.

Figure 6: Meek rules (Meek, 1995). The dashed lines partition nodes into sets based on ancestral relationships. Only Meek rule 1 establishes an ancestor for a node that previously had none, forcing a component to split.

B OMITTED PROOFS OF UPPER BOUND

We will start by showing the following lemma.

Lemma 11. *Let $S \subset V$ be a prefix node set. Let A, B and C be disjoint subsets such that $A \subset V$, and $B, C \subset \bar{S}$. Let $\mathcal{H} = \mathcal{G}[\bar{S} \cup A]$ be the induced subgraph of $\mathcal{G}$ on the node set $\bar{S} \cup A$. Then $A \not\perp_{\mathcal{G}} B \mid (S \setminus A) \cup C$ if and only if $A \not\perp_{\mathcal{H}} B \mid C$.*

Proof. We will first prove the *if* direction. Assume $A \not\perp_{\mathcal{H}} B \mid C$. This means there exists an active path P in $\mathcal{H}$ between a node in A and a node in B, given C. By definition of $\mathcal{H}$ as $\mathcal{G}[\bar{S} \cup A]$, all nodes on the path P necessarily belong to $\bar{S} \cup A$, which means no node on P is part of $S \setminus A$. For P to be active in $\mathcal{H}$ given C, every non-collider on P must not be in C, and every collider on P must either be in C or have a descendant in C. Now, considering

this same path P within the graph $\mathcal{G}$ and conditioned on $(S \setminus A) \cup C$: any non-collider on P, not being in C (due to being active in $\mathcal{H}$) and not being in $S \setminus A$ (due to being fully in $\mathcal{H}$) is therefore not in $(S \setminus A) \cup C$. Similarly, any collider on P that is in $\texttt{anc}_{\mathcal{H}}[C]$ is consequently in $\texttt{anc}_{\mathcal{G}}[(S \setminus A) \cup C]$. Since path P satisfies the criteria for an active path in $\mathcal{G}$ when conditioned on $(S \setminus A) \cup C$, it follows that $A \not\perp_{\mathcal{G}} B \mid (S \setminus A) \cup C$.

For the *only if* direction, we assume $A \not\perp_{\mathcal{G}} B \mid (S \setminus A) \cup C$. This assumption implies there exists a path P between a node in A and a node in B that is active in $\mathcal{G}$ when conditioned on $(S \setminus A) \cup C$; we can select P such that only its endpoints are in $A \cup B$. For P to be active, any non-collider on this path must not be in $(S \setminus A) \cup C$, which directly means non-colliders on P must reside in $A \cup \bar{S} \setminus C$. Colliders on P, on the other hand, must be in $\texttt{anc}_{\mathcal{G}}[(S \setminus A) \cup C]$. Consider a collider c on path P. If c were in $\texttt{anc}_{\mathcal{G}}[S \setminus A]$, then because S is a prefix set (which implies $\texttt{anc}_{\mathcal{G}}[S \setminus A] \subseteq S$), c itself must be in S. As c is an intermediate node on P (not in $A \cup B$), c being in S means $c \in S \setminus A$. The argument then follows that if such a collider $c \in S \setminus A$ is on P, the prefix nature of S necessitates that at least one of c's neighbors on path P must also be in $S \setminus A$ while being an intermediate node on P as $B \subset \bar{S}$. This neighbor, however, must function as a non-collider on P while also being in $S \setminus A$, thereby blocking P in $\mathcal{G}$, a contradiction. Therefore, any collider on path P must be in $\texttt{anc}_{\mathcal{G}}[C] \setminus S \subseteq \texttt{anc}_{\mathcal{H}}[C]$. Given that non-colliders are in $A \cup \bar{S} \setminus C$, and colliders are in $\texttt{anc}_{\mathcal{H}}[C]$, it follows that P is active in $\mathcal{H}$ given C. This implies $A \not\perp_{\mathcal{H}} B \mid C$. □

We will make use of the following lemma.

Lemma 12. *Let $S \subset V$ be a prefix set. Let $a, b \in V$ and $W \subset \bar{S}$ be such that $a \perp b \mid S \cup W$ and $a \not\perp b \mid S \cup W'$ for all $W' \subset W$. Then, $W \subseteq \texttt{anc}(a) \cup \texttt{anc}(b)$.*

Proof. We aim to show that every element $w \in W$ must be a member of $\texttt{anc}(a) \cup \texttt{anc}(b)$. The argument can be understood by iteratively considering how elements of W contribute to blocking paths. If $W = \varnothing$, the inclusion in $\texttt{anc}(a) \cup \texttt{anc}(b)$ is trivial.

1. *Initial element.* Since $W' = \varnothing$ is a proper subset of W, we have $a \not\perp b \mid S$. Therefore, there must exist an active path P between a and b given S. Since S is prefix, by Lemma 11, there are no intermediate nodes in S and therefore, the path does not contain colliders. This implies that all nodes on P are necessarily within $\texttt{anc}(a) \cup \texttt{anc}(b)$. Since $a \perp b \mid S \cup W$, this path P must be blocked by $S \cup W$. Thus, at least one element from W, termed x_1, must lie on P as a non-collider. Being a node on P, x_1 is therefore in $\texttt{anc}(a) \cup \texttt{anc}(b)$.

2. *Subsequent elements.* Let $X_k = \{x_1, \ldots, x_k\}$ be a subset of W for which it has been established that $X_k \subseteq \texttt{anc}(a) \cup \texttt{anc}(b)$. If $X_k \neq W$, we must have $a \not\perp b \mid S \cup X_k$. So there exists an active path, P_k, between a and b given $S \cup X_k$. Since $S \cup X_k$ does not block P_k, a node from $W \setminus X_k$, termed x_{k+1}, must block P_k. Since the P_k is active, any collider on P_k must also be in $\texttt{anc}[X_k]$. Since $X_k \subseteq \texttt{anc}(a) \cup \texttt{anc}(b)$ we have $\texttt{anc}[X_k] \subseteq \texttt{anc}(a) \cup \texttt{anc}(b)$. This implies that all nodes on the entire path P_k are in $\texttt{anc}(a) \cup \texttt{anc}(b)$. Since x_{k+1} is on P_k, x_{k+1} must also be in $\texttt{anc}(a) \cup \texttt{anc}(b)$.

Therefore every element in W must be in $\texttt{anc}(a) \cup \texttt{anc}(b)$. □

B.1 Proof Of Lemma 7

To prove Lemma 7, we will make use of the following definition and lemma.

Definition 13 (Set A_S)**.** Let $S \subset V$ be a set of nodes not necessarily prefix. For all $w \in \bar{S}$, let $w \in A_S$ if and only if $u \perp v \mid S$ and $u \not\perp v \mid S \cup \{w\}$ for some $u \in V$ and $v \in \bar{S}$.

Lemma 14. *If $w \in A_S$, then $\texttt{des}[w] \subset A_S$. Furthermore, $A_S \cap \texttt{src}(\bar{S}) = \varnothing$.*

Proof. We first show that if $w \in A_S$, then it must hold that $\texttt{des}[w] \subset A_S$: since $w \in A_S$, there exists a node $u \in V$ and $v \in \bar{S}$ such that $u \perp v \mid S$ and $u \not\perp v \mid S \cup \{w\}$. We now show that for any $x \in \texttt{des}[w]$, we have $u \not\perp v \mid S \cup \{x\}$.

Since $u \not\perp v \mid S \cup \{w\}$, there is a path P from u to v that is active given $S \cup \{w\}$. Therefore, all non-colliders on P are not in $S \cup \{w\}$ and all colliders on P are in $\texttt{anc}[S \cup \{w\}]$. Since $x \in \texttt{des}[w]$, all colliders on P are in

$\texttt{anc}[S \cup \{x\}]$. If all non-colliders are not in $S \cup \{x\}$, then P is an active path from u to v given $S \cup \{x\}$, and thus $u \not\perp v \mid S \cup \{x\}$. Otherwise there is a non-collider on P that is x.

Since $u \perp v \mid S$, the path P is inactive given S. From above we know that all non-colliders on P are not in S. Therefore there exists a collider on P that is not in $\texttt{anc}[S]$. Suppose the leftmost and rightmost such colliders are k, k' (it is possible that $k = k'$), then k, k' must be in $\texttt{anc}[S \cup \{w\}] \setminus \texttt{anc}[S] \subseteq \texttt{anc}[w] \subset \texttt{anc}[x]$. Therefore, the path P takes the form: $u - \cdots \to k \leftarrow \cdots \to k' \leftarrow \cdots - v$. If x is between u and k (or between k' and v), consider the path Q in the graph by cutting out the parts between x and k' (or between k and x) on P and replacing them with directed edges from k' to x (or from k to x). Compared to P, the additional non-colliders on Q are all on the directed path from k' to x (or from k to x). They are not in S since $k, k' \notin \texttt{anc}[S]$, and thus Q has no non-colliders in S.

Compared to P, there is no collider on P that is not in $\texttt{anc}[S]$ and is still on Q by the fact that k, k' are leftmost and rightmost colliders on P that are not in $\texttt{anc}[S]$. Therefore, x must be a collider on Q, or else Q is active given S and $u \not\perp v \mid S$. Therefore all non-colliders on Q are not in $S \cup \{x\}$. Every collider on Q is either x or a collider of P, which is in $\texttt{anc}[S \cup \{x\}]$. Thus Q is active given $S \cup \{x\}$. Therefore $u \not\perp v \mid S \cup \{x\}$.

If x is between k and k' we can replace the part between k and x by the direct path from k to x and the part between x and k' by the direct path from k' to x. There is no additional collider on the new path and all non-colliders are not in S since $k, k' \notin \texttt{anc}[S]$. Therefore the path is active given $S \cup \{x\}$ and $u \not\perp v \mid S \cup \{x\}$.

Next we show that $A_S \cap \texttt{src}(\bar{S}) = \varnothing$: for contradiction assume that there exists a node $a \in \texttt{src}(\bar{S})$ such that $a \notin \bar{S} \backslash A_S$, that is $a \in \texttt{src}(\bar{S})$ and for some node $u \in V, v \in \bar{S}$, $u \perp v \mid S$ and $u \not\perp v \mid S \cup \{a\}$.

Since $u \perp v \mid S$ and $u \not\perp v \mid S \cup \{a\}$, there exists a path P between u and v which is inactive when conditioned on S but is active upon conditioning on $S \cup \{a\}$. Moreover, this path contains a node b that is a collider on P and satisfies: $a \in \texttt{des}[b]$ and $\texttt{des}[b] \cap S = \varnothing$. Since $\texttt{des}[b] \cap S = \varnothing$, we have that $b \in \bar{S}$. Furthermore, since $b \in \bar{S}, a \in \texttt{src}(\bar{S})$ and $a \in \texttt{des}[b]$, this implies that $b = a$. Therefore, the path P takes the form: $P = v - \cdots \to a \leftarrow \cdots - u$. All the colliders on the path P either belong to or have descendant in the set $S \cup \{a\}$.

Now consider the path $v - \cdots \to a$ and note that it is active given S. Let k be the number of nodes between v and a on this path $v - v_1 - \cdots - v_k \to a$. It is immediate that $v_k \in S$ since $a \in \texttt{src}(\bar{S})$. However, since $v_k \in S$, and since we condition on the set S, this should be a collider for the path Q to be active, which is not possible. Thus, we get a contradiction, which completes the proof. □

Lemma 7. *Let S be a prefix node set. If $w \in D_S^m$, then $\texttt{des}[w] \subseteq D_S^m$. Furthermore, we have $D_S^m \cap \texttt{src}(\bar{S}) = \varnothing$.*

Proof. Since $w \in D_S^m$, there exists two nodes $u \in V$, $v \in \bar{S}$ and a set $W \subset \bar{S}$ with $|W| = m$ such that $u \perp v \mid S \cup W$ and $u \not\perp v \mid S \cup W \cup \{w\}$. Let $S' = S \cup W$. We have that $w \in A_{S'}$ and by Lemma 14, $\texttt{des}[w] \subset A_{S'}$, $A_{S'} \cap \texttt{src}(\bar{S'}) = \varnothing$. Since $A_{S'} \subset D_S^m$ we have that $\texttt{des}[w] \subset D_S^m$. Additionally, $\texttt{src}(\bar{S}) \subset W \cup \texttt{src}(\bar{S'})$ and $w \notin W$, therefore $w \notin \texttt{src}(\bar{S})$. □

B.2 Proof Of Lemma 8

To prove Lemma 8 we will make use of the following result.

Lemma 15. *Let S be a prefix set. Let $w \in \bar{S}$ be such that $w \notin F_S^1 \cup \cdots \cup F_S^{m-1}$ and $w \in F_S^m$. Let $u \in S$ and $W \subset \bar{S}$ be such that $u \not\perp w \mid S$ and $u \perp w \mid S \cup W$. Then $W \subset \texttt{anc}(w)$.*

Proof. Since $u \not\perp w \mid S \cup W'$ for all $W' \subset W$, by Lemma 12 we have $W \subset \texttt{anc}(u) \cup \texttt{anc}(w)$. However, given that $u \in S$, S is a prefix set and $W \subset \bar{S}$, it necessarily follows that $W \subset \texttt{anc}(w)$. □

Lemma 8. *Let S be a prefix node set. If $w \in F_S^m \setminus \left(F_S^1 \cup \cdots \cup F_S^{m-1}\right)$, then $\texttt{des}[w] \subseteq F_S^m \cup D_S^m$. Furthermore, $F_S^m \cap \texttt{src}(\bar{S}) = \varnothing$.*

Proof. We first show that if $w \in F_S^m \setminus F_S^1 \cup \cdots \cup F_S^{m-1}$, then for any $y \in \texttt{des}(w)$, we have $y \in F_S^m \cup D_S^m$. Since $w \in F_S^m$, we have $u \not\perp w \mid S$ and $u \perp w \mid S \cup W$ for some $u \in S$.

Take the active path between u and w given S and extend it by the directed path from w to y. Note that none of nodes on the directed path from w to y are in S, since S is prefix and $w \notin S$. Therefore, this extended path is also active given S, which means $u \not\perp y \mid S$.

Thus, if $y \notin F_S^m$, then it must hold that $u \not\perp y \mid S \cup W$. This means there is an active path, denoted by P, between u and y given $S \cup W$. Consider extending this path by the directed path from w to y, denoted as Q. Compared to P, the additional non-colliders on Q are not in $S \cup W$: for S, this is because S is prefix, $w \notin S$, and all additional non-colliders are descendants of w; for W, this is because by Lemma 15 we have $W \cap \texttt{des}(w) = \varnothing$. Thus, Q is active given $S \cup W$, unless y is a collider on Q. Since $u \perp w \mid S \cup W$, the path Q must be inactive given $S \cup W$, which means y is a collider on Q. This means Q is active given $S \cup W \cup \{y\}$. Therefore, $u \not\perp w \mid S \cup W \cup \{y\}$. Together with $u \perp w \mid S \cup W$, we have $y \in D_S^m$.

Next we show that $F_S^m \cap \texttt{src}(\bar{S}) = \varnothing$. Since $w \in F_S^m$, we have $u \not\perp w \mid S$ and $u \perp w \mid S \cup W$ for some $u \in S$ and $W \subset \bar{S}$, $|W| = m$. Let $W' \subset \bar{S}$ be the smallest set such that $u' \perp w \mid S \cup W'$ for some $u' \in S$. We must have $1 \leq |W'| \leq |W|$. Additionally, Lemma 15 shows that $W' \subset \texttt{anc}(w)$. Since W' is a non-empty subset of $\bar{S}$ containing ancestors of w we have $w \notin \texttt{src}(\bar{S})$. □

B.3 Proof Of Theorem 1

We develop the proof of Theorem 1 through a series of intermediate results. Our first key lemma establishes that any node serving as the collider in a v-structure is necessarily contained within a set D_S^m.

Lemma 16. *Let $S \subset V$ be a prefix node set. If there are three nodes $u \in V$, $k, v \in \bar{S}$ that form a v-structure $u \to k \leftarrow v$, then there exists a set $W \subset \bar{S}$ such that $u \perp v \mid S \cup W$ and $u \not\perp v \mid S \cup W \cup \{k\}$.*

Proof. The path $u \to k \leftarrow v$ is active given k and therefore $u \not\perp v \mid S \cup W \cup \{k\}$ for any $W \subset \bar{S}$. We will now see that for $W = (\texttt{pa}(u) \cup \texttt{pa}(v)) \setminus S$ we have $u \perp v \mid S \cup W$. If $u \not\perp v \mid S \cup W$ then there exists an active path between u and v given $S \cup W$. Let $u - u' - \cdots - v' - v$ be this path. If we have $u \leftarrow u'$ or $v' \to v$ then the path would be inactive. We therefore have $u \to u' - \cdots - v' \leftarrow v$ and the path must contain at least one collider. Let c be a collider in this path, thus $c \in \texttt{anc}[S \cup W]$. If $c \in \texttt{anc}[S]$, since S is prefix, $c \in S$. In this case, the nodes on the path adjacent to c are also in S and the path must be inactive. By acyclicity, if there is a collider $c \in \texttt{anc}[W]$ in the path then there must be at least two colliders. Let c, c' be the leftmost and rightmost colliders on the path. We must have $c \in \texttt{anc}[\texttt{pa}(v)] \setminus S$ and $c' \in \texttt{anc}[\texttt{pa}(u)] \setminus S$ and the graph contains a cycle $c' \to \cdots \to u \to \cdots \to c \to \cdots \to v \to c'$, a contradiction to $\mathcal{G}$ being a DAG. □

We now provide a result characterizing specific nodes in D_S^m.

Lemma 17. *Let S be a prefix set. If $w \in D_S^m$ and $\texttt{anc}(w) \cap D_S^m = \varnothing$, then w must serve as the collider in a v-structure $a \to w \leftarrow b$.*

Proof. Since $w \in D_S^m$, there exists $u \in V$, $v \in \bar{S}$ and $W \subset \bar{S}$ with $|W| = m$ such that $u \perp v \mid S \cup W$ and $u \not\perp v \mid S \cup W \cup \{w\}$. Additionally, the active path between u and v given $S \cup W \cup \{w\}$ must contain a v-structure $a \to c \leftarrow b$ with $c \in \texttt{anc}[w] \cap D_S^m$. But $\texttt{anc}(w) \cap D_S^m = \varnothing$ and therefore $w = c$. Thus, $a \to w \leftarrow b$ is a v-structure. □

We then establish a relationship between the set F_S^m and undirected cliques, which follows from the observation that F_S^m includes downstream nodes of edges oriented by Meek rules.

Lemma 18. *Let S be a prefix set. Let $m > 0$ and $V' \subset \bar{S}$ such that,*

1. *There does not exist a v-structure $a \to c \leftarrow b$ with $c, b \in V'$.*
2. *There is an edge between nodes in V' oriented by a Meek rule.*
3. *$S \cup V'$ is prefix.*
4. *$V' \cap (F_S^1 \cup \cdots \cup F_S^{m-1}) = \varnothing$.*

Then there exists an undirected clique of size larger or equal to m in the subgraph induced by V' in $\mathcal{E}(\mathcal{G})$.

Proof. Let $V' \subset \bar{S}$ satisfy conditions (1)–(4). By condition (2), there exists an edge $u \to v$ in $\mathcal{E}(\mathcal{G})$ with $u, v \in V'$ that is oriented by a Meek rule.

The application of a Meek rule to orient an edge $u \to v$ requires at least one existing oriented edge between nodes in $\texttt{anc}(v)$, as observed in Proposition 10. Thus, the orientation of any edge using a Meek rule depends on prior orientations within $\texttt{anc}(v)$. We can trace this chain of dependencies backward through ancestors. Since $\mathcal{G}$ is acyclic and finite, this dependency chain must terminate.

This termination guarantees the existence of an edge $x \to y$ between two nodes $x, y \in \bar{S}$ such that there are no prior oriented edges in $\mathcal{E}(\mathcal{G})$ between nodes in $\texttt{anc}(y) \cap \bar{S}$. Condition (3) implies $x, y \in V'$ and condition (1) requires this edge to be oriented by a Meek rule.

Thus, $x \to y$ is oriented by directed edges between nodes in S and nodes in V' or directed edges between nodes in S. We now determine which Meek rule must have oriented this edge. We proceed by eliminating rules 2, 3 and 4.

R2: Requires z such that $x \to z$ and $z \to y$. Since $x, z \in \texttt{anc}(y)$ and $x \to z$, we must have $z \in S$, but then, we must also have $x \in S$, a contradiction.

R3: Requires x, y to form a v-structure with another $z \in V$, a contradiction.

R4: Requires z, z' such that $z \to z'$, $z' \to y$, $z \sim x$, $z' \sim x$ and $z \not\sim y$. Since $z, z' \in \texttt{anc}(y)$ we must have $z \in S$. This together with $z \sim x$ and $z \not\sim y$ means we have $z \to x$, and therefore, $x \to y$ is also oriented by Meek rule 1.

Therefore, $x \to y$ is oriented by Meek R1. This implies there exists $z \in S$ such that $z \to x$ and $z \not\sim y$ in $\mathcal{E}(\mathcal{G})$.

By condition (4) we have $y \notin F_S^1 \cup \cdots \cup F_S^{m-1}$. Therefore, $z \not\perp y \mid S \cup W$ for any $W \subset \bar{S}$ such that $|W| < m$. Let $K = \texttt{pa}(y) \setminus S$. Since $z \in S$ we have $\texttt{pa}(z) \subseteq S$, and therefore, $z \perp y \mid S \cup K$. Thus, $|K| \geq m$.

By condition (1) all nodes in K must be adjacent. Finally, since we selected $x \to y$ such that there is no directed edge in $\mathcal{E}(\mathcal{G})$ between nodes in $\texttt{anc}(y) \cap \bar{S}$, then $K \subset \texttt{anc}(y) \cap \bar{S} \subset V'$ must form an undirected clique in the essential graph of at least size m. □

Corollary 19. *Let S be a prefix set. Let $m > 0$ and $V' \subset \bar{S}$ such that,*

1. *There does not exist a v-structure $a \to c \leftarrow b$ with $c, b \in V'$.*
2. *There exists a clique $W \subset V'$ of size larger or equal to m.*
3. *$S \cup V'$ is prefix.*
4. *$V' \cap (F_S^1 \cup \cdots \cup F_S^{m-1}) = \varnothing$.*

Then there exists an undirected clique of size larger or equal to m in the subgraph induced by V' in $\mathcal{E}(\mathcal{G})$.

Proof. If W is undirected in the essential graph, we are done.

Otherwise, if W is not undirected in the essential graph, we can apply a Meek rule to one of the edges in it. By Lemma 18 there exists an undirected clique of size larger or equal to m in the subgraph induced by V' in $\mathcal{E}(\mathcal{G})$. □

We now extend the characterization of minimal separators in chordal undirected graphs to DAGs. The proof of this extension utilizes the moral graph construction and the concept of separation in undirected graphs. For a DAG $\mathcal{G}$, its *moral graph*, denoted $\mathcal{G}^m$, is the undirected graph created by adding edges between parents sharing common children in $\mathcal{G}$ and then removing all arrowheads. In an undirected graph, a node set C *separates* two disjoint node sets A and B if all paths between A and B intersect C. The following result is a key component of our argument.

Proposition 20 (Proposition 3.25 in Lauritzen (1996))**.** *Let A, B and C be disjoints subsets of V. Then C d-separates A from B in $\mathcal{G}$ if and only if C separates A from B in $\mathcal{G}[\texttt{anc}[A \cup B \cup C]]^m$.*

Lemma 21. *Let $S \subset V$ be a prefix node set. If for some $u \in V$, $v \in \bar{S}$ and $W \subset \bar{S}$ we have $u \perp v \mid S \cup W$, $u \not\perp v \mid S \cup W'$ for all $W' \subset \bar{S}$ such that $|W'| < |W|$ and $(\mathtt{anc}[u] \cup \mathtt{anc}[v]) \setminus S$ contains no v-structures then W is a clique in $\mathcal{G}$.*

Proof. Let $\mathcal{H} = \mathcal{G}[\bar{S} \cup \{u\}]$. By Lemma 11 we have that $u \perp_{\mathcal{G}} v \mid S \cup W'$ for some $W' \subset \bar{S}$ if and only if $u \perp_{\mathcal{H}} v \mid W'$. By Lemma 12 we have that $W \subseteq \mathtt{anc}(u) \cup \mathtt{anc}(v)$ and therefore $\mathtt{anc}[\{u, v\} \cup W] = \mathtt{anc}[\{u, v\}]$. Let $\mathcal{F} = \mathcal{H}[\mathtt{anc}[\{u, v\}]]$. Applying Proposition 20 we have that for any $W' \subset \mathtt{anc}(\{u, v\})$, W' d-separates u and v in $\mathcal{H}$ if and only if W' separates u and v in $\mathcal{F}^m$. As $(\mathtt{anc}[u] \cup \mathtt{anc}[v]) \setminus S$ does not contain any v-structures and $\mathcal{F}^m$ does not have any added adjacencies, any cycle of of four or more nodes will have a chord. Therefore, $\mathcal{F}^m$ is a chordal graph. Since W is a minimal separating set in a chordal graph, by Theorem 1 in Dirac (1961) it is a clique in $\mathcal{F}^m$. Since $\mathcal{F}^m$ has the same adjacencies as $\mathcal{F}$, W forms a clique in $\mathcal{F}$ and also in $\mathcal{G}$, because $\mathcal{F}$ is a subgraph of $\mathcal{G}$. □

The following result establishes that if a node belongs to D_S^m, then, a clique must exist among its ancestors.

Lemma 22. *Let S be a prefix set. If $w \notin D_S^0 \cup \cdots \cup D_S^{m-1}$ and $w \in D_S^m$ then, there exists a set of nodes $W \subset \mathtt{anc}(w) \cap \bar{S} \setminus D_S^m$ of size larger or equal to m that forms a clique in $\mathcal{G}$.*

Proof. Let c be a node in $\mathtt{anc}[w] \cap \bar{S}$ that acts as a collider in a v-structure $a \to c \leftarrow b$, with $a \in V$, $b \in \bar{S}$, chosen such that no node in $\mathtt{anc}(c)$ also serves as such a collider. This node must exist due to $\mathcal{G}$ being acyclic and having a finite amount of nodes.

From Lemma 16, we know there exists a set $W \subset \bar{S}$ that separates a and b given S, that is, $a \perp b \mid S \cup W$. By the selection of c, the set $\mathtt{anc}(c) \cap \bar{S}$ contains no v-structures. As $a, b \in \mathtt{anc}(c)$ it follows that $(\mathtt{anc}[a] \cup \mathtt{anc}[b]) \setminus S$ contains no v-structures. Then, by Lemma 21 the smallest set W that satisfies $a \perp b \mid S \cup W$ must form a clique.

Now, because $a \to c \leftarrow b$ forms a v-structure, conditioning on the collider c induces a dependence. Specifically $a \not\perp b \mid S \cup W \cup \{c\}$ holds, where W is the smallest separating set identified previously. Furthermore, we are given $w \notin D_S^0 \cup \cdots \cup D_S^{m-1}$, and by Lemma 7, it follows that $c \notin D_S^0 \cup \cdots \cup D_S^{m-1}$. This condition on c, implies that $|W| \geq m$. □

Finally, we establish the guarantees of Algorithm 1, as stated in Theorem 1, through the sequence of lemmas presented below.

Lemma 23. *In Algorithm 1, if at any iteration the working graph $\mathcal{E}$ contains a clique of size larger or equal to ℓ in V' then $\mathcal{E}(\mathcal{G})$ restricted to the nodes in V' also contains an undirected clique of size larger or equal to ℓ.*

Proof. We will prove that $\mathcal{E}(\mathcal{G})$ restricted to V' contains an undirected clique of size larger or equal to ℓ. By Theorem 6, $S \cup V'$ is a prefix set.

If V' contains a v-structure, let $a \to c \leftarrow b$ a v-structure such that $c, b \in V'$, and such that there is no v-structure $a' \to c' \leftarrow b'$ with $c', b' \in \mathtt{anc}(c) \cap \bar{S}$. Note that $|V'|$ is bounded and therefore this v-structure must exist. By Lemma 16 we have $c \in D_S^k$ for some $k \geq \ell$. By Lemma 22 there exists a clique W of size larger or equal to m in $\mathtt{anc}(c) \cap \bar{S}$. We also have that $S \cup (\mathtt{anc}(c) \cap \bar{S})$ is prefix and since $c \notin D_S^0 \cup \cdots \cup D_S^{\ell-1} \cup F_S^1 \cup \cdots \cup F_S^{\ell-1}$, by Lemma 8 we have $(\mathtt{anc}(c) \cap \bar{S}) \cap (F_S^1 \cup \cdots \cup F_S^{\ell-1}) = \varnothing$. By Corollary 19 there exists an undirected clique in $\mathtt{anc}(c) \cap \bar{S}$.

If V' does not contain a v-structure we will prove by contradiction. Assume that there does not exist a clique of size larger or equal to ℓ in $\mathcal{G}[V']$. Then, we must have $u \sim v$ in $\mathcal{E}$ but not in $\mathcal{G}$ for some $u, v \in V'$. This means that there exists a subset $W \subset V'$, $|W| \geq \ell$ such that $u \perp v \mid S \cup W$ and is minimal. We have that $(\mathtt{anc}[u] \cup \mathtt{anc}[v]) \setminus S$ does not contain v-structures and by Lemma 21, it must be a clique in $\mathcal{G}$. We have $|W| \geq \ell$ and by Corollary 19 we will have an undirected clique in the essential graph restricted to V'. □

Lemma 24. *Let $S_1, \ldots, S_m$ be the partition on V created by Algorithm 1 and let $u \to c \leftarrow v$ be a v-structure in $\mathcal{G}$. Then, $u \in S_i$, $v \in S_j$ and $c \in S_k$ with $k > i, j$.*

Proof. Without loss of generality, assume $j \geq i$. Since $S_1 \cup \cdots \cup S_j$ is prefix by Theorem 6, we must have $k \geq j$. If not $k > j$ then $k = j$. Let $S = S_1 \cup \cdots \cup S_{j-1}$. By Lemma 16 we have $u \perp v \mid S \cup W$ and $u \not\perp v \mid S \cup W \cup \{c\}$ for some $W \subset \bar{S}$. Let $W' \subset \bar{S}$ be the set with the least amount of nodes that satisfies this and let $k = |W'|$. We

have that $c \in D_S^k$ and $c \notin D_S^1 \cup \cdots \cup D_S^{k-1}$. By Lemma 22 there exists a clique of size larger or equal to k in $\texttt{anc}(c) \cap \bar{S} \subseteq S_j$, meaning, D_S^k must have been computed, a contradiction to $c \in S_j$. □

Lemma 25. *The graph created by Algorithm 1 has the same skeleton as $\mathcal{G}$.*

Proof. We will prove by contradiction. Let $\mathcal{E}$ denote the graph created by Algorithm 1 and assume that $\mathcal{E}$ and $\mathcal{G}$ do not have the same skeleton.

Let $u, v \in V$. If $u \sim v$ in $\mathcal{G}$ we have $u \not\perp v \mid W$ for all $W \subset V$. Therefore, if $\text{sk}(\mathcal{G}) \neq \text{sk}(\mathcal{E})$, there must exist $u \in S_i$ and $v \in S_j$ such that $u \sim v$ in $\mathcal{E}$ but not in $\mathcal{G}$.

Denote $S = S_1 \cup \cdots \cup S_{j-1}$. There must exist a set $W \subset S_j$ such that $u \perp v \mid S \cup W$ and is minimal. By Lemma 24, $(\texttt{anc}[u] \cup \texttt{anc}[v]) \setminus S$ does not contain any v-structures and by Lemma 21 we have that W forms a clique. This contradicts Algorithm 1 not removing this edge. □

Lemma 26. *Let $S_1, \ldots, S_m$ be the partition on V created by Algorithm 1. For all $u, v \in V$ if $u \to v$ in $\mathcal{E}(\mathcal{G})$ then $u \in S_i$ and $v \in S_j$ such that $i < j$.*

Proof. We prove this by contradiction. Assume there exists a directed edge $u \to v$ in $\mathcal{E}(\mathcal{G})$ such that both u and v belong to the same component S_k.

By Lemma 24, for any v-structure $a \to c \leftarrow b$ in $\mathcal{G}$, we will have $a \in S_i, b \in S_j, c \in S_k$ with $k > i, j$. This implies that if we have $u \to v$ in $\mathcal{E}(\mathcal{G})$ and u, v belong to the same component, the edge must have been oriented by a Meek rule.

Let $S = S_1 \cup \cdots \cup S_{k-1}$ and let s' denote the largest clique in $\mathcal{E}(\mathcal{G})[S_k]$. By Theorem 6, S is a prefix set. By Lemma 24, there does not exist a v-structure $a \to c \leftarrow b$ with $c, b \in S_k$. By definition, we have $S_k \cap (D_S^0 \cup \cdots \cup D_S^{s'} \cup F_S^1 \cup \cdots \cup F_S^{s'}) = \varnothing$ and by Lemma 18, there must exist a clique in $\mathcal{E}(\mathcal{G})[S_k]$ of size strictly larger than s', a contradiction. □

Theorem 1. *Given infinite samples from a distribution respecting a DAG $\mathcal{G}$, GAS (i.e., Algorithm 1) outputs $\mathcal{E}(\mathcal{G})$ using $p^{\mathcal{O}(s)}$ CI tests. Here, s is the size of the maximum undirected clique in $\mathcal{E}(\mathcal{G})$.*

Proof. We will show that Algorithm 1 outputs the essential graph of $\mathcal{G}$ and that it performs, at most, $p^{\mathcal{O}(s)}$ CI tests.

Let $\mathcal{E}$ denote the output of Algorithm 1. By Lemma 25, v and w are adjacent in $\mathcal{E}$ if and only if v and w are adjacent in $\mathcal{G}$. We will now prove that if $v \to w$ in $\mathcal{E}$ then also $v \to w$ in $\mathcal{G}$. If $v \to w$ in $\mathcal{E}$ we must have $v \in S_i$ and $w \in S_j$ for some $i < j$. By Theorem 6, $S = S_1 \cup \cdots \cup S_i$ is prefix, therefore, we have $v \to w$ in $\mathcal{G}$. Finally, using Lemma 26, we have $v \to w$ in $\mathcal{E}$ if and only if we have $v \to w$ in $\mathcal{E}(\mathcal{G})$. Therefore, $\mathcal{E} = \mathcal{E}(\mathcal{G})$.

By Lemma 23 the algorithm will calculate sets D_S^m and F_S^m of up to $m \leq s$ where s is the largest undirected clique in the essential graph of $\mathcal{G}$. By Theorem 6 an iteration requires at most $\mathcal{O}(p^{s+3})$ CI tests. Thus, the algorithm uses $\mathcal{O}(p^{s+4})$ CI tests, proving Theorem 1. □

C STEP-BY-STEP EXAMPLE OF ALGORITHM 1

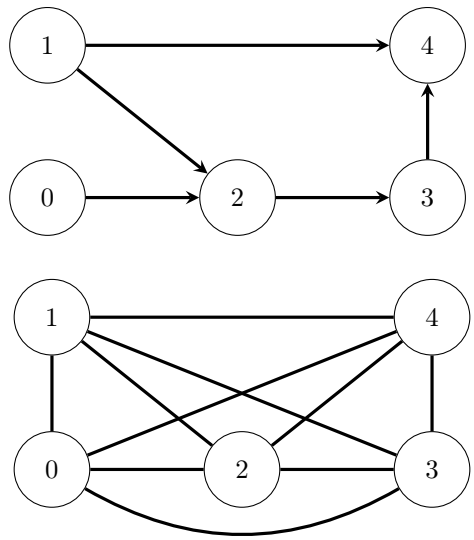
We now provide a step-by-step example of Algorithm 1.

Example 27. Suppose we have a probability distribution $\mathbb{P}$ that is Markov and faithful to the graph below. In this case, the graph and the essential graph are equal.

Step 0 (Initialization, Lines 1–3). We initialize an empty prefix node set $S = \varnothing$, an empty list $\mathcal{S}$, and a complete undirected graph $\mathcal{E}$.

Step 1 (Prefix node set expansion, Lines 5–21). We now learn the partial ordering of the essential graph through D_S^m and F_S^m, while establishing adjacencies.

The expansion step is repeated as long as the prefix node set does not contain all nodes ($S \neq V$). For each expansion, we work on a set of working nodes, denoted V'. This set excludes the prefix set and any identified

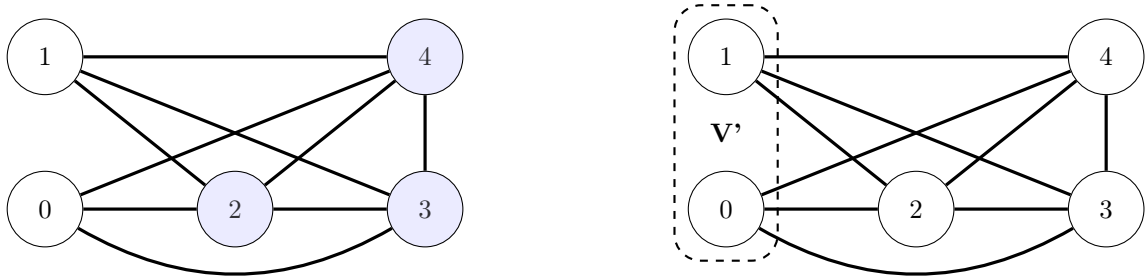


downstream nodes. In each expansion, we indicate the prefix node set S. The set of working nodes, V', is specified only when it differs from $\bar{S}$ (the set of nodes not in S).

For each ℓ, we remove edges between nodes if there exists a set $W \subseteq V'$ with $|W| = \ell$ that renders them independent. Nodes identified as being in D_S^ℓ are colored blue, while nodes in F_S^ℓ are colored green.

- First prefix node set expansion ($S = \varnothing, \mathcal{S} = []$).

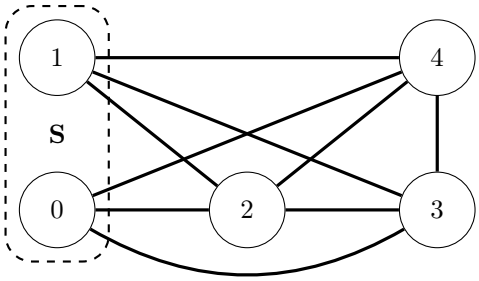
  Nodes 0 and 1 are found to be conditionally independent given the empty set, so the edge between them is removed. Furthermore, we find that conditioning on node 2 makes 0 and 1 dependent (unblocking the v-structure $0 \to 2 \leftarrow 1$). Similarly, conditioning on 3 or 4 (descendants of the collider) also makes 0 and 1 dependent. Therefore, we identify $D_S^0 = \{2, 3, 4\}$. The set of working nodes is then updated for the next iteration: $V' = V' \setminus D_S^0 = \{0, 1, 2, 3, 4\} \setminus \{2, 3, 4\} = \{0, 1\}$. For $\ell = 1$ no additional adjacencies are removed. The expansion stops here since no clique of size 2 exists in the current working set $V' = \{0, 1\}$.



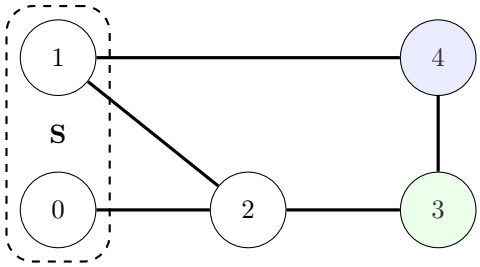
(a) $\ell = 0$. (b) $\ell = 1$.

- Second prefix node set expansion ($S = \{0, 1\}, \mathcal{S} = [\{0, 1\}]$).

  For $\ell = 0$, no new independencies are found. For $\ell = 1$, we remove several edges based on conditional independencies found. For instance, the edge $(0, 3)$ is removed because $0 \perp 3 \mid (S \setminus \{0\}) \cup \{2\}$. All other non-adjacencies in the true graph are also discovered and their corresponding edges removed during this phase. Additionally, we find $0 \not\perp 3 \mid S \setminus \{0\}$ and $0 \perp 3 \mid (S \setminus \{0\}) \cup \{2\}$, which identifies $3 \in F_S^1$. Similarly, $1 \perp 3 \mid (S \setminus \{1\}) \cup \{2\}$ and $1 \not\perp 3 \mid (S \setminus \{1\}) \cup \{2, 4\}$ identifies $4 \in D_S^1$. The working set is updated to $V' = V' \setminus (F_S^1 \cup D_S^1) = \{2, 3, 4\} \setminus (\{3\} \cup \{4\}) = \{2\}$. Finally, since V' contains no clique of size 2, this expansion stops here.

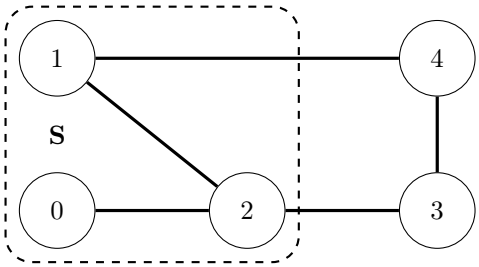


(a) $\ell = 0$.

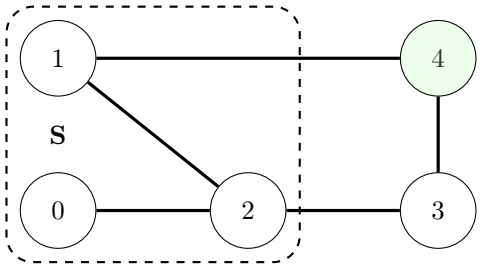


(b) $\ell = 1$.

- Third prefix node set expansion ($S = \{0, 1, 2\}, \mathcal{S} = [\{0, 1\}, \{2\}]$).
  We find $2 \not\perp 4 \mid S \setminus \{2\}$ and $2 \perp 4 \mid (S \setminus \{2\}) \cup \{3\}$, so $4 \in F_S^1$.

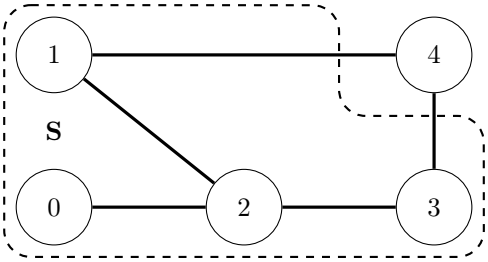


(a) $\ell = 0$.

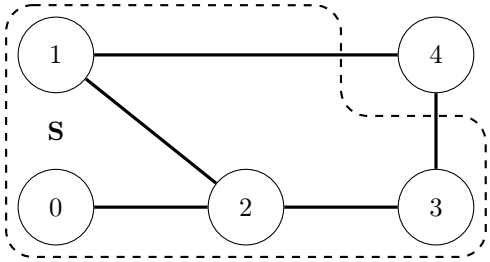


(b) $\ell = 1$.

- Fourth prefix node set expansion ($S = \{0, 1, 2, 3\}, \mathcal{S} = [\{0, 1\}, \{2\}, \{3\}]$).
  Only one node remains outside of S. No adjacencies are removed.

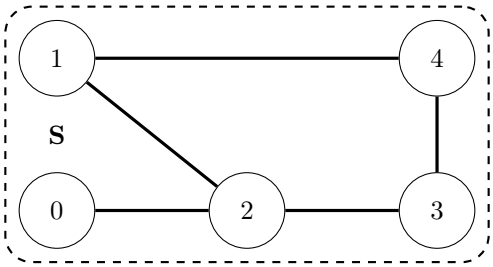


(a) $\ell = 0$.

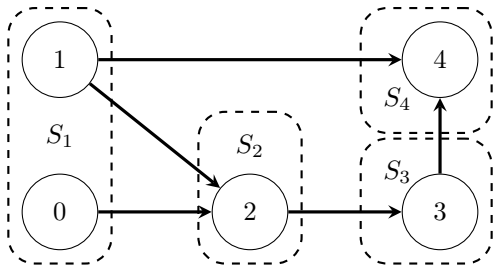


(b) $\ell = 1$.

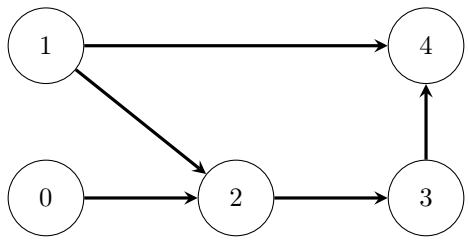
- End of prefix node set expansion ($S = \{0, 1, 2, 3, 4\}, \mathcal{S} = [\{0, 1\}, \{2\}, \{3\}, \{4\}]$).



Step 2 (Edge replacement, Lines 22–23). Since all remaining edges connect nodes in different components of the partition $\mathcal{S}$, we orient them from components with smaller indices to those with larger indices.



Step 3 (End, Line 24). The essential graph is now fully recovered.



D EXAMPLE OF CORRECTNESS WITHOUT FAITHFULNESS

The following example serves to prove Proposition 3.

Example 28. Suppose we have a probability distribution $\mathbb{P}$ modeled by the graph $\mathcal{G}$ in Figure 12 and such that the random variables X_i are related to each other by the following structural equations

$$X_j = \sum_{i<j} a_{ij} X_i + \varepsilon_j \tag{3}$$

where $\varepsilon_j \sim \mathcal{N}(0,1)$ and $a_{ij} \in [-1,1]$ are parameters with $a_{ij} \neq 0$ if and only if X_i causes X_j.

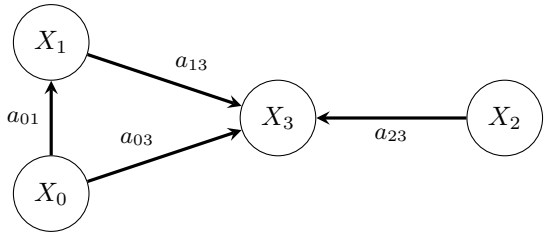


Figure 12: An example demonstrating how Algorithm 1 can recover the correct causal graph when the generating probability distribution is Markov but not faithful to that graph.

According to Proposition 4.1. in Uhler et al. (2013), the condition:

$$a_{03}a_{13} - a_{01} = 0 \tag{4}$$

holds if and only if the conditional independence $X_0 \perp X_1 \mid \{X_2, X_3\}$ is induced by $\mathbb{P}$. Therefore, if Equation (4) is satisfied, this constitutes a faithfulness violation.

By selecting a_{ij} such that this is the only faithfulness violation present, Algorithm 1 successfully recovers the true essential graph $\mathcal{E}(\mathcal{G})$. This is due to the algorithm identifying X_3 as a downstream node very early. Specifically, for $S = \varnothing$, the algorithm finds that X_1 and X_2 are independent, but become dependent when conditioning on X_3. This reveals the v-structure $X_1 \to X_3 \leftarrow X_2$ and places X_3 in D_S^0. Because X_3 is identified as downstream, it is excluded from certain future conditioning sets, meaning the test $X_0 \perp X_1 \mid \{X_2, X_3\}$—which represents the only faithfulness violation—is never performed by Algorithm 1.

E IMPLEMENTATION DETAILS

All graph manipulations required by Algorithm 1 are implemented using the NetworkX Python library (Hagberg et al., 2008), available under the 3-clause BSD license.

We now provide the specific implementations of the steps in Algorithm 1.

Algorithm 2 Algorithm used to remove edges from $\mathcal{E}$ in Algorithm 1

Input: CI queries from $\mathbb{P}$, graph $\mathcal{E}$, prefix set S, nodes V', condition set size m.
Output: $\mathcal{E}$, sepset.

```
for u ∈ V, v ∈ V' such that u − v in E do
    repeat
        Choose a new set W ⊆ V' \ {u, v} with |W| = m;
        if u ⊥ v | S ∪ W then
            Remove u − v from E;
            sepset(u, v) ← W;
    until u and v are no longer adjacent or all sets W have been considered
return E, sepset
```

Algorithm 3 Algorithm used to calculate D_S^m

Input: CI queries from $\mathbb{P}$, graph $\mathcal{E}$, prefix set S, nodes V', sepset from Algorithm 2.
Output: D_S^m.

```
D_S^m ← ∅;
P ← ∅;
for u ∈ V, v ∈ V' such that |sepset(u, v) \ S| = m do          ▷ Identify v-structures
    for w in adj(E, u) ∩ adj(E, v) ∩ V' do
        if w not in sepset(u, v) then
            D_S^m ← D_S^m ∪ {w};
            P ← P ∪ {{u, v}};
for {u, v} in P do                                              ▷ Identify descendants of the v-structures
    for w in V' \ D_S^m do
        if u ⊥̸ v | S ∪ sepset(u, v) ∪ {w} then
            D_S^m ← D_S^m ∪ {w};
return D_S^m
```

Algorithm 4 Algorithm used to calculate F_S^m

Input: CI queries from $\mathbb{P}$, prefix set S, nodes V', sepset from Algorithm 2.
Output: F_S^m.

```
F_S^m ← ∅;
for u ∈ S, v ∈ V' such that sepset(u, v) > 0 do
    repeat
        Choose a new set W ⊆ V' \ {u, v} with |W| = m;
        if u ⊥ v | S ∪ W then
            F_S^m ← F_S^m ∪ {v};
    until v ∈ F_S^m or all sets W have been considered
return F_S^m
```

To solve the k-clique problem, we adapt the Bron-Kerbosch algorithm (Bron and Kerbosch, 1973), originally used for enumerating maximal cliques, by terminating the search as soon as a clique of size k is found.

To efficiently compute the sets D_S^m and F_S^m, we optimize the process by removing adjacencies once a separating set is found (see Algorithm 2) and storing these sets.

The calculation of D_S^m (detailed in Algorithm 3) is a two-stage process. First, it identifies initial v-structures

by iterating through non-adjacent node pairs (u, v) and checking if a common neighbor w is absent from their separating set, reusing the results from Algorithm 2. Second, it finds all descendants of these v-structures by performing targeted additional CI tests to see which nodes w, when added to a conditioning set, break a known independence. Lemma 17 guarantees that that the set D_S^m consists exclusively of colliders within v-structures and their descendants.

The calculation of F_S^m (Algorithm 4) identifies the desired nodes by leveraging the previously identified separating sets. The definition for $w \in F_S^m$ requires two conditions: (1) $u \not\perp w \mid S$ for some $u \in S$, and (2) $u \perp w \mid S \cup W$ for some $W \subseteq \bar{S}$ of size m. The first condition is implicitly satisfied by Line 2. Therefore, simply finding a separating set W of size m for a pair (u, w) with $u \in S$ is sufficient to place w in F_S^m.

F ALGORITHM VARIANT WITH ADDITIONAL TESTS

We introduce a variant of Algorithm 1, which we call `GAS+`, that replaces the final orientation loop (Lines 22–23) with additional CI tests (see Algorithm 5). This variant returns a different graph only if faithfulness is violated. In our experiments (Figure 3), this variant acts as a normalized baseline for comparison.

Algorithm 5 Modification of the final loop in Algorithm 1 (Lines 22–23)

Create empty graph $\mathcal{D}$ on the node set V.
for S_i in $\mathcal{S} = [S_1, \ldots, S_m]$ **do**
  Add $v - w$ to $\mathcal{D}$ iff $v \not\perp w \mid S_1 \cup \cdots \cup S_i \setminus \{v, w\}$.
for $v \in S_i, w \in S_j$ with $i < j$ **do**
  Add $v \to w$ to $\mathcal{D}$ iff $v \not\perp w \mid S_1 \cup \cdots \cup S_j \setminus \{v, w\}$.
return $\mathcal{D}$

Next, we will show that under faithfulness, $\mathcal{D}$ is precisely $\mathcal{E}(\mathcal{G})$. We will start by proving that if we have $v - w$ in $\mathcal{D}$ then v and w will be adjacent in $\mathcal{G}$. If we have $v - w$ in $\mathcal{D}$, we must have $v, w \in S_i$ for some $S_i \in \mathcal{S}$. Denote $S = (S_1 \cup \cdots \cup S_i) \setminus \{v, w\}$. Since $v \not\perp w \mid S$ there exists an active path between v and w given S. Assume this path contains more than two nodes. We have that $S \cup \{v, w\}$ is prefix, therefore, there must exist a collider on the path. Any collider on the active path must also be in S, however, in this case, the adjacent nodes will also be in S and the path would be inactive. The only remaining case is if $v \sim c \sim w$ is a v-structure for some $c \in S$ but since $v, w \in S_i$ then $c \in S_i$ and by Lemma 24 we have a contradiction. Therefore, we have $v \sim w$.

If we have $v \to w$ in $\mathcal{D}$ we must have $v \in S_i$, $w \in S_j$ for some $S_i, S_j \in \mathcal{S}$ and $i < j$. Denote $S = (S_1 \cup \cdots \cup S_j) \setminus \{v, w\}$. Since $v \not\perp w \mid S$ there exists an active path between v and w given S. We have that $S \cup \{v, w\}$ is prefix, therefore, there must exist a collider on the path. Any collider in the active path must be in S, however, in this case, the adjacent nodes will also be in S and the path would be inactive. The only remaining case is if $v \to c \leftarrow w$, but that means $c \in S_j$. If not $v \sim w$ then $v \sim c \sim w$ is a v-structure, by Lemma 24 we have $c \in S_k$ with $k > j$, a contradiction. Since $S_1 \cup \cdots \cup S_i$ is prefix we must have $v \to w$.

By Lemma 26 there are no directed edges in the essential graph of $\mathcal{G}$ other than those in $\mathcal{D}$.

Finally, we will prove that if $v \sim w$ in $\mathcal{G}$ then $v \sim w$ in $\mathcal{D}$ and therefore we have all the adjacencies. If $v \sim w$ in $\mathcal{G}$ we will have $v \not\perp w \mid W$ for any $W \subset V$. With the intra and inter component tests we will also have $v \sim w$ in $\mathcal{D}$.

G EXPERIMENTAL SETUP

Hardware. All experiments were conducted on a single device with 1TB of RAM.

The conditional independence test employed across all compared algorithms was the partial correlation test (using Fisher's z-transform) at a significance level of $\alpha = 0.05$, unless otherwise specified. However, to ensure fair comparisons, we utilized different implementations of this test: results presented in Figures 3 are using the implementation from the `causaldag` package (Squires, 2018), while results in Figures 4, 5 and 13 are using the implementation provided by the `causal-learn` package (Zheng et al., 2024). In both cases, the default implementation of each tester and algorithm is used.

G.1 Linear Gaussian Synthetic Data

In these experiments, we simulate an observational setting. For each run, 10,000 data samples were generated from a linear Structural Causal Model (Pearl, 2009) with additive Gaussian noise, defined by a randomly generated underlying causal DAG $\mathcal{G}$. Edge weights for the SCM are drawn from $[-1, -0.25] \cup [0.25, 1]$, ensuring they are bounded away from 0.

G.2 SERGIO

The SERGIO simulation framework, developed by Dibaeinia and Sinha (2020), enables the generation of single-cell transcriptomics data that realistically reflects gene regulatory networks defined by the user.

For this, we use the DS1 gene regulatory network also introduced in Dibaeinia and Sinha (2020), which was designed based on known regulatory pathways in E. coli. This network comprises 100 genes and 258 edges. Key structural characteristics include 10 designated regulator genes, which are the only nodes with outgoing edges, and a maximum node out-degree of 76. Such a high maximum degree significantly impacts the runtime of constraint-based algorithms like `PC` and `FCI`, which become computationally prohibitive due to their search procedures. Using this DS1 GRN as the ground truth, we simulated gene expression datasets consisting of 2,700 cells. Finally, for the conditional independence tests used by `GAS` and `GAS+`, we set the significance level to $\alpha = 10^{-4}$.

G.3 Real-World Example

For our real-world application, we utilize the Airfoil Self-Noise NASA dataset, which was introduced by Brooks et al. (1989) and is available under the CC BY 4.0 license. For this experiment, we set the significance level to $\alpha = 10^{-4}$.

H ADDITIONAL EXPERIMENTS

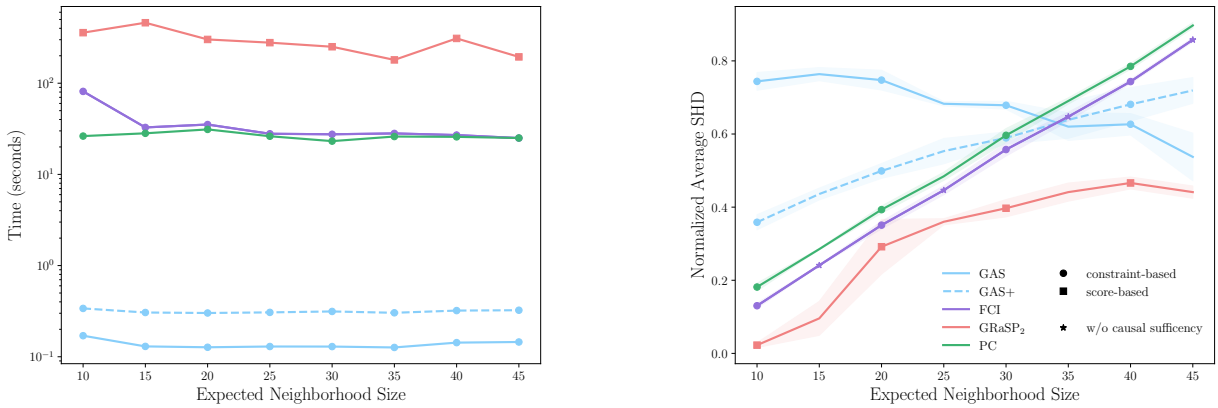
This section extends our empirical evaluation of `GAS` and `GAS+`. We analyze their performance across diverse graph structures, network sizes, and sample sizes to assess their robustness and scalability.

H.1 Increasing Neighborhood Size In Erdős-Rényi Graphs

Figure 13 shows the execution time and Structural Hamming Distance. As shown in Figure 14, `GAS` and `GAS+` significantly reduce the number of conditional independence tests compared to `PC`, often by more than an order of magnitude. Regarding accuracy, Figure 15 breaks down errors into extra and missing edges, while Table 1 provides complementary skeleton-level metrics, including precision, recall, and F1-score.

Table 1: Additional skeleton metrics for `GAS+` and `PC` across Erdős-Rényi graphs of 50 nodes and increasing expected neighborhood size.

Exp. Neigh. Size	Precision		Recall		F1-Score	
	GAS+	PC	GAS+	PC	GAS+	PC
15	0.42	**0.83**	**0.92**	0.29	**0.58**	0.43
25	0.57	**0.81**	**0.81**	0.16	**0.67**	0.27
35	0.73	**0.87**	**0.77**	0.12	**0.75**	0.21
45	0.93	**0.96**	**0.75**	0.09	**0.83**	0.17



(a) (b)

Figure 13: Comparison across Erdős-Rényi graphs on 50 nodes and increasing expected neighborhood size. Results are averaged over 3 runs. (a) Execution time (seconds) presented on a logarithmic scale. (b) Accuracy comparison measured by Structural Hamming Distance (SHD) between predicted and ground-truth graphs normalized by the number of possible edges. Shaded regions show the standard deviation.

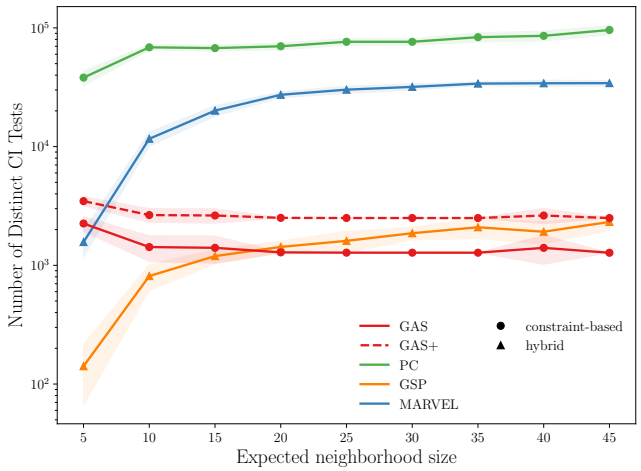
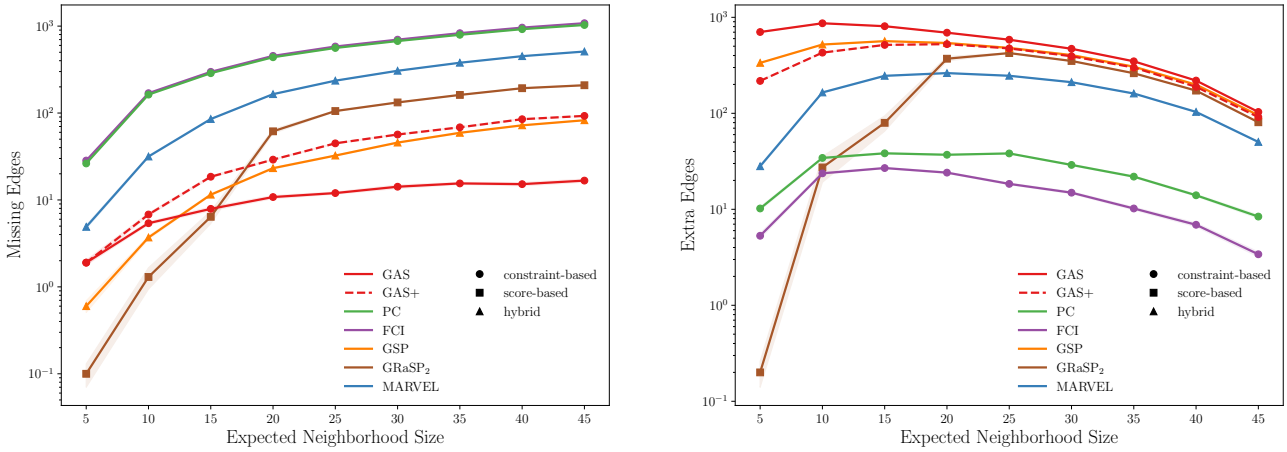


Figure 14: Number of distinct CI tests performed across Erdős-Rényi graphs of 50 nodes and increasing expected neighborhood size. Results are averaged over 10 runs.



(a) (b)

Figure 15: Comparison across Erdős-Rényi graphs on 50 nodes and increasing expected neighborhood size. Results are averaged over 10 runs. (a) Number of missing edges between predicted and ground-truth skeletons. (b) Number of extra edges between predicted and ground-truth skeletons. Shaded regions show the standard deviation.

H.2 Increasing Number Of Nodes In Erdős-Rényi Graphs

To evaluate scalability with respect to graph size, we conducted experiments on Erdős-Rényi graphs with an increasing number of nodes. Figure 16 reports the execution time and Structural Hamming Distance (SHD), while Figure 17 details the number of distinct conditional independence (CI) tests. It is important to note that for the experiments with a fixed expected neighborhood size (Figure 17a), the graphs become effectively sparser as the number of nodes increases. `GAS` and `GAS+` demonstrate superior scaling compared to the standard `PC` algorithm and `MARVEL` (Mokhtarian et al., 2025) when the edge probability is held constant (Figure 17b). While the hybrid algorithm `GSP` performs the fewest distinct CI tests, this lower test count does not strictly correspond to lower computational cost, as evidenced by the superior runtime scaling of `GAS` and `GAS+` in Figure 16.

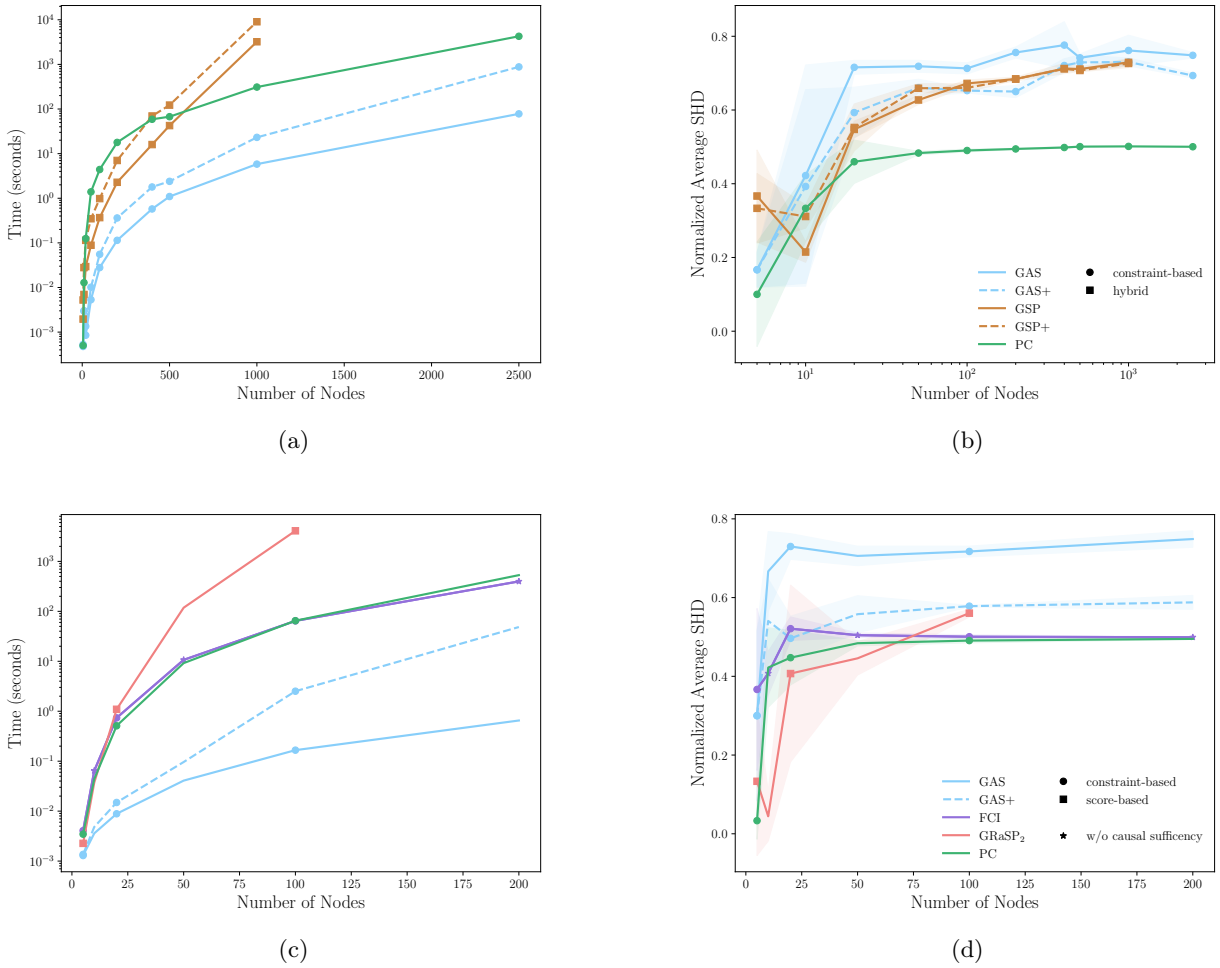
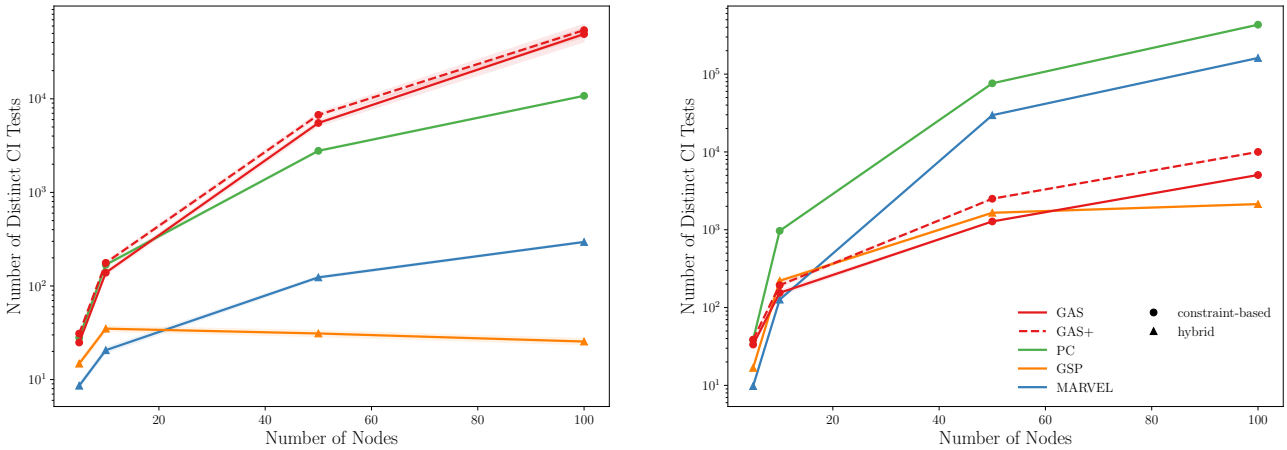


Figure 16: Comparison across Erdős-Rényi graphs of increasing number of nodes and edge probability $p = 0.5$. Results are averaged over 3 runs. (a), (c) Execution time (seconds) presented on a logarithmic scale. (b), (d) Accuracy comparison measured by the Structural Hamming Distance between predicted and ground-truth graphs normalized by the number of possible edges. Shaded regions show the standard deviation.

H.3 Scale-Free Graphs

Given that many real-world networks exhibit scale-free properties, we evaluate our algorithms on graphs generated using the Barabási–Albert (BA) model. Table 2 reports the Structural Hamming Distance and execution time for graphs with increasing connectivity, controlled by the BA model's parameter m.



(a) Expected neighborhood size 2

(b) Edge probability $p = 0.5$

Figure 17: Comparison of the number of distinct conditional independence tests across Erdős-Rényi graphs of increasing size. Results are averaged over 10 runs; shaded regions indicate the standard deviation.

Table 2: Comparison across Barabási–Albert graphs of 40 nodes and increasing parameter m. Results are averaged over 3 runs.

m	Exp. Neigh. Size	Structural Hamming Distance			Execution Time (s)		
		PC	GAS	GAS+	PC	GAS	GAS+
5	8.57	**105.00**	380.67	283.33	1.7576	**0.0377**	0.0432
10	14.29	**225.00**	426.00	289.67	1.4815	**0.0085**	0.0138
15	17.14	297.00	477.33	**296.33**	1.7844	**0.0074**	0.0192
20	17.14	305.00	450.00	**294.67**	3.3219	**0.0071**	0.0150

H.4 Varying Sample Size

Finally, we investigate the impact of sample size on the accuracy of the learned graph structures. We generated datasets of varying sizes from a 50-node Erdős-Rényi graph with an edge probability of $p = 0.5$. Table 3 shows the skeleton Structural Hamming Distance for each method.

Table 3: Skeleton Structural Hamming Distance between the predicted and ground-truth graphs for increasing sample sizes. Data was generated from 50-node Erdős-Rényi graphs with an edge probability of $p = 0.5$. Results are averaged over 3 runs.

Number of Samples	PC	GAS	GAS+
50	**568.33**	585.00	598.00
100	**567.00**	579.67	573.33
500	540.00	597.67	**514.33**
1000	554.33	596.33	**500.33**
5000	523.00	607.00	**492.33**
10000	538.00	604.00	**476.33**
100000	534.67	605.67	**508.33**