Don't Test What You Can Deduce: Causal Discovery with Logical Inference

Jonghwan Kim[1] **Sanghack Lee**[*1]

[1]Graduate School of Data Science, Seoul National University, South Korea
[*]Correspondence to: sanghack@snu.ac.kr

Abstract

Constraint-based causal discovery relies on conditional independence tests (CITs), which are highly unstable in high-dimensional settings. As the conditioning set grows, CITs suffer from low statistical power, leading to frequent false negatives. In the context of structure learning, this causes error propagation that degrades the accuracy of the estimated graph. We propose DF-PC (Deduce-First PC), a theoretically sound framework that integrates graphoid-based reasoning into the PC algorithm. Unlike prior approaches that primarily utilize deduction for conflict resolution or additive checks, DF-PC adopts a *proactive* "Deduce-First" strategy: it prioritizes logical deduction from strictly lower-order tests to preemptively replace high-order CITs. This "Deduce-First" strategy enables structure learning with potentially fewer CITs while improving performance. Empirical evaluations across various settings demonstrate that DF-PC achieves competitive learning performance and computational efficiency.

1 INTRODUCTION

Causal discovery is a fundamental challenge in machine learning and data science [Spirtes et al., 2000, Pearl, 2009, Cinelli et al., 2025]. Unlike standard statistical learning, which focuses on correlations and prediction, causal discovery aims to uncover the underlying data-generating mechanisms, enabling the estimation of intervention effects and counterfactual reasoning. Distinguishing between cause and effect is critical across high-stakes domains, from identifying genetic drivers of disease in computational biology to understanding policy impacts in economics. Consequently, developing efficient and reliable algorithms to recover causal graphs—typically represented as directed acyclic graphs—remains a central pursuit in the field.

Among various structure learning approaches, constraint-based methods stand as a dominant paradigm, with the seminal PC algorithm [Spirtes et al., 2000] serving as its foundation. Operating under the assumptions of causal Sufficiency, the causal Markov condition and faithfulness, constraint-based algorithms reconstruct the causal graph through a systematic sequence of conditional independence tests (CITs). The process generally begins with a complete undirected graph and iteratively removes edges based on independence findings, progressively increasing the size of the conditioning set. Theoretically, provided with an infinite sample size, the PC algorithm is guaranteed to recover the Markov equivalence class of the true causal graph.

However, in real-world scenarios characterized by finite samples, the assumption of perfect testing fails. CITs exhibit significant statistical instability, a phenomenon where the reliability of the test degrades as the complexity of the query increases. Specifically, as the learning proceeds and the size of the conditioning set grows, the effective sample size shrinks, drastically diminishing the statistical power of the tests. Consequently, high-order CITs are highly prone to errors, particularly Type II errors (i.e., failing to reject the null hypothesis of independence). These errors propagate through the learning process in complex ways, leading to the incorrect removal or retention of other edges [Spirtes et al., 2000, Cornia and Mooij, 2014, Malinsky, 2024].

Recognizing these vulnerabilities, recent research has focused on restricting the number of CITs to ensure the robustness of structure learning. One line of work, such as Anytime FCI [Spirtes, 2001] and subsequent works [Textor et al., 2015, Wienöbst and Liskiewicz, 2020, Kocaoglu, 2023], explicitly limits the execution of statistical tests and allows the algorithm to stop early. While this strategy successfully mitigates error propagation, it inherently restricts the output to incompletely specified partial graphs. In a different direction, other recent studies [Shiragur et al., 2024, Franquesa Monés, 2025] have proposed alternative algo-

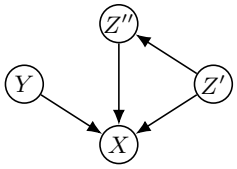

Figure 1: The ground truth causal graph [Kim et al., 2024]

rithms that try to learn the causal graph with fewer tests through more efficient search strategies. Despite their differences, existing approaches largely treat individual CITs as isolated sources of information, overlooking the potential to exploit the logical relationships between the tests themselves to reduce the statistical burden.

To address this challenge, we introduce **DF-PC (Deduce-First PC)**, a theoretically sound and practical algorithm that tightly integrates deductive reasoning into the constraint-based discovery framework. Our key insight is that CI statements are not mutually independent; they are logically constrained by the rules from graphoid axioms [Pearl and Paz, 1987]. Therefore, many high-order dependencies—the ones most prone to statistical instability—can be logically entailed by the results of *strictly lower-order* tests, which are deemed statistically much more reliable. Distinct from prior approaches that primarily utilize deduction for conflict resolution [Bromberg and Margaritis, 2009] or additive robustness checks [Kim et al., 2024], DF-PC adopts a *proactive* "Deduce-First" strategy. By preemptively inferring CI statements from established lower-order test results, DF-PC bypasses unreliable high-order CITs during structure learning. This mechanism not only significantly reduces the number of required CITs but also effectively mitigates the risk of error propagation.

Motivating example Consider the motivating example in Figure 1. In the initial phase (Level 0), the PC algorithm exhaustively tests all pairs for marginal independence, correctly establishing that $Y \perp\!\!\!\perp Z'$. However, as the algorithm proceeds to Level 1, a standard implementation naively tests the conditional independence of connected pairs, such as Y and X, conditioning on their neighbors (e.g., Z'). In contrast, one can observe that the previously established marginal independence $Y \perp\!\!\!\perp Z'$ logically constrains the relationship between Y and X. Specifically, if Y and Z' are independent, conditioning on Z' cannot render the dependent pair Y, X independent. Recognizing this entailment, we can skip the statistical test entirely. By prioritizing such logical deductions over estimation, this approach buffers the structure learning process against the noise inherent in high-order CI testing.

Contributions First, we propose the **DF-PC** algorithm, a theoretically sound and practical modification of PC that integrates deduction to reduce the number of CITs. Second, we provide theoretical results on the soundness of our method and bounds on the number of CITs required. Finally, we empirically demonstrate that DF-PC can achieve competitive performance in terms of both accuracy and computational efficiency.

2 PRELIMINARIES

Notations We consider a set of m random variables $\mathbf{V} = \{X_1, \ldots, X_m\}$. Following standard convention, we use capital letters (e.g., X) to denote individual variables, and lowercase letters (e.g., x) for their observed values. We use bold capital letters (e.g., $\mathbf{Z}$) to denote sets of variables, and bold lowercase letters (e.g., $\mathbf{z}$) to denote sets of values. We often highlight disjoint union by employing $\sqcup$ instead of generic set union $\cup$.

A Bayesian Network is defined as a pair $(\mathcal{G}, P)$, consisting of a Directed Acyclic Graph (DAG) $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ and a joint probability distribution $P(\mathbf{V})$. We denote the set of adjacent nodes of a node X as $\mathrm{adj}(X)$. We represent a conditional independence (CI) query as $CI(X; Y \mid \mathbf{Z})$ and denote its outcome by $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ for independence and $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$ for dependence.

Assumptions Throughout this work, we operate under three standard assumptions from the causal discovery literature [Spirtes et al., 2000]. First, we assume the *causal Markov condition*, under which every conditional independence implied by d-separation in $\mathcal{G}$ holds in the joint distribution P; that is, for any disjoint subsets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subset \mathbf{V}$, $(\mathbf{X} \perp\!\!\!\perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp_{P} \mathbf{Y} \mid \mathbf{Z})$. Second, we assume *faithfulness*, meaning that P entails no additional independencies beyond those implied by $\mathcal{G}$, establishing the equivalence $(\mathbf{X} \perp\!\!\!\perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z}) \iff (\mathbf{X} \perp\!\!\!\perp_{P} \mathbf{Y} \mid \mathbf{Z})$. Third, we assume *causal sufficiency*, i.e., there are no latent common causes among the variables in $\mathbf{V}$.

Rules from Graphoid Axioms and Faithfulness In graphical models, CI relations among random variables do not exist in isolation; rather, they are governed by a set of fundamental logical rules. These rules dictate how certain independencies naturally imply others, providing a foundational algebraic framework for understanding how CI properties interact within a given probability distribution.

The independence model of any arbitrary probability distribution inherently satisfies four foundational axioms: Symmetry, Decomposition, Weak Union, and Contraction. A model satisfying these basic rules is formally called a *semi-graphoid* [Pearl and Paz, 1987]. If the distribution is strictly positive, an additional rule called Intersection also holds, elevating the model to a *graphoid* [Pearl and Paz, 1987, Lauritzen, 1996]. Although these axioms provide a sound logical basis, they are mathematically incomplete [Studený, 1992], meaning that there exists no finite complete charac-

Algorithm 1 DEDUCE-DEP [Kim et al., 2024]

Input: $\{X\}, \{Y\}, \mathbf{Z}$ disjoint subsets of $\mathbf{V}$, reliability threshold K (default 1)
Output: Whether $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$ is deducible.
if $|\mathbf{Z}| \leq K$ **return** FALSE
for $Z \in \mathbf{Z}$
  $\mathbf{Z}' \leftarrow \mathbf{Z} \setminus \{Z\}$
  for $(A, B, \mathbf{C})$ in $\{(X, Y, \mathbf{Z}'), (X, Z, \mathbf{Z}'), (Y, Z, \mathbf{Z}')\}$
    if $(A \perp\!\!\!\perp B \mid \mathbf{C})$ **and not** DEDUCE-DEP$(A, B, \mathbf{C})$
      mark $(A; B \mid \mathbf{C})$ as ind
    else mark $(A; B \mid \mathbf{C})$ as dep
  if $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}') \oplus \big((X \not\perp\!\!\!\perp Z \mid \mathbf{Z}') \wedge (Y \not\perp\!\!\!\perp Z \mid \mathbf{Z}')\big)$
    return TRUE
return FALSE

terization for CI relations across all possible distributions.

Under the *faithfulness* assumption, the independence model satisfies additional properties. As established in the literature [Bromberg and Margaritis, 2009, Sadeghi, 2017], faithfulness promotes the model to a *compositional graphoid* (which additionally satisfies the Composition rule) and endows it with *weak transitivity*. Consequently, the CI relations are governed by a set of rules detailed in Appendix A.

3 REVISITING GRAPHOID-BASED REASONING

Under faithfulness, Kim et al. [2024] derive rules that infer higher-order dependence from lower-order (in)dependence findings. We summarize their main results below.

Proposition 1 (Proposition 1 in Kim et al. [2024])**.** *Given a faithful Bayesian network $(\mathcal{G}, P)$, let $\{X\}, \{Y\}$ and $\mathbf{Z}$ be disjoint subsets of $\mathbf{V}$ and partition $\mathbf{Z} = \mathbf{Z}' \sqcup \{Z''\}$. Then $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$ holds if one of the following is true:*

1. $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}') \wedge (X \perp\!\!\!\perp Z'' \mid \mathbf{Z}')$,
2. $(X \perp\!\!\!\perp Y \mid \mathbf{Z}') \wedge (X \not\perp\!\!\!\perp Z'' \mid \mathbf{Z}') \wedge (Y \not\perp\!\!\!\perp Z'' \mid \mathbf{Z}')$.

By symmetry, the first condition may equivalently use $(Y \perp\!\!\!\perp Z'' \mid \mathbf{Z}')$ instead of $(X \perp\!\!\!\perp Z'' \mid \mathbf{Z}')$.

Corollary 1 (Corollary 1 in Kim et al. [2024])**.** *Under the faithful Bayesian network $(\mathcal{G}, P)$, let $\{X\} \sqcup \{Y\} \sqcup \mathbf{Z} \subseteq \mathbf{V}$, $Z \in \mathbf{Z}$, and $\mathbf{Z}' = \mathbf{Z} \setminus \{Z\}$. If*

$$(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}') \oplus \big((X \not\perp\!\!\!\perp Z \mid \mathbf{Z}') \wedge (Y \not\perp\!\!\!\perp Z \mid \mathbf{Z}')\big),$$

then $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$ holds.

Based on these results, their algorithm DEDUCE-DEP (Alg. 1) performs deduction of *dependence* with additional CITs as needed, providing a robustness check on independence findings during structure learning (e.g., PC).

3.1 CHARACTERIZING QUERY HISTORY IN PC-LIKE ALGORITHMS

To understand the practical implications of integrating DEDUCE-DEP into a constraint-based causal discovery framework like the PC algorithm, we first analyze their interaction. Specifically, our goal is to clarify which conditional independence test results are already available to DEDUCE-DEP by the time a query $CI(X; Y \mid \mathbf{Z})$ is evaluated, and consequently to characterize the number of "additional" CI tests—beyond those performed by PC itself—that DEDUCE-DEP may require. We begin by formalizing the structural properties of algorithms that share the systematic query schedule of PC.

Definition 1 (PC-like algorithm)**.** A constraint-based algorithm is classified as *PC-like* if it features a sound and complete skeleton learning phase before edge orientation, starting from a complete undirected graph, and obeys the following properties:

1. **Level-wise Monotonicity**: Primary CI queries are processed in nondecreasing conditioning-set size. For an edge that survives a level, all eligible conditioning sets at that level are examined.
2. **Local Adjacency Constraint**: For any ordered pair of variables (X, Y), the conditioning set $\mathbf{S}$ evaluated for the query $CI(X; Y \mid \mathbf{S})$ satisfies $\mathbf{S} \subseteq \mathrm{adj}(X)_{\mathcal{G}'} \setminus \{Y\}$, where $\mathcal{G}'$ denotes the partially undirected skeleton at the exact moment the query is invoked.

Examples include PC [Spirtes et al., 2000], FCI [Spirtes et al., 2000], CPC [Ramsey et al., 2006], PC-stable [Colombo and Maathuis, 2014], KPC [Kocaoglu, 2023], PC-Guess [Hiremath et al., 2025], and other constraint-based skeleton learning algorithms.

Given this definition, we formalize the *query history* as the set of CI outcomes already known when a PC-like algorithm examines a query $CI(X; Y \mid \mathbf{Z})$. Completing the lower levels does not mean evaluating every possible lower-order query, because early stopping and edge removal can skip queries.

Proposition 2 (Query history of a PC-like algorithm)**.** *Let X and Y be distinct variables, and let $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$ be a nonempty set of variables. If a PC-like algorithm examines the CI query $CI(X; Y \mid \mathbf{Z})$ with $|\mathbf{Z}| \geq 1$, then at the moment of this evaluation, the algorithm's query history contains:*

1. *the results of all pairwise CI queries $CI(U; V \mid \emptyset)$ for all distinct variables U, V; and*
2. *the results of all CI queries $CI(X; T \mid \mathbf{S})$ yielding dependence $(X \not\perp\!\!\!\perp T \mid \mathbf{S})$, for every T and $\mathbf{S}$ satisfying $T \in \{Y\} \cup \mathbf{Z}$ and $\mathbf{S} \subseteq (\{Y\} \cup \mathbf{Z}) \setminus \{T\}$ s.t. $|\mathbf{S}| < |\mathbf{Z}|$.*

All omitted proofs are provided in Appendix B. Prop. 2 reveals a crucial property of PC-like algorithms when integrated with DEDUCE-DEP. To see this, suppose we partition the conditioning set as $\mathbf{Z} = \mathbf{Z}' \sqcup \{Z\}$. By applying the proposition to $T \in \{Y, Z\}$ and $\mathbf{S} = \mathbf{Z}'$, we find that by the time the algorithm is scheduled to evaluate $CI(X; Y \mid \mathbf{Z}' \sqcup \{Z\})$, it has already established the dependence outcomes:

$$(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}') \qquad \text{and} \qquad (X \not\perp\!\!\!\perp Z \mid \mathbf{Z}').$$

This intrinsic property allows DEDUCE-DEP to systematically reuse previously performed CIT results.

However, it is critical to note that these two CI statements alone do *not* logically complete the deductive rule.

The deductive rule used by DEDUCE-DEP requires knowledge of a third CI query: $CI(Y; Z \mid \mathbf{Z}')$. Whether the baseline PC-like schedule has already produced this specific result depends entirely on the adjacency structure of Y or Z at the earlier stage of the run. In particular, it is possible that $CI(Y; Z \mid \mathbf{Z}')$ has not been queried. In such cases, the deductive engine cannot proceed without requesting this additional CIT.

3.2 THE COMPLEXITY BOTTLENECK OF DEDUCE-DEP

While Prop. 2 demonstrates that some test history can be reused, the fact that the deductive engine must actively request additional CITs for missing historical queries introduces a severe computational bottleneck. To quantify this overhead, we first formalize a worst-case upper bound on the number of CITs required by DEDUCE-DEP to resolve a single query.

Proposition 3 (Worst-case upper bound on the total number of additional CITs required by DEDUCE-DEP)**.** *Suppose we run the PC algorithm and encounter a CI query $CI(X; Y \mid \mathbf{Z})$ with $|\mathbf{Z}| = k > 0$. Assume that we attempt to resolve this query using* DEDUCE-DEP*, with its reliability threshold set to zero. Then, in the worst case, the number of* new *CITs performed by* DEDUCE-DEP *is* $\frac{k}{8}\big(2^k(k+3) - 4(k+1)\big)$*, which is* $O(k^2 2^k)$.

Remark 1. The key idea is that, for a query with conditioning set size k, DEDUCE-DEP may recursively inspect lower-order conditioning sets. At level ℓ, there are $\binom{k}{\ell}$ possible conditioning sets, which gives rise to the 2^k-scale term when summed over ℓ. For each such conditioning set, the procedure may need to examine CI relations among

$$\binom{k+2-\ell}{2}$$

variable pairs, which contributes the quadratic factor in k. Thus, the total number of potentially required CITs is

$$T(k) = \sum_{\ell=0}^{k-1} \binom{k}{\ell} \binom{k+2-\ell}{2}.$$

Using Prop. 2, some of these CIT results are already available from the query history. After subtracting the reusable results $R(k)$, we obtain the number of newly required CITs.

While Prop. 3 bounds the overhead for a *single* query, this additive cost compounds significantly over the entire execution of the algorithm. Building on the preceding analyses, we now establish a worst-case upper bound on the number of CITs required by the PC algorithm augmented with DEDUCE-DEP.

Proposition 4 (Worst-case upper bound on the total number of CITs required by PC with DEDUCE-DEP)**.** *Let m denote the number of variables and Δ the maximum degree of the underlying graph. Under the oracle CI tester, the total number of CITs performed by the PC algorithm with* DEDUCE-DEP *in the worst case, is bounded by* $O\big(m^{\Delta+2} + m^2 \cdot \Delta^2 2^{\Delta}\big)$.

Remark 2. The first term, $O(m^{\Delta+2})$, corresponds to the standard worst-case upper bound on the number of CITs performed by PC-stable. The second term accounts for the additional deduction overhead. Since each independence finding removes one edge, there can be at most $O(m^2)$ such events over the full run. For each such event, Prop. 3 implies a worst-case deduction cost of $O(\Delta^2 2^{\Delta})$, where Δ is the maximum degree. Therefore, the total worst-case cost is bounded by

$$O\big(m^{\Delta+2} + m^2 \Delta^2 2^{\Delta}\big)\,.$$

This compounded computational overhead highlights a critical limitation of integrating logical deduction *additively* into constraint-based skeleton learning. To overcome this bottleneck, the deductive reasoning must be reconfigured from an additive confirmation step into a preemptive filter.

4 THE DEDUCE-FIRST PC ALGORITHM

We first discuss how to effectively exploit the query history to reduce redundant tests, and then present the concrete algorithm that implements this strategy, accompanied by theoretical soundness and complexity guarantees.

4.1 FROM TEST-FIRST TO DEDUCE-FIRST

As analyzed in the previous section, the query complexity of the DEDUCE-DEP algorithm [Kim et al., 2024] is significantly bounded when constrained by the query history of PC-like algorithms. However, a practical challenge remains: the integration of DEDUCE-DEP into PC has traditionally

been *additive*. In this naive configuration, the algorithm performs all standard PC tests *plus* the additional queries required by the deductive engine. Consequently, the total number of CITs increases monotonically compared to the baseline PC algorithm. As reported in [Kim et al., 2024], this often leads to a substantial increase in runtime, negating the benefits of improved robustness.

We identify that this inefficiency stems from a failure to fully exploit the structured *query history* inherent to constraint-based algorithms. PC proceeds level-wise, accumulating the lower-order CI results queried before the current level; early stopping and edge removal may leave other lower-order queries unevaluated. For instance, the complete set of marginal independence results (level 0) serves as a logical foundation that can theoretically entail numerous conditional independence relations at level 1 without requiring further statistical interaction.

To capitalize on this observation, we propose a paradigm shift from "Test-First" to **"Deduce-First."** We invert the control flow: before executing any high-order CIT, the algorithm first consults the deductive engine using the accumulated history of strictly lower-order results. A statistical test is triggered *only* if the query cannot be resolved through logical inference. This approach transforms deduction from an additive cost into a subtractive efficiency mechanism, offering the potential to skip a significant number of unstable high-order tests.

In order to maximize the coverage of this "Deduce-First" gatekeeper, we expand the scope of logical inference. While prior works primarily focused on rules for propagating *dependence* (e.g., for robustness check), effective structure learning relies heavily on identifying *independence* as well. Therefore, we incorporate deductive rules for conditional independence alongside the existing rules for deducing conditional dependence. The following proposition is a logical consequence of the Composition and Weak Union axioms under faithfulness.

Proposition 5. *Under the faithful Bayesian network* $(\mathcal{G}, P)$*, let* $\mathbf{X}$*,* $\mathbf{Y}$*, and* $\mathbf{Z}$ *be disjoint subsets of* $\mathbf{V}$ *where* $\mathbf{Z}$ *is partitioned into* $\mathbf{Z}'$ *and* $\mathbf{Z}''$ *such that* $\mathbf{Z} = \mathbf{Z}' \sqcup \mathbf{Z}''$, $|\mathbf{Z}''| = 1$*. Then,* $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ *if one of the following holds:*

1. $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}') \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Z}'' \mid \mathbf{Z}')$
2. $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}') \wedge (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}'' \mid \mathbf{Z}')$

Remark 3. At first glance, the condition $(X \perp\!\!\!\perp Y \mid \mathbf{Z}')$ in Prop. 5 might appear redundant for the primary edge query $CI(X; Y \mid \mathbf{Z})$, as the edge (X, Y) would typically have been removed in previous steps of the PC if this lower-order independence held. However, this proposition plays a critical role in recursively deducing the outcome of a CI query from query history. When evaluating a high-order query involving a conditioning set $\mathbf{Z}$, the algorithm recursively verifies the independence of conditioning variables (e.g., determining if $Y \perp\!\!\!\perp Z \mid \mathbf{Z} \setminus \{Z\}$ for some $Z \in \mathbf{Z}$). Since the edge (Y, Z) may have been removed at a lower order (where $|\mathbf{Z} \setminus \{Z\}| < |\mathbf{Z}|$), Prop. 5 allows the algorithm to logically confirm this independence in the current context using cached results, without performing an additional CIT.

Algorithm 2 DF-PC: Deduce-First PC Algorithm

Input: a set of variables $\mathbf{V}$, CI tester
Output: a CPDAG
Initialize $\mathcal{G}$ with a complete undirected graph
Initialize shared CI cache $\mathcal{C} \leftarrow \emptyset$
for $k \in 0, 1, 2, \ldots, |\mathbf{V}| - 2$
  for an ordered adjacent pair $(X, Y) \in \mathcal{G}$ s.t. $|\text{adj}(X)_{\mathcal{G}}| > k$
    for $\mathbf{S} \subseteq \text{adj}(X)_{\mathcal{G}} \setminus \{Y\}$ s.t. $|\mathbf{S}| = k$
      Query DEDUCE$(X, Y, \mathbf{S})$ and record the result
      if $(X \perp\!\!\!\perp Y \mid \mathbf{S})$
        break
  for each recorded $(X \perp\!\!\!\perp Y \mid \mathbf{S})$
    Remove edge $X - Y$ from $\mathcal{G}$
Orient $\mathcal{G}$ for unshielded colliders
Complete orientation of $\mathcal{G}$ with Meek's rules
return $\mathcal{G}$

4.2 THE DF-PC ALGORITHM

By combining this expanded rule set with the "Deduce-First" strategy, we present the **Deduce-First PC (DF-PC)** algorithm (Alg. 2) that effectively exploits the query history for logical substitution of CITs. The design of DF-PC adopts the level-wise skeleton search of the PC-stable algorithm [Colombo and Maathuis, 2014] to ensure soundness. However, it fundamentally alters the mechanism of examining CI queries.

In standard PC, every query $CI(X; Y \mid \mathbf{Z})$ is immediately dispatched to a statistical CI tester. In contrast, DF-PC intercepts these queries and passes them through a deductive inference engine, DEDUCE (Alg. 3). This engine acts as a logical filter: it attempts to resolve the query by propagating the results of strictly lower-order tests using the recursive function DEDUCE-HELPER (Alg. 4). The physical CI tester is invoked *only* as a fallback measure when logical inference fails to provide a definitive answer.

DF-PC retains early stopping by default. In finite samples, skipped queries can change the lower-order history available for later deduction, so its skeleton can depend on query order despite level-wise adjacency updates.

Algorithm Walkthrough. Since DEDUCE (Alg. 3) is a wrapper function, we focus on how the DEDUCE-HELPER procedure (Alg. 4) operates to deduce the outcome of a CI query.

DEDUCE-HELPER begins by initializing a flag seenInd to FALSE (Line 3), which tracks whether an independence

Algorithm 3 DEDUCE

1: **Input:** $\{X\}, \{Y\}, \mathbf{Z}$ disjoint subsets of $\mathbf{V}$, CI tester
2: **Output:** the outcome of $CI(X;Y \mid \mathbf{Z})$
3: outcome $\leftarrow$ DEDUCE-HELPER$(X, Y, \mathbf{Z})$
4: **if** outcome $=$ UNKNOWN
5: **return** CIT$(X;Y \mid \mathbf{Z})$
6: **else**
7: **return** outcome

Algorithm 4 DEDUCE-HELPER

1: **Input:** $\{X\}, \{Y\}, \mathbf{Z}$ disjoint subsets of $\mathbf{V}$
2: **Output:** the outcome of $CI(X;Y \mid \mathbf{Z})$
3: seenInd $\leftarrow$ FALSE
4: **for** $Z \in \mathbf{Z}$
5: $\mathbf{Z}' \leftarrow \mathbf{Z} \setminus \{Z\}$
6: **for** $(A, B, \mathbf{C})$ in $\{(X, Y, \mathbf{Z}'), (X, Z, \mathbf{Z}'), (Y, Z, \mathbf{Z}')\}$
7: **if** $(A, B, \mathbf{C})$ is not cached
8: Query DEDUCE-HELPER$(A, B, \mathbf{C})$ & record the result
9: **if** any condition in Prop. 1 is satisfied
10: **return** DEP
11: **else if** any condition in Prop. 5 is satisfied
12: seenInd $\leftarrow$ TRUE
13: **if** seenInd $=$ TRUE
14: **return** IND
15: **else**
16: **return** UNKNOWN

relation has been logically derived. The algorithm then iterates through each variable Z in the conditioning set $\mathbf{Z}$ (Line 4) to attempt a reduction. For each Z, it defines a reduced set $\mathbf{Z}' = \mathbf{Z} \setminus \{Z\}$ (Line 5) and retrieves the status of the three component triplets required for logical inference: $(X, Y \mid \mathbf{Z}')$, $(X, Z \mid \mathbf{Z}')$, and $(Y, Z \mid \mathbf{Z}')$. If any of these results are missing from the cache, DEDUCE-HELPER is recursively called to resolve them (Lines 6–8).

Using these components, the algorithm first checks for *dependence* (Lines 9–10). If the condition in Prop. 1 holds, it implies logical dependence; thus, the function immediately returns DEP, short-circuiting further search. If dependence is not found, it checks for *independence* using the condition of Prop. 5 (Lines 11–12). If the condition holds, the seenInd flag is set to TRUE. Unlike dependence, finding an independence does not return immediately, as the algorithm prioritizes finding a conflict (dependence) first if one exists.

After checking all $Z \in \mathbf{Z}$, if the seenInd flag is TRUE, the algorithm returns IND (Lines 13–14), confirming that the independence is logically entailed. If neither dependence nor independence could be deduced after checking all decompositions, the algorithm returns UNKNOWN.

4.3 ILLUSTRATIVE EXAMPLE

To demonstrate how the DF-PC algorithm operates, we provide a detailed comparison of its execution against PC-stable

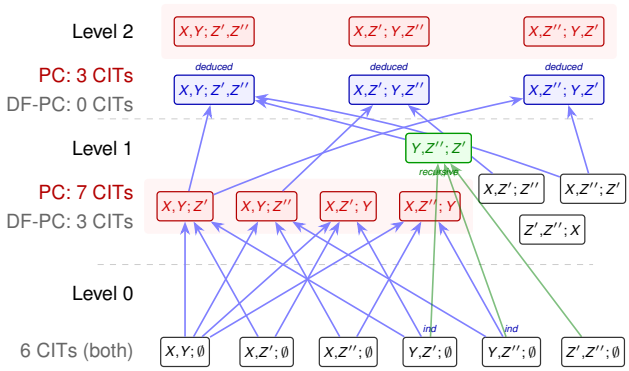

Figure 2: Hasse-like diagram of CI queries on the motivating example (Fig. 1), arranged by conditioning set size. Black nodes: CITs performed by both algorithms. Red nodes: CITs performed only by PC-stable (skipped by DF-PC). Blue nodes: queries deduced by DF-PC without any CIT. Green nodes: intermediate queries resolved internally by DEDUCE-HELPER. Directed edges indicate logical entailment via Prop. 1 or Prop. 5. PC-stable performs 16 CITs in total, whereas DF-PC performs only 9.

on the motivating example shown in Fig. 1. As visually mapped by the diagram in Fig. 2, we compare the query behavior of both algorithms with a focus on the CITs explicitly performed or logically deduced at each level (i.e., the conditioning set size of CI queries). We assume that an oracle CI tester is given for both algorithms.

At Level 0 ($k = 0$), both algorithms begin by testing all pairs for unconditional independence. They execute queries such as $CI(X;Y \mid \emptyset)$, $CI(Y;Z' \mid \emptyset)$, and others. Given an oracle CI tester, both algorithms correctly identify the marginal independencies $Y \perp\!\!\!\perp Z'$ and $Y \perp\!\!\!\perp Z''$, removing the edges (Y, Z') and (Y, Z'') from the skeleton. All other pairs are found to be *dependent*.

At Level 1 ($k = 1$), PC-stable continues to test all remaining edges using the available neighbors. For the edge (X, Y), it explicitly performs the tests on $CI(X;Y \mid \{Z'\})$ and $CI(X;Y \mid \{Z''\})$. Both tests return *dependent*. DF-PC, however, applies deductive reasoning instead of performing statistical tests. For the query $CI(X;Y \mid \{Z'\})$, DEDUCE (Alg. 3) calls its recursive function DEDUCE-HELPER (Alg. 4) to deduce its outcome. The recursive function then considers the triplet of queries involving the variables X, Y, Z' with conditioning set $\emptyset$: $CI(X;Y \mid \emptyset)$, $CI(X;Z' \mid \emptyset)$, and $CI(Y;Z' \mid \emptyset)$. It checks the previous results: $X \not\perp\!\!\!\perp Y$, $X \not\perp\!\!\!\perp Z'$, and $Y \perp\!\!\!\perp Z'$. Since the condition for deducing *dependence* is met, the recursive function and its wrapper DEDUCE return *dep*. In this manner, DF-PC skips the explicit CITs for $CI(X;Y \mid \{Z'\})$ and $CI(X;Y \mid \{Z''\})$.

At Level 2 ($k = 2$), PC-stable proceeds to test queries with size-2 subsets, performing the test $CI(X;Y \mid \{Z', Z''\})$.

This test requires conditioning on two variables and returns *dependent*. DF-PC avoids this test by recursively applying deduction rules. To determine the dependency of the query $CI(X;Y \mid \{Z', Z''\})$, the recursive function of DEDUCE recursively checks subsets by removing one variable from the conditioning set. It selects Z'' to remove, leaving $\{Z'\}$ as the base set. It then evaluates the triplet of queries conditioned on $\{Z'\}$: $CI(X;Y \mid \{Z'\})$, $CI(Y;Z'' \mid \{Z'\})$, and $CI(X;Z'' \mid \{Z'\})$.

First, the outcomes of queries $CI(X;Y \mid \{Z'\})$ and $CI(X;Z'' \mid \{Z'\})$ are already given as *dependent* (Level 1 results). Second, for $CI(Y;Z'' \mid \{Z'\})$, neither algorithm performed testing or deduction. Therefore, the recursive function of DEDUCE tries to recursively apply deductive rules on this new query. It then checks the triplet of queries involving Y, Z'', Z' with conditioning set $\emptyset$. Since $Y \perp\!\!\!\perp Z''$, $Y \perp\!\!\!\perp Z'$ and $Z' \not\perp\!\!\!\perp Z''$ (Level 0 results), the condition for deducing *independence* from Prop. 5 is satisfied. Thus, $CI(Y;Z'' \mid \{Z'\})$ is deduced as *independent*. Finally, combining these results—$CI(X;Y \mid \{Z'\})$ and $CI(X;Z'' \mid \{Z'\})$ are *dependent* and $CI(Y;Z'' \mid \{Z'\})$ is *independent*—the algorithm triggers rules for deducing dependence for the original query. Taking the outcome of the query from its recursive function, DEDUCE thus returns *dependent* on $CI(X;Y \mid \{Z', Z''\})$ without any additional tests. Therefore, DF-PC skips the explicit CIT for $CI(X;Y \mid \{Z', Z''\})$.

4.4 THEORETICAL GUARANTEES

The DF-PC algorithm preserves all soundness guarantees of the standard PC-stable algorithm but may require fewer CITs. In the following, we provide soundness results for DF-PC (Thm. 1, Thm. 2), along with best- and worst-case bounds on the number of CITs required by DF-PC (Thm. 3).

Theorem 1 (Soundness of DEDUCE)**.** *Let (G, P) be a faithful Bayesian network and let* CIT *be an oracle CI tester. Assume that the cache at entry to* DEDUCE *is sound: every cached* IND *or* DEP *label is correct in P, while* UNKNOWN *labels impose no condition. For all distinct variables X, Y and all conditioning sets $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$,*

$$\text{DEDUCE}(X, Y, \mathbf{Z}) = \text{IND} \implies X \perp\!\!\!\perp Y \mid \mathbf{Z},$$
$$\text{DEDUCE}(X, Y, \mathbf{Z}) = \text{DEP} \implies X \not\perp\!\!\!\perp Y \mid \mathbf{Z}.$$

Theorem 2 (Soundness and completeness of DF-PC)**.** *Let (G, P) be a faithful Bayesian network and let* CIT *be an oracle CI tester. Then, under the oracle* CIT*, the DF-PC algorithm is sound and complete for recovering a pattern that represents G.*

Theorem 3 (Bounds on the number of CITs required by DF-PC)**.** *Let (G, P) be a faithful Bayesian network and let* CIT *be an oracle CI tester. Let m denote the number of variables and Δ the maximum degree of the underlying graph. The total number of CITs performed by DF-PC is bounded as follows:*

1. ***Worst-case Upper Bound:*** $O(m^{\Delta+2})$,
2. ***Best-case Lower Bound:*** $\Omega(m^2)$.

Furthermore, the best-case lower bound is tight: there exists a faithful DAG for which DF-PC achieves exact structure recovery using only the initial $\binom{m}{2}$ marginal CITs.

Remark 4. When the true graph has no edges, skeleton discovery reduces to a single round of marginal independence tests, which is the minimum possible for any algorithm. Interestingly, DF-PC can also attain this same minimum on certain non-trivial topologies where $|\mathbf{V}| - 1$ edges are present. Consider, for example, a collider hub: a star-like DAG where $X_i \to Y \leftarrow X_j$ for all $i \neq j$. In such a graph, variables X_i and X_j are marginally independent ($X_i \perp\!\!\!\perp X_j$), which is discovered at the marginal level and leads to the immediate deletion of $X_i - X_j$ edges. Crucially, using our deductive rules, DF-PC logically infers that the remaining adjacencies $X_i - Y$ must exist due to the marginal independencies among the parents X_i, thereby preserving all true edges and finalizing the skeleton without performing any CIT beyond the marginal level. Such query reduction is unattainable by standard approaches like PC-stable. A formal construction is provided in the proof of Theorem 3; for empirical evidence, see Appendix C.

The power of the "subtractive" mechanism in DF-PC fundamentally stems from how the algorithm utilizes its query history, specifically by extracting and maximizing the value of independence findings. In stark contrast to traditional PC-like algorithms, where a discovered independence is used merely to delete an edge and passively reduce the search space for future queries, DF-PC maximizes the utility of this information. Rather than simply discarding an edge and moving on, the algorithm leverages these independence findings as critical logical constraints. Through the deduction process, DF-PC actively preempts redundant tests and carefully mitigates the possibility of false negatives (i.e., instances where true edges are erroneously deleted).

In essence, this theoretical bound is achieved because DF-PC effectively exploits independence findings during skeleton learning. While the algorithm inherently tracks the entire query history, the queries evaluated as independent are its true driving force. Without these specific findings acting as structural anchors, the deductive framework of DF-PC would offer no advantage over standard approaches.

5 EMPIRICAL EVALUATIONS

In this section, we empirically evaluate the proposed DF-PC algorithm against the PC-stable baseline. Our objective is to

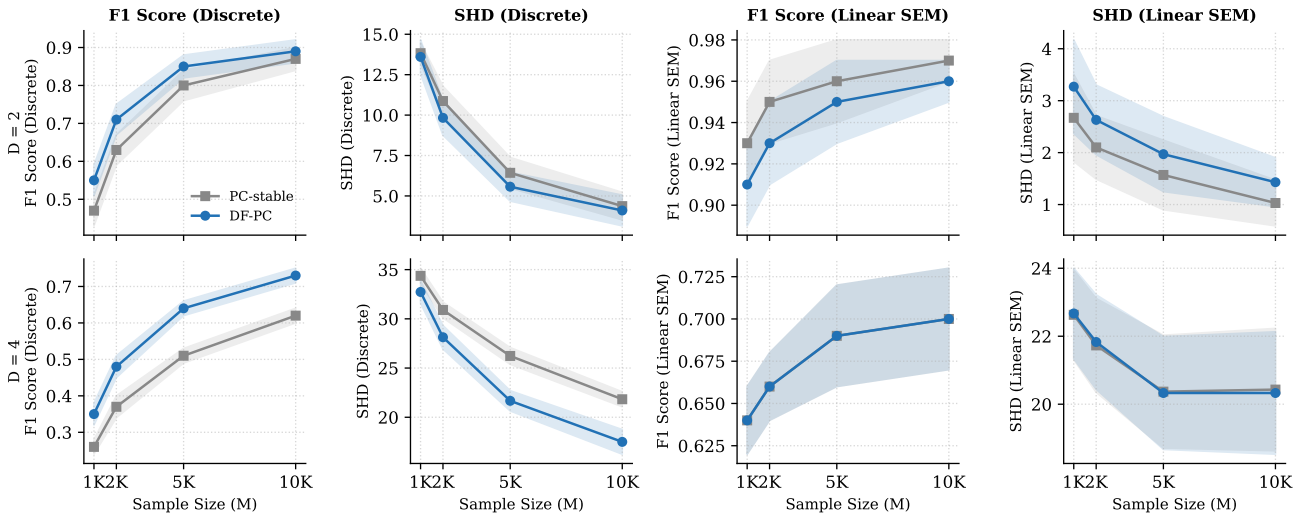

Figure 3: **Learning Performance comparison** (F1 Score and SHD vs. Sample Size M) on ER graphs ($|V| = 20$, $D \in \{2, 4\}$) with 95% confidence intervals across Discrete and Linear SEM datasets. The full numerical results (including Precision and Recall) are provided in Table 3 in the Appendix.

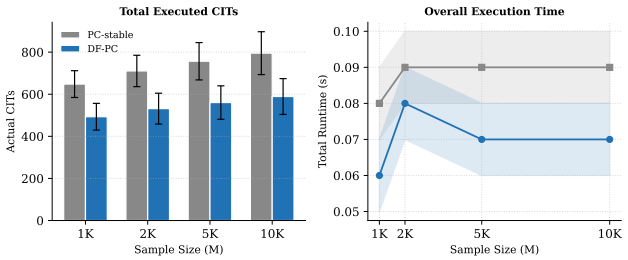

Figure 4: **Computational Efficiency on Linear SEM** (ER graphs, $|\mathbf{V}| = 20$, $D = 2$) with 95% confidence intervals. Left panel shows the actual CITs performed, and right panel shows the total runtime. The full numerical results are provided in Table 4 in the Appendix.

investigate how integrating logical deduction into constraint-based causal discovery affects both computational efficiency and structural learning accuracy in finite-sample regimes.

5.1 EXPERIMENTAL SETUP

Graph Generation and Topology We conduct experiments using synthetic datasets to strictly control the experimental conditions. We generated Directed Acyclic Graphs (DAGs) using two distinct topological models: Erdős-Rényi (ER) random graphs [Erdős and Rényi, 1960] and Scale-Free (SF) networks [Barabási and Albert, 1999]. To evaluate scalability, we varied the number of variables $N \in \{10, 20, 30\}$ and the density of the graphs by setting the expected average degree $D \in \{2, 4\}$. To ensure computational tractability when generating discrete data, the maximum in-degree of any node was strictly capped at 10.

Data Generation and CITs For each generated DAG, we sampled finite datasets of varying sizes $M \in \{1000, 2000, 5000, 10000\}$ under two distinct generating mechanisms, allowing us to evaluate the algorithms using different CITs: (1) **Discrete Data:** We modeled the variables using categorical distributions with a cardinality of $c = 3$. conditional probability tables were uniformly sampled, with a floor $p = 0.2$ added to ensure strictly positive probabilities and avoid deterministic relationships. Independence was assessed using the standard χ^2 (Chi-square) test. (2) **Linear SEM Data:** We modeled continuous variables using a linear Structural Equation Model (SEM) with additive Gaussian noise. To ensure sufficiently strong causal signals, edge weights were sampled uniformly from the intervals $[-2.0, -0.5] \cup [0.5, 2.0]$. Independence was evaluated using the Partial Correlation (Fisher-Z) test.

Evaluation Metrics We evaluate the algorithms along two dimensions: accuracy and efficiency. (1) **Accuracy:** From each algorithm's output, we construct the undirected skeleton and compare it with the ground-truth skeleton. In our main text figures, we focus on the adjacency *F1-score* and *Structural Hamming Distance (SHD)* [Tsamardinos et al., 2006] to visually demonstrate structural accuracy, with the complete set of accuracy metrics (including *Precision* and *Recall*) provided in the Appendix. (2) **Efficiency:** In the main text, we evaluate computational efficiency via the total number of executed CITs (*Actual CITs*) and the overall *Execution Time* (seconds), with the full breakdown of efficiency metrics (e.g., *CIT Time*) provided in the Appendix.

Implementation Details Our implementation and experiments focus on skeleton discovery; CPDAG orientation is not included in the released implementation. For CIT, both

algorithms utilized a fixed significance level of $\alpha = 0.01$. To ensure statistical significance, every experimental configuration was repeated across 30 independent trials, and we report the mean performance along with 95% confidence intervals.

Extended Analyses In addition to the synthetic benchmarks presented, we conducted supplemental experiments to investigate DF-PC. We evaluate (1) performance in nonlinear environments; (2) performance limits under best- and worst-case structures; (3) sensitivity to the significance threshold α; (4) performance on a large graph; and (5) performance on a semi-synthetic dataset for practical relevance. Detailed settings and full results for these analyses are provided in Appendix C.

5.2 RESULTS

Learning Accuracy: Figure 3 visualizes the structural accuracy of the recovered skeletons. DF-PC shows comparable or superior performance to PC-stable across most configurations in terms of adjacency F1 and skeleton SHD. The performance gain is prominent in discrete settings, whereas both methods show comparable accuracy in Linear SEM. This structural advantage might stem from executing fewer statistical tests, which can reduce the accumulation of test errors. However, we note that it can also propagate errors in the CI results on which it relies. As the sample size (M) grows, the reliability of individual CITs improves, naturally narrowing the performance gap between the two algorithms. Furthermore, DF-PC sustains this comparative advantage or equivalence even as graph density (D) increases. The full numerical results (including Precision and Recall) are provided in Table 3 in the Appendix.

Computational Efficiency: Figure 4 demonstrates the efficiency gains of our approach. DF-PC significantly reduces the total number of executed CITs, primarily by bypassing CITs ($|\mathbf{Z}| > 0$) via logical deduction. Runtime savings depend on the cost of CI testing relative to deduction and cache processing. With inexpensive CI tests, this overhead can offset the savings. As demonstrated in our nonlinear dataset experiments (Appendix C), minimizing the number of CITs often leads to a dramatic decrease in execution time. These results demonstrate substantial CIT savings, while runtime and accuracy benefits vary across settings. The full numerical results are provided in Table 4 in the Appendix.

6 CONCLUSION

We presented DF-PC, a variant of PC that not only uses independence findings as edge-deletion signals but also as ingredients of logical deduction. In our experiments, DF-PC reduced higher-order CITs, with runtime and accuracy benefits varying across settings. Future work will include improving robustness to false positives in the deduction process.

Acknowledgements

We thank anonymous reviewers for constructive comments to improve the manuscript. This work was supported by the NRF (RS-2023-00211904) grant funded by the Korean government.

References

Ankur Ankan and Johannes Textor. pgmpy: A python toolkit for bayesian networks. *Journal of Machine Learning Research*, 25(265):1–8, 2024. URL http://jmlr.org/papers/v25/23-0487.html.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

Facundo Bromberg and Dimitris Margaritis. Improving the reliability of causal discovery from small data sets using argumentation. *Journal of Machine Learning Research*, 10(12):301–340, 2009.

Carlos Cinelli, Avi Feller, Guido Imbens, Edward Kennedy, Sara Magliacane, and Jose Zubizarreta. Challenges in statistics: A dozen challenges in causality and causal inference. *arXiv preprint arXiv:2508.17099*, 2025.

Diego Colombo and Marloes H. Maathuis. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782, 2014.

Nicholas Cornia and Joris M. Mooij. Type-ii errors of independence tests can lead to arbitrarily large errors in estimated causal effects: An illustrative example. In *CI@ UAI*, pages 35–42, 2014.

Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

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

Marc Franquesa Monés. Bounding the number of conditional independence tests in constraint-based causal discovery. Master's thesis, Universitat Politècnica de Catalunya, 2025.

Dan Geiger. *Graphoids: A qualitative framework for probabilistic inference*. PhD thesis, University of California, Los Angeles, Department of Computer Science, 1990.

Sujai Hiremath, Dominik Janzing, Philipp Faller, Patrick Blöbaum, Elke Kirschbaum, Shiva Prasad Kasiviswanathan, and Kyra Gan. From guess2graph: When and how can unreliable experts safely boost causal discovery in finite samples? *arXiv preprint arXiv:2510.14488*, 2025.

Markus Kalisch and Peter Bühlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(22): 613–636, 2007. URL http://jmlr.org/papers/v8/kalisch07a.html.

Jonghwan Kim, Inwoo Hwang, and Sanghack Lee. Causal discovery with deductive reasoning: one less problem. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

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

Steffen L Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

Junning Li and Z Jane Wang. Controlling the false discovery rate of the association/causality structure learned with the pc algorithm. *Journal of Machine Learning Research*, 10 (2), 2009.

Daniel Malinsky. A cautious approach to constraint-based causal model selection. *arXiv preprint arXiv:2404.18232*, 2024.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Judea Pearl and Azaria Paz. Graphoids: Graph-based Logic for Reasoning about Relevance Relations. In B. du Boulay, D. Hogg, and L. Steels, editors, *Advances in Artificial Intelligence II*, pages 357–363. North-Holland Publishing Co., 1987.

Joseph Ramsey, Jiji Zhang, and Peter L. Spirtes. Adjacency-faithfulness and conservative causal inference. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 401–408, 2006.

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

Kirankumar Shiragur, Jiaqi Zhang, and Caroline Uhler. Causal discovery with fewer conditional independence tests. *arXiv preprint arXiv:2406.01823*, 2024.

Peter Spirtes. An anytime algorithm for causal inference. In *International Workshop on Artificial Intelligence and Statistics*, pages 278–285. PMLR, 2001.

Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

Eric V Strobl, Peter L Spirtes, and Shyam Visweswaran. Estimating and controlling the false discovery rate of the pc algorithm using edge-specific p-values. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10 (5):1–37, 2019.

Milan Studenỳ. Conditional independence relations have no finite complete characterization. In *Information Theory, Statistical Decision Functions and Random Processes. Transactions of the 11th Prague Conference vol. B*, pages 377–396, 1992.

Johannes Textor, Alexander Idelberger, and Maciej Liśkiewicz. Learning from pairwise marginal independencies. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, 2015.

Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1): 31–78, 2006.

Marcel Wienöbst and Maciej Liskiewicz. Recovering causal structures from low-order conditional independencies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10302–10309, 2020.

Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 804–813, 2011.

Yujia Zheng, Biwei Huang, Wei Chen, Joseph Ramsey, Mingming Gong, Ruichu Cai, Shohei Shimizu, Peter Spirtes, and Kun Zhang. Causal-learn: Causal discovery in python. *Journal of Machine Learning Research*, 25(60):1–8, 2024.

Don't Test What You Can Deduce: Causal Discovery with Logical Inference (Supplementary Material)

Jonghwan Kim[1] **Sanghack Lee**[*1]

[1]Graduate School of Data Science, Seoul National University, South Korea
[*]Correspondence to: sanghack@snu.ac.kr

A RULES FROM GRAPHOID AXIOMS AND FAITHFULNESS

We present rules derived from graphoid axioms [Pearl and Paz, 1987, Geiger, 1990, Bromberg and Margaritis, 2009].

$$\begin{aligned}
\text{(Symmetry) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \iff (\mathbf{Y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{Z}) \\
\text{(Decomposition) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}) \\
\text{(Weak Union) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}, \mathbf{W}) \\
\text{(Contraction) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}, \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) \\
\text{(Intersection) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}, \mathbf{W}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}, \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z})
\end{aligned}$$

Under a (causal) faithfulness assumption, we have a more relaxed set of rules as follows.

$$\begin{aligned}
\text{(Symmetry) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \iff (\mathbf{Y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{Z}) \\
\text{(Composition) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) \\
\text{(Decomposition) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}) \\
\text{(Intersection) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}, \mathbf{W}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}, \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) \\
\text{(Weak Union) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}, \mathbf{W}) \\
\text{(Contraction) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}, \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) \\
\text{(Weak Transitivity) } & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}, W) \implies (\mathbf{X} \perp\!\!\!\perp W \mid \mathbf{Z}) \vee (W \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})
\end{aligned}$$

Each non-bold letter in weak transitivity is a variable (i.e., a singleton).

B OMITTED PROOFS

Proposition 2 (Query history of a PC-like algorithm)**.** *Let X and Y be distinct variables, and let $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$ be a nonempty set of variables. If a PC-like algorithm examines the CI query $CI(X; Y \mid \mathbf{Z})$ with $|\mathbf{Z}| \geq 1$, then at the moment of this evaluation, the algorithm's query history contains:*

1. *the results of all pairwise CI queries $CI(U; V \mid \emptyset)$ for all distinct variables U, V; and*
2. *the results of all CI queries $CI(X; T \mid \mathbf{S})$ yielding dependence $(X \not\perp\!\!\!\perp T \mid \mathbf{S})$, for every T and $\mathbf{S}$ satisfying $T \in \{Y\} \cup \mathbf{Z}$ and $\mathbf{S} \subseteq (\{Y\} \cup \mathbf{Z}) \setminus \{T\}$ s.t. $|\mathbf{S}| < |\mathbf{Z}|$.*

Proof. A PC-like algorithm starts from the complete undirected graph. By the *level-wise monotonicity* property, it processes primary CI queries in nondecreasing conditioning-set size $\ell = 0, 1, 2, \ldots$. For each edge that survives level ℓ, all eligible conditioning sets of size ℓ are examined. Once an edge is removed, no further tests involving that edge are performed. Suppose that at some moment the algorithm examines the CI query

$$CI(X; Y \mid \mathbf{Z}), \qquad |\mathbf{Z}| = \ell \geq 1.$$

(1) All unconditional pairwise CI queries correspond to the case $\ell = 0$. By *level-wise monotonicity*, since the algorithm examines $CI(X; Y \mid \mathbf{Z})$ at level $\ell \geq 1$, it has already exhaustively completed level $\ell = 0$. It therefore necessarily has the results of all pairwise queries (U, V) for all singleton variables U, V.

(2) By the *local adjacency constraint*, evaluating $CI(X; Y \mid \mathbf{Z})$ at level ℓ requires that $\mathbf{Z} \subseteq \mathrm{adj}(X)_{\mathcal{G}'} \setminus \{Y\}$. This implies that the edge (X, Y) and all edges (X, Z_i) for $Z_i \in \mathbf{Z}$ remain in the current skeleton at the moment of this query. Consequently, none of the edges (X, T) with

$$T \in \{Y\} \cup \mathbf{Z}$$

have been removed at any previous level. Consider any

$$T \in \{Y\} \cup \mathbf{Z} \quad \text{and} \quad \mathbf{S} \subseteq (\{Y\} \cup \mathbf{Z}) \setminus \{T\}$$

with $|\mathbf{S}| = \ell' < \ell$. By *level-wise monotonicity*, the query $CI(X; T \mid \mathbf{S})$ must have been evaluated at the earlier level ℓ'. Furthermore, since $\mathbf{S}$ consists entirely of nodes adjacent to X that survived up to level ℓ, it was inherently a valid conditioning set under the *local adjacency constraint* at level ℓ'. Since the edge (X, T) is still present at level ℓ, it could not have been removed at level ℓ' or any earlier level. Therefore, for every such $\mathbf{S}$, the algorithm must have already performed the corresponding test and obtained the dependence result

$$(X \not\perp\!\!\!\perp T \mid \mathbf{S}).$$

This concludes the proof. □

Proposition 3 (Worst-case upper bound on the total number of additional CITs required by DEDUCE-DEP)**.** *Suppose we run the PC algorithm and encounter a CI query $CI(X; Y \mid \mathbf{Z})$ with $|\mathbf{Z}| = k > 0$. Assume that we attempt to resolve this query using* DEDUCE-DEP*, with its reliability threshold set to zero. Then, in the worst case, the number of* new *CITs performed by* DEDUCE-DEP *is* $\frac{k}{8}\big(2^k(k+3) - 4(k+1)\big)$,

which is $O(k^2 2^k)$.

Proof. Consider a query of the form $(X; Y \mid \mathbf{Z})$ with $|\mathbf{Z}| = k > 0$. Suppose we set the reliability threshold of DEDUCE-DEP to be 0. The number of CITs required by DEDUCE-DEP can be computed level by level. At level $l < k$, DEDUCE-DEP considers all conditioning sets of size l constructed from the elements of $\mathbf{Z}$. The number of such conditioning sets is $\binom{k}{l}$. For each conditioning set, the algorithm examines conditional independence relations between all unordered pairs of variables selected from

$$\{X, Y\} \cup (\mathbf{Z} \setminus \mathbf{S}),$$

where $\mathbf{S}$ denotes the chosen conditioning set with $|\mathbf{S}| = l$. Since the size of this set is $k + 2 - l$, the number of variable pairs to be tested is $\binom{k+2-l}{2}$. Therefore, the number of CITs required at level l is given by

$$\binom{k}{l}\binom{k+2-l}{2}.$$

Summing over all levels $l = 0, \ldots, k-1$, the total number of CITs required by DEDUCE-DEP is

$$T(k) = \sum_{l=0}^{k-1} \binom{k}{l}\binom{k+2-l}{2} = 2^{k-3}(k^2 + 7k + 8) - 1.$$

By Prop. 2, some CIT outcomes are already available when examining $(X; Y \mid \mathbf{Z})$ and can be reused. At level $l = 0$, all unconditional (pairwise) CITs over $\{X, Y\} \cup \mathbf{Z}$ are assumed to be available, yielding

$$R_0(k) = \binom{k}{0}\binom{k+2}{2} = \binom{k+2}{2}.$$

For each level $1 \leq l < k$, there are $\binom{k}{l}$ choices of conditioning sets $\mathbf{S} \subseteq \mathbf{Z}$ with $|\mathbf{S}| = l$. Prop. 2 implies that reusable tests at this level are restricted to pairs that include X, i.e., (X, V) with $V \in \{Y\} \cup (\mathbf{Z} \setminus \mathbf{S})$, whose count is $|\{Y\} \cup (\mathbf{Z} \setminus \mathbf{S})| = k + 1 - l = \binom{k+1-l}{1}$. Hence the number of reusable CITs at level l is

$$\binom{k}{l}\binom{k+1-l}{1}.$$

Summing over all levels gives the total number of reusable CITs:

$$R(k) = \binom{k+2}{2} + \sum_{l=1}^{k-1} \binom{k}{l} \binom{k+1-l}{1}$$
$$= \frac{k+2}{2}\big(2^k + k - 1\big).$$

Therefore, the worst-case upper bound on the number of *new* CITs required by DEDUCE-DEP is

$$N(k) = T(k) - R(k)$$
$$= 2^{k-3}\big(k^2 + 7k + 8\big) - 1 - \frac{k+2}{2}\big(2^k + k - 1\big).$$

which simplifies to the closed form

$$= \frac{k}{8}\Big(2^k(k+3) - 4(k+1)\Big).$$

In particular, $N(k) = O(k^2 2^k)$ and for large k,

$$N(k) \approx \frac{k^2}{8}\, 2^k.$$

This concludes the proof. □

Proposition 4 (Worst-case upper bound on the total number of CITs required by PC with DEDUCE-DEP)**.** *Let m denote the number of variables and Δ the maximum degree of the underlying graph. Under the oracle CI tester, the total number of CITs performed by the PC algorithm with* DEDUCE-DEP *in the worst case, is bounded by $O\big(m^{\Delta+2} + m^2 \cdot \Delta^2 2^{\Delta}\big)$.*

Proof. Let m_{reach} be the maximum conditioning-set size evaluated by the PC algorithm under an oracle CI tester. The number of CI queries posed in the worst case of PC algorithm is bounded by $2\binom{m}{2}\sum_{l=0}^{m_{reach}}\binom{m-2}{l}$. By Proposition 1 of Kalisch and Bühlmann [2007], $m_{reach} \in \{\Delta - 1, \Delta\}$ holds. Hence, $m_{reach} \leq \Delta$. Using the maximum degree of the true graph, the number of CI queries can be upper bounded by $2\binom{m}{2}\sum_{l=0}^{\Delta}\binom{m-2}{l} = O(m^{\Delta+2})$. Since DEDUCE-DEP is invoked only when a primary query evaluates to an independence (ind) result from CI tester, we must bound the total number of such occurrences.

An ind outcome directly implies the removal of an edge from the skeleton graph. Because the algorithm initializes with a complete undirected graph containing $\binom{m}{2}$ edges, the ind result can occur at most $O(m^2)$ times globally.

For each of these $O(m^2)$ instances, the algorithm invokes DEDUCE-DEP, which explores a recursive tree up to depth $\ell \leq \Delta$. By Prop. 3, the maximum number of additional CITs invoked by DEDUCE-DEP for a single query at the maximum depth is bounded by $O(\Delta^2 2^{\Delta})$.

Therefore, the total number of CITs is the sum of the primary statistical tests performed by the PC-stable algorithm and the additional fallback CITs generated by DEDUCE-DEP:

$$O\big(m^{\Delta+2} + m^2 \cdot \Delta^2 2^{\Delta}\big)\,.$$

□

Proposition 5. *Under the faithful Bayesian network $(\mathcal{G}, P)$, let $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ be disjoint subsets of $\mathbf{V}$ where $\mathbf{Z}$ is partitioned into $\mathbf{Z}'$ and $\mathbf{Z}''$ such that $\mathbf{Z} = \mathbf{Z}' \sqcup \mathbf{Z}''$, $|\mathbf{Z}''| = 1$. Then, $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ if one of the following holds:*

1. $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}') \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Z}'' \mid \mathbf{Z}')$
2. $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}') \wedge (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}'' \mid \mathbf{Z}')$

Proof. We prove each item below.

(1) We begin with the definition of the composition rule:

$$(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}') \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Z}'' \mid \mathbf{Z}') \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y}, \mathbf{Z}'' \mid \mathbf{Z}').$$

By the weak union rule, we have:

$$\implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}', \mathbf{Z}'').$$

Therefore, if $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}')$ and $(\mathbf{X} \perp\!\!\!\perp \mathbf{Z}'' \mid \mathbf{Z}')$, then $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}' \cup \mathbf{Z}'')$.

(2) The proof is identical to that of item (1), except that we replace $(\mathbf{X} \perp\!\!\!\perp \mathbf{Z}'' \mid \mathbf{Z}')$ with $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}'' \mid \mathbf{Z}')$. This follows since conditional independence is symmetric in $\mathbf{X}$ and $\mathbf{Y}$. □

Lemma 1 (Soundness of DEDUCE-HELPER)**.** *Let (G, P) be a faithful Bayesian network. A cache is sound if every stored* IND *label denotes an independence in P and every stored* DEP *label denotes a dependence in P;* UNKNOWN *imposes no condition. For every sound initial cache and every valid query $CI(X; Y \mid \mathbf{Z})$,* DEDUCE-HELPER *terminates, preserves soundness of the cache, and satisfies*

$$\text{DEDUCE-HELPER}(X, Y, \mathbf{Z}) = \text{IND} \implies X \perp\!\!\!\perp Y \mid \mathbf{Z},$$
$$\text{DEDUCE-HELPER}(X, Y, \mathbf{Z}) = \text{DEP} \implies X \not\perp\!\!\!\perp Y \mid \mathbf{Z}.$$

Here a deductive rule is applied only when each of its required premises has the corresponding decisive label in the cache.

Proof. We use strong induction on $k = |\mathbf{Z}|$. For $k = 0$, the loop is empty, the procedure returns UNKNOWN, and the cache remains sound trivially.

Let $k \geq 1$ and suppose the assertion holds for all smaller conditioning sets. Start the current call DEDUCE-HELPER$(X, Y, \mathbf{Z})$ with any sound initial cache. In the loop, each child query has conditioning set $\mathbf{Z} \setminus \{Z\}$ of size $k - 1$. For an uncached query, the inductive hypothesis applies to the recursive call with the cache as it stands at that moment: the call terminates, preserves soundness of the cache, and returns either a correct decisive answer or UNKNOWN. Recording this answer therefore keeps the cache sound. This argument applies successively to every child call.

We now analyze the return branch taken by DEDUCE-HELPER. If the procedure returns DEP, the required premises of one of the rules in Proposition 1 are true, so its conclusion $X \not\perp\!\!\!\perp Y \mid \mathbf{Z}$ holds in P. If the procedure returns IND, at least one iteration established the premises of Proposition 5, while no iteration satisfied the dependence rule. In this case, $X \perp\!\!\!\perp Y \mid \mathbf{Z}$ holds in P. If the procedure returns UNKNOWN, neither a dependence nor an independence rule applies for any $Z \in \mathbf{Z}$ and the result itself makes no claim. All loops are finite, so the current call terminates. This proves the inductive assertion. □

Theorem 1 (Soundness of DEDUCE)**.** *Let (G, P) be a faithful Bayesian network and let* CIT *be an oracle CI tester. Assume that the cache at entry to* DEDUCE *is sound: every cached* IND *or* DEP *label is correct in P, while* UNKNOWN *labels impose no condition. For all distinct variables X, Y and all conditioning sets $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$,*

$$\text{DEDUCE}(X, Y, \mathbf{Z}) = \text{IND} \implies X \perp\!\!\!\perp Y \mid \mathbf{Z},$$
$$\text{DEDUCE}(X, Y, \mathbf{Z}) = \text{DEP} \implies X \not\perp\!\!\!\perp Y \mid \mathbf{Z}.$$

Proof. By Lemma 1, a decisive answer from DEDUCE-HELPER is correct. If DEDUCE-HELPER returns UNKNOWN, DEDUCE returns the correct answer by invoking a CIT. These cases exhaust the control flow of DEDUCE, so DEDUCE terminates and always returns the correct conditional (in)dependence statements. The helper preserves cache soundness by Lemma 1, and recording the correct answer returned by DEDUCE preserves it as well. Thus, starting from the empty cache, induction on the number of calls ensures that every call begins with a sound cache. □

Theorem 2 (Soundness and completeness of DF-PC)**.** *Let (G, P) be a faithful Bayesian network and let* CIT *be an oracle CI tester. Then, under the oracle* CIT*, the DF-PC algorithm is sound and complete for recovering a pattern that represents G.*

Proof. Algorithmically, DF-PC algorithm is identical to the order-independent PC-stable algorithm, with the sole distinction that the standard conditional independence oracle is replaced by the DEDUCE procedure. By Thm. 1, the DEDUCE procedure is sound: it returns *independence* if and only if $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ holds in the underlying faithful distribution P.

Consequently, for every query $CI(X; Y \mid \mathbf{Z})$ generated during the skeleton discovery phase, the decision to remove an edge in DF-PC is identical to that of the oracle PC-stable algorithm. Since the resulting skeleton and the recorded separating sets are identical, the subsequent orientation of v-structures and the application of Meek's rules yield the exact same CPDAG. Since the oracle PC-stable algorithm is sound and complete [Colombo and Maathuis, 2014], DF-PC outputs a pattern that correctly represents $\mathcal{G}$. □

Theorem 3 (Bounds on the number of CITs required by DF-PC)**.** *Let (G, P) be a faithful Bayesian network and let* CIT *be an oracle CI tester. Let m denote the number of variables and Δ the maximum degree of the underlying graph. The total number of CITs performed by DF-PC is bounded as follows:*

1. ***Worst-case Upper Bound:*** $O(m^{\Delta+2})$,
2. ***Best-case Lower Bound:*** $\Omega(m^2)$.

Furthermore, the best-case lower bound is tight: there exists a faithful DAG for which DF-PC achieves exact structure recovery using only the initial $\binom{m}{2}$ marginal CITs.

Proof. We prove the upper and lower bounds separately.

Worst-case Upper Bound: DF-PC algorithm shares the skeleton discovery framework of the PC-stable algorithm, generating at most the same number of CI queries, bounded by $O(m^{\Delta+2})$. For each query, the DEDUCE procedure (Alg. 3) triggers at most one statistical CIT, strictly as a fallback when the purely logical DEDUCE-HELPER (Alg. 4) returns unknown. Since DF-PC evaluates no more queries than PC-stable and performs at most one CIT per query, its total number of CITs in the worst case is strictly bounded by $O(m^{\Delta+2})$, matching the PC-stable algorithm.

Best-case Lower Bound: By definition of a PC-like algorithm, any PC-like algorithm must perform at least $\binom{m}{2}$ marginal independence tests at level $\ell = 0$. To prove this lower bound is tight for DF-PC, it suffices to construct a faithful DAG $\mathcal{G}$ of m nodes for which DF-PC requires zero CITs for all levels $\ell \geq 1$.

Consider a graph $\mathcal{G}$ defined as follows: let X be the target node, and let $\mathbf{M} = \mathbf{V} \setminus \{X\}$ be the set of remaining $m - 1$ nodes. The edges are defined such that every node in M points to X (i.e., $Y_i \to X$ for all $Y_i \in \mathbf{M}$), and there are no edges among the nodes in $\mathbf{M}$.

Level 0 (Marginal Tests): At level 0, DF-PC performs $\binom{m}{2}$ statistical CITs. The outcomes are recorded as follows:

1. $(Y_i \perp\!\!\!\perp Y_j \mid \emptyset)$ for all distinct $Y_i, Y_j \in \mathbf{M}$.
2. $(X \not\perp\!\!\!\perp Y_i \mid \emptyset)$ for all $Y_i \in \mathbf{M}$.

Following level 0, all edges between nodes in $\mathbf{M}$ are removed. The only remaining edges are $X - Y_i$. Therefore, for levels $\ell \geq 1$, the primary queries generated by the PC skeleton phase will only be of the form $CI(X; Y_i \mid \mathbf{Z})$ where $\mathbf{Z} \subseteq \mathbf{M} \setminus \{Y_i\}$. However, internally, the DEDUCE procedure may recursively query relations among nodes in $\mathbf{M}$. We prove by induction on $k = |\mathbf{Z}|$ that all subsequent queries are resolved purely by deduction.

Claim 1: Pure Deduction for Independence in $\mathbf{M}$. For any $Y_i, Y_j \in \mathbf{M}$ and $\mathbf{Z} \subseteq \mathbf{M} \setminus \{Y_i, Y_j\}$, DEDUCE$(Y_i, Y_j, \mathbf{Z})$ returns IND without invoking a CIT.
Proof by Induction: The base case $|\mathbf{Z}| = 0$ is given by Level 0 search. As the inductive hypothesis, assume the claim holds for any valid conditioning set of size $k - 1$. Let $Y_i, Y_j \in \mathbf{M}$ and $\mathbf{Z} = \mathbf{Z}' \cup \{Z\} \subseteq \mathbf{M} \setminus \{Y_i, Y_j\}$. To evaluate $CI(Y_i; Y_j \mid \mathbf{Z})$, the recursive function of DEDUCE checks $CI(Y_i; Y_j \mid \mathbf{Z}')$, $CI(Y_i; Z \mid \mathbf{Z}')$, and $CI(Y_j; Z \mid \mathbf{Z}')$. By the inductive hypothesis, all three are recursively resolved as IND without CIT. Since $(Y_i \perp\!\!\!\perp Y_j \mid \mathbf{Z}')$ and $(Y_i \perp\!\!\!\perp Z \mid \mathbf{Z}')$ hold, the predicate of Prop. 5 is satisfied, hence the recursive function and its wrapper DEDUCE returns IND.

Claim 2: Pure Deduction for Dependence between X and $\mathbf{M}$. For any $Y_i \in \mathbf{M}$ and $\mathbf{Z} \subseteq \mathbf{M} \setminus \{Y_i\}$, DEDUCE$(X, Y_i, \mathbf{Z})$ returns DEP without invoking a CIT.
Proof by Induction: The base case $|\mathbf{Z}| = 0$ is given by Level 0 search. As the inductive hypothesis, assume the claim holds for any valid conditioning set of size $k - 1$. Let $Y_i \in \mathbf{M}$ and $\mathbf{Z} = \mathbf{Z}' \cup \{Z\} \subseteq \mathbf{M} \setminus \{Y_i\}$. Then, the recursive function of DEDUCE checks $CI(X; Y_i \mid \mathbf{Z}')$, $CI(X; Z \mid \mathbf{Z}')$, and $CI(Y_i; Z \mid \mathbf{Z}')$ to resolve input query $CI(X; Y_i \mid \mathbf{Z})$. By the inductive hypothesis, the first query is resolved as DEP without CIT. By Claim 1, the third query is resolved as IND without CIT. Since $(X \not\perp\!\!\!\perp Y_i \mid \mathbf{Z}')$ and $(Y_i \perp\!\!\!\perp Z \mid \mathbf{Z}')$ hold, the predicate of Prop. 1 is satisfied. Therefore, the recursive function and its wrapper DEDUCE returns DEP.

Conclusion: For the graph $\mathcal{G}$, after the initial $\binom{m}{2}$ marginal tests, all primary queries (Claim 2) and secondary recursive queries (Claim 1) are logically resolved by the DEDUCE procedure. Thus, zero additional CITs are performed. Since no algorithm can bypass the $\binom{m}{2}$ marginal tests without prior knowledge, this establishes the tight best-case lower bound. □

C APPENDIX FOR EMPIRICAL EVALUATIONS

C.1 EXPERIMENTAL SETUP

To rigorously evaluate the performance and efficiency of the proposed DF-PC algorithm, we conducted comprehensive experiments using synthetic datasets. We compare our *DF-PC* variant against the PC-stable algorithm. All experiments were executed using 48 cores on an Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz to fully leverage the parallelizable nature of the experiments. Plus, all experiments were integrated with the `causal-learn` [Zheng et al., 2024] library for CIT implementations. To avoid intractable memory complexity in discrete Bayesian networks caused by exponentially large conditional probability tables, we restrict the maximum in-degree of any generated graph to 10. The source code for our implementation is available at `https://github.com/snu-causality-lab/DF-PC`.

We structure our evaluation into distinct experimental settings as follows:

Experiment 1: Synthetic Benchmarks (Scalability and Density) To assess the foundational scalability and robustness of DF-PC across various dimensions, we varied the number of variables, sample sizes, graph topologies, and data types.

- **Graph Topology:** We generated random DAGs with $N \in \{10, 20, 30\}$ nodes and average degrees $D \in \{2, 4\}$. We utilized both Erdős-Rényi (ER) and Scale-Free (SF) models (via Barabási-Albert).
- **Data Generation and CITs:**
  - *Discrete Data:* Sampled from discrete Bayesian networks with cardinality $c = 3$ and a floor $p = 0.2$. Independence was tested using the standard χ^2 test.
  - *Linear SEM Data:* Sampled from linear Structural Equation Models (SEMs) with additive Gaussian noise. Edge weights were sampled uniformly from $[-2.0, -0.5] \cup [0.5, 2.0]$. Independence was assessed via Partial Correlation (Fisher-Z test).
- **Sample Sizes (M):** $M \in \{1000, 2000, 5000, 10000\}$.
- **Hyperparameters:** Significance level $\alpha = 0.01$. Each configuration was repeated for 30 independent trials.

Experiment 2: Nonlinear Performance Logical deduction is particularly beneficial when statistical tests are computationally expensive. We verified this by applying DF-PC to nonlinear environments where kernel-based tests are required.

- **Graph Topology:** $N = 10$, average degree $D = 2$ (ER and SF graphs).
- **Data Generation and CITs:** We generated data using nonlinear additive noise models. The nonlinear functional relationships were modeled using 1-hidden-layer Multi-Layer Perceptrons (MLPs) with 100 hidden units, sigmoid activation functions, and Gaussian noise. Independence was evaluated using the Kernel-based Conditional Independence (KCI) test [Zhang et al., 2011].
- **Sample Size (M):** $M = 1000$.
- **Hyperparameters:** $\alpha = 0.01$, repeated for 30 independent trials.

Experiment 3: Extreme Structural Cases (Best vs. Failure Points) DF-PC's efficiency partially relies on local structural motifs (e.g., v-structures). To identify its theoretical boundary conditions on finite datasets, we designed specific extreme structures.

- **Graph Topology:** $N = 11$.
  - *Collider Hub (Best Case):* A star graph where $N - 1$ periphery nodes have directed edges pointing into a single central target node.
  - *Source Hub (Failure Case):* A star graph where a single central source node has directed edges pointing out to $N - 1$ periphery nodes.
- **Data Generation and CITs:** Linear SEM with Gaussian noise, assessed via Partial Correlation.

- **Sample Size (M):** $M = 2000$.
- **Hyperparameters:** $\alpha = 0.01$, repeated for 10 independent trials.

Experiment 4: Sensitivity to Significance Level (α) Because deductive logic assumes the absolute correctness of prior statistical tests, we investigated how varying strictness translates to overall logical stability and accuracy.

- **Graph Topology:** $N = 40$, average degree $D = 2$ (ER graphs).
- **Data Generation and CITs:** Linear SEM, assessed via Partial Correlation.
- **Sample Size (M):** $M = 5000$.
- **Hyperparameters:** We varied the significance level $\alpha \in \{0.05, 0.01, 0.001, 0.0001\}$. Each configuration was repeated for 10 independent trials.

Experiment 5: Large-Scale Networks ($N = 100$) To assess the stability and efficiency of DF-PC in high-dimensional spaces, we scaled the number of variables to $N = 100$, where the complexity of the conditioning set search space increases significantly.

- **Graph Topology:** $N = 100$, with both Erdős-Rényi (ER) and Scale-Free (SF) structures at densities $D \in \{2, 4\}$.
- **Data Generation and CITs:** Linear SEM, assessed via Partial Correlation.
- **Sample Size (M):** $M = 1000$.
- **Hyperparameters:** We applied a conservative significance level $\alpha = 0.001$, considering massive amounts of tests required in high-dimensional causal discovery. Each configuration was repeated for 30 independent trials to ensure statistical significance.

Experiment 6: Stability on Semi-Synthetic Benchmarks To further validate the practical utility of DF-PC across diverse structural patterns, we evaluated the algorithm on two well-established Bayesian network benchmarks from the agricultural and biological domains.

- **Graph Topology:** We utilized the *Mildew* (35 nodes, 46 edges) and *Barley* (48 nodes, 84 edges) networks, which exhibit a higher degree of structural complexity and variable cardinality.
- **Data Generation and CITs:** Categorical data were simulated from the respective `pgmpy` [Ankan and Textor, 2024] example models. Conditional independence was assessed via the χ^2 test.
- **Sample Size (M):** We tested two distinct regimes: $M \in \{2000, 5000\}$.
- **Hyperparameters:** $\alpha = 0.01$, repeated for 30 independent trials.

Experiment 7: Baseline Comparison with PC with DEDUCE-DEP To provide a comprehensive comparison across the full spectrum of deduction strategies, we evaluated three algorithms: PC-stable, PC with DEDUCE-DEP [Kim et al., 2024], and DF-PC.

- **Graph Topology:** $N = 20$ nodes, average degree $D = 4$, with both Erdős-Rényi (ER) and Scale-Free (SF) graph models.
- **Data Generation and CITs:**
  - *Discrete Data:* Bayesian networks with cardinality $c = 3$ and floor $p = 0.2$, tested via the χ^2 test.
  - *Linear SEM Data:* Linear SEMs with Gaussian noise and edge weights from $[-2.0, -0.5] \cup [0.5, 2.0]$, tested via Partial Correlation (Fisher-Z test).
- **Sample Size (M):** $M = 5000$.
- **Hyperparameters:** $\alpha = 0.01$. Each configuration was repeated for 30 independent trials.

Experiment 8: High-Density Stress Test To evaluate DF-PC's robustness under extreme graph density conditions where the conditioning set search space grows significantly, we tested on highly connected graphs.

- **Graph Topology:** $N = 30$ nodes, density parameter $D = 6$, with both ER and SF graph models. For ER, D specifies the target average degree; for SF, the attachment parameter is $D/2$.

- **Data Generation and CITs:**
  - *Discrete Data:* Bayesian networks tested via the χ^2 test.
  - *Linear SEM Data:* Linear SEMs tested via Partial Correlation.
- **Sample Size (M):** $M = 5000$.
- **Hyperparameters:** $\alpha = 0.01$, repeated for 30 independent trials.

Experiment 9: Runtime Scaling To characterize the computational scaling behavior of DF-PC, we measured execution time under two complementary scenarios: increasing the number of nodes while fixing density, and increasing density while fixing the number of nodes. For ER graphs, D specifies the target average degree. For SF graphs, the attachment parameter is D, rather than $D/2$ as in Experiment 8; equal values of D therefore do not represent matched average degrees across these experiments.

- **Setup A (Node Scaling):** $N \in \{10, 20, 30, 40, 50\}$ nodes with a fixed density parameter $D = 3$, using both ER and SF graph models.
- **Setup B (Density Scaling):** $N = 30$ nodes with density parameter $D \in \{2, 3, 4, 5, 6\}$, using both ER and SF graph models.
- **Data Generation and CITs:** Linear SEM data, tested via Partial Correlation.
- **Sample Size (M):** $M = 5000$.
- **Hyperparameters:** $\alpha = 0.01$, repeated for 30 independent trials.

Experiment 10: Oracle CIT Skip Analysis by $|\mathbf{Z}|$ To precisely quantify how many CITs DF-PC saves at each conditioning set size without confounding from statistical test errors, we employed an oracle conditional independence tester based on d-separation.

- **Setup A (Node Scaling):** $N \in \{10, 20, 30, 40\}$ nodes with average degree $D = 3$, using ER graphs.
- **Setup B (Density Scaling):** $N = 20$ nodes with average degree $D \in \{2, 3, 4, 5\}$, using ER graphs.
- **CIT Oracle:** All CI queries are resolved exactly via d-separation on the true underlying DAG (no statistical error).
- **Hyperparameters:** Repeated for 30 independent trials.

Experiment 11: Robustness against Low-Order CIT Errors Since DF-PC's deduction engine propagates outcomes of lower-order CITs to infer higher-order queries, errors in early tests could cascade through the deduction process. This experiment assesses the sensitivity of DF-PC to such errors.

- **Graph Topology:** $N = 20$ nodes, average degree $D = 3$, using ER graphs.
- **CIT Oracle with Injected Errors:** An oracle CIT based on d-separation, with random bit-flip errors injected into low-order CIT results ($|\mathbf{Z}| \leq 1$) at rates $p \in \{0.00, 0.01, 0.05, 0.10, 0.20\}$. Error injection is deterministic per query key (using a hash-based seed) to ensure reproducibility.
- **Compared Algorithms:** PC-stable and DF-PC.
- **Hyperparameters:** Repeated for 30 independent trials.

C.2 EXPERIMENTAL RESULTS

C.2.1 Synthetic Benchmark Results

We provide all experimental results with varying numbers of nodes ($|\mathbf{V}| = 10, 20, 30$) in Tables 1 to 6.

Table 1: Learning Performance for $|\mathbf{V}| = 10$ with 95% confidence interval. M corresponds to sample size, and D corresponds to average degree.

Graph	D	M	Method	Discrete				Linear SEM			
				F1	Precision	Recall	SHD	F1	Precision	Recall	SHD
ER	2	1000	PC-stable	0.47±0.06	**0.93**±0.07	0.33±0.05	6.87±0.58	**0.92**±0.02	**0.98**±0.02	0.88±0.03	**1.43**±0.38
			DF-PC	**0.58**±0.06	0.91±0.05	**0.44**±0.06	**6.03**±0.72	0.92±0.02	0.96±0.02	**0.88**±0.03	1.53±0.38
		2000	PC-stable	0.64±0.05	**0.99**±0.02	0.49±0.05	5.13±0.54	**0.94**±0.02	**0.98**±0.01	0.90±0.03	**1.20**±0.32
			DF-PC	**0.73**±0.05	0.94±0.03	**0.61**±0.06	**4.30**±0.65	0.93±0.02	0.96±0.02	**0.90**±0.03	1.33±0.34
		5000	PC-stable	0.81±0.03	**1.00**±0.00	0.69±0.05	3.07±0.46	**0.96**±0.02	**0.99**±0.01	**0.93**±0.03	**0.80**±0.34
			DF-PC	**0.87**±0.04	0.94±0.03	**0.82**±0.05	**2.30**±0.61	0.95±0.02	0.98±0.02	**0.93**±0.03	0.93±0.35
		10000	PC-stable	0.90±0.03	**1.00**±0.00	0.83±0.04	1.67±0.44	**0.97**±0.02	**0.99**±0.01	**0.95**±0.03	**0.63**±0.32
			DF-PC	**0.93**±0.02	0.96±0.02	**0.91**±0.03	**1.27**±0.36	0.96±0.02	0.98±0.02	**0.95**±0.03	0.77±0.33
	4	1000	PC-stable	0.27±0.04	**0.97**±0.03	0.16±0.03	16.93±0.55	0.68±0.03	**0.92**±0.03	0.54±0.03	10.13±0.83
			DF-PC	**0.35**±0.04	0.94±0.04	**0.22**±0.03	**15.90**±0.67	**0.68**±0.03	0.91±0.03	**0.55**±0.03	**10.10**±0.83
		2000	PC-stable	0.37±0.03	**0.99**±0.01	0.23±0.02	15.37±0.40	0.70±0.03	0.90±0.03	0.57±0.03	9.87±0.77
			DF-PC	**0.47**±0.04	0.95±0.02	**0.32**±0.04	**14.03**±0.70	**0.70**±0.02	**0.90**±0.03	**0.57**±0.02	**9.80**±0.77
		5000	PC-stable	0.55±0.03	**1.00**±0.00	0.38±0.03	12.33±0.55	**0.75**±0.02	**0.89**±0.02	**0.64**±0.03	**8.67**±0.69
			DF-PC	**0.64**±0.04	0.98±0.01	**0.48**±0.04	**10.57**±0.80	**0.75**±0.02	**0.89**±0.02	**0.64**±0.03	**8.67**±0.69
		10000	PC-stable	0.65±0.04	**1.00**±0.00	0.49±0.04	10.17±0.80	0.78±0.02	0.90±0.03	0.69±0.03	7.80±0.82
			DF-PC	**0.75**±0.03	0.99±0.01	**0.61**±0.05	**7.93**±0.89	**0.78**±0.02	**0.90**±0.03	**0.69**±0.03	**7.77**±0.82
SF	2	1000	PC-stable	0.40±0.06	**0.91**±0.08	0.27±0.04	6.77±0.43	**0.97**±0.02	**0.99**±0.01	0.95±0.03	**0.53**±0.32
			DF-PC	**0.44**±0.06	0.88±0.08	**0.31**±0.05	**6.57**±0.48	0.96±0.02	0.98±0.01	**0.95**±0.03	0.60±0.29
		2000	PC-stable	0.52±0.06	**0.92**±0.08	0.38±0.06	5.77±0.59	**0.99**±0.01	**0.99**±0.01	**0.98**±0.02	**0.23**±0.20
			DF-PC	**0.57**±0.06	0.90±0.05	**0.44**±0.06	**5.53**±0.61	0.98±0.01	0.97±0.02	**0.98**±0.02	0.43±0.26
		5000	PC-stable	0.64±0.06	**0.96**±0.03	0.50±0.07	4.67±0.65	**1.00**±0.00	**1.00**±0.00	**1.00**±0.01	**0.03**±0.07
			DF-PC	**0.71**±0.06	0.93±0.03	**0.60**±0.07	**4.03**±0.64	0.99±0.01	0.99±0.01	**1.00**±0.01	0.13±0.12
		10000	PC-stable	0.75±0.05	**0.98**±0.02	0.63±0.07	3.43±0.63	**1.00**±0.01	**1.00**±0.01	**1.00**±0.01	**0.07**±0.09
			DF-PC	**0.80**±0.05	0.95±0.03	**0.73**±0.07	**2.90**±0.65	0.99±0.01	0.99±0.01	**1.00**±0.01	0.17±0.17
	4	1000	PC-stable	0.25±0.04	**0.96**±0.05	0.15±0.03	13.70±0.49	0.77±0.03	**0.92**±0.03	0.67±0.03	**6.30**±0.85
			DF-PC	**0.31**±0.05	0.95±0.05	**0.19**±0.04	**13.03**±0.68	**0.77**±0.03	0.91±0.04	**0.68**±0.04	6.33±0.88
		2000	PC-stable	0.36±0.04	**0.97**±0.04	0.22±0.03	12.50±0.55	0.79±0.03	**0.90**±0.04	0.70±0.03	**6.10**±0.75
			DF-PC	**0.40**±0.05	0.96±0.04	**0.26**±0.04	**12.00**±0.71	**0.79**±0.03	0.89±0.03	**0.71**±0.03	**6.10**±0.77
		5000	PC-stable	0.44±0.05	**0.98**±0.03	0.29±0.04	11.40±0.69	0.83±0.03	0.91±0.03	0.76±0.03	5.10±0.75
			DF-PC	**0.48**±0.06	0.94±0.04	**0.34**±0.06	**10.80**±1.02	**0.83**±0.03	**0.91**±0.03	**0.76**±0.03	**5.07**±0.75
		10000	PC-stable	0.54±0.06	**1.00**±0.00	0.39±0.06	9.83±0.89	0.85±0.03	**0.91**±0.03	0.79±0.03	4.57±0.88
			DF-PC	**0.61**±0.06	0.98±0.02	**0.46**±0.06	**8.70**±1.04	**0.85**±0.03	0.91±0.03	**0.80**±0.03	**4.50**±0.88

Table 2: Computational Efficiency for $|\mathbf{V}| = 10$ with 95% confidence interval. Cond. CIT denotes conditional independence tests ($|Z| > 0$). Req. denotes the total number of query requests made by the algorithm. OH Time denotes the overhead time excluding CIT time. Time metrics are in seconds.

Graph	D	M	Method	Discrete						Linear SEM					
				Act. CIT	Cond. CIT	Req.	Tot. Time	CIT Time	OH Time	Act. CIT	Cond. CIT	Req.	Tot. Time	CIT Time	OH Time
ER	2	1000	PC-stable	52.97±3.39	7.97±3.39	**98.20**±3.51	0.01±0.00	0.01±0.00	0.00±0.00	167.53±20.90	122.53±20.90	**271.90**±32.52	0.02±0.00	0.02±0.00	0.00±0.00
			DF-PC	**45.30**±0.33	**0.30**±0.33	105.43±12.46	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**129.87**±21.83	**84.87**±21.83	273.73±32.72	**0.02**±0.00	**0.02**±0.00	0.00±0.00
		2000	PC-stable	62.27±8.11	17.27±8.11	**108.07**±8.54	0.01±0.00	0.01±0.00	0.00±0.00	179.53±21.67	134.53±21.67	**289.97**±34.21	0.02±0.00	0.02±0.00	0.00±0.00
			DF-PC	**46.00**±0.71	**1.00**±0.71	115.40±13.59	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**139.57**±22.75	**94.57**±22.75	292.00±34.47	**0.02**±0.00	**0.02**±0.00	0.00±0.00
		5000	PC-stable	78.03±12.71	33.03±12.71	**125.67**±13.59	0.01±0.00	0.01±0.00	0.00±0.00	186.13±22.34	141.13±22.34	**297.03**±35.03	0.02±0.00	0.02±0.00	0.00±0.00
			DF-PC	**49.50**±3.14	**4.50**±3.14	136.67±16.83	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**144.77**±23.86	**99.77**±23.86	301.23±35.58	**0.02**±0.00	**0.01**±0.00	0.00±0.00
		10000	PC-stable	89.40±13.07	44.40±13.07	**138.03**±14.00	0.02±0.01	0.02±0.01	0.00±0.00	192.40±23.35	147.40±23.35	**306.10**±36.35	0.02±0.00	0.02±0.00	0.00±0.00
			DF-PC	**50.67**±3.12	**5.67**±3.12	144.77±14.66	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**150.10**±24.43	**105.10**±24.43	307.30±36.46	**0.02**±0.00	**0.02**±0.00	0.00±0.00
	4	1000	PC-stable	52.23±2.73	7.23±2.73	**97.70**±3.14	0.01±0.00	0.01±0.00	0.00±0.00	479.83±33.90	434.83±33.90	**729.23**±51.94	0.07±0.01	0.06±0.01	0.00±0.00
			DF-PC	**45.67**±0.82	**0.67**±0.82	99.77±4.53	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**399.63**±34.63	**354.63**±34.63	734.63±51.87	**0.06**±0.01	**0.05**±0.00	0.01±0.00
		2000	PC-stable	61.27±4.96	16.27±4.96	**107.33**±5.54	0.01±0.00	0.01±0.00	0.00±0.00	574.27±36.08	529.27±36.08	**869.73**±57.88	0.07±0.01	0.07±0.01	0.00±0.00
			DF-PC	**46.37**±1.07	**1.37**±1.07	117.57±10.41	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**487.80**±40.48	**442.80**±40.48	873.60±58.49	**0.07**±0.01	**0.06**±0.01	0.01±0.00
		5000	PC-stable	91.37±11.71	46.37±11.71	**141.10**±13.58	0.02±0.00	0.02±0.00	0.00±0.00	718.40±44.83	673.40±44.83	**1058.87**±67.83	0.07±0.01	0.07±0.01	0.00±0.00
			DF-PC	**55.10**±6.58	**10.10**±6.58	174.07±25.89	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**606.43**±45.42	**561.43**±45.42	1060.63±67.64	**0.07**±0.01	**0.06**±0.00	0.01±0.00
		10000	PC-stable	135.73±21.54	90.73±21.54	**190.93**±25.00	0.03±0.01	0.03±0.01	0.00±0.00	871.40±62.95	826.40±62.95	**1271.90**±94.32	0.09±0.01	0.09±0.01	0.00±0.00
			DF-PC	**77.83**±23.05	**32.83**±23.05	242.97±50.59	**0.02**±0.01	**0.02**±0.01	0.00±0.00	**746.13**±66.51	**701.13**±66.51	1276.90±95.91	**0.09**±0.01	**0.07**±0.01	0.02±0.00
SF	2	1000	PC-stable	47.40±1.39	2.40±1.39	**92.40**±1.39	0.01±0.00	0.01±0.00	0.00±0.00	162.80±19.54	117.80±19.54	**236.73**±18.92	0.02±0.00	0.02±0.00	0.00±0.00
			DF-PC	**45.00**±0.00	**0.00**±0.00	**92.40**±1.39	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**99.03**±11.58	**54.03**±11.58	237.97±18.60	**0.02**±0.00	**0.01**±0.00	0.00±0.00
		2000	PC-stable	50.23±1.79	5.23±1.79	**95.33**±1.79	0.01±0.00	0.01±0.00	0.00±0.00	171.20±18.35	126.20±18.35	**245.50**±18.03	0.02±0.00	0.02±0.00	0.00±0.00
			DF-PC	**45.10**±0.20	**0.10**±0.20	96.40±2.74	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**103.97**±13.18	**58.97**±13.18	249.50±19.53	**0.01**±0.00	**0.01**±0.00	0.00±0.00
		5000	PC-stable	54.77±2.77	9.77±2.77	**99.90**±2.77	0.01±0.00	0.01±0.00	0.00±0.00	177.57±18.50	132.57±18.50	**251.57**±18.02	0.02±0.00	0.02±0.00	0.00±0.00
			DF-PC	**45.20**±0.27	**0.20**±0.27	104.57±5.85	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**104.70**±13.06	**59.70**±13.06	252.50±17.98	**0.01**±0.00	**0.01**±0.00	0.00±0.00
		10000	PC-stable	63.80±5.19	18.80±5.19	**109.20**±5.31	0.01±0.00	0.01±0.00	0.00±0.00	174.43±18.24	129.43±18.24	**248.17**±17.58	0.02±0.00	0.02±0.00	0.00±0.00
			DF-PC	**45.40**±0.47	**0.40**±0.47	114.97±7.73	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**102.63**±11.87	**57.63**±11.87	255.33±26.47	**0.01**±0.00	**0.01**±0.00	0.00±0.00
	4	1000	PC-stable	48.70±1.65	3.70±1.65	**93.87**±1.70	0.01±0.00	0.01±0.00	0.00±0.00	346.03±37.20	301.03±37.20	**477.47**±45.13	0.04±0.00	0.04±0.00	0.00±0.00
			DF-PC	**45.20**±0.27	**0.20**±0.27	95.30±2.57	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**211.87**±28.39	**166.87**±28.39	500.90±47.80	**0.03**±0.00	**0.02**±0.00	0.01±0.00
		2000	PC-stable	51.43±2.35	6.43±2.35	**96.80**±2.43	0.01±0.00	0.01±0.00	0.00±0.00	471.40±54.39	426.40±54.39	**620.10**±62.83	0.05±0.01	0.05±0.01	0.00±0.00
			DF-PC	**45.40**±0.37	**0.40**±0.37	98.07±3.41	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**275.93**±37.87	**230.93**±37.87	649.47±77.77	**0.04**±0.01	**0.03**±0.00	0.01±0.00
		5000	PC-stable	59.00±5.47	14.00±5.47	**104.87**±5.88	0.01±0.00	0.01±0.00	0.00±0.00	624.73±82.92	579.73±82.92	**790.33**±85.19	0.07±0.01	0.07±0.01	0.00±0.00
			DF-PC	**46.37**±1.21	**1.37**±1.21	111.40±9.76	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**332.70**±47.92	**287.70**±47.92	799.97±84.46	**0.05**±0.01	**0.04**±0.01	0.01±0.00
		10000	PC-stable	70.10±8.28	25.10±8.28	**117.17**±9.23	0.02±0.00	0.02±0.00	0.00±0.00	779.47±136.58	734.47±136.58	**958.37**±133.01	0.09±0.02	0.08±0.02	0.00±0.00
			DF-PC	**47.90**±2.14	**2.90**±2.14	127.00±12.62	**0.01**±0.00	**0.01**±0.00	0.00±0.00	**398.50**±56.01	**353.50**±56.01	978.83±131.29	**0.06**±0.01	**0.04**±0.01	0.02±0.00

Table 3: Learning Performance for $|\mathbf{V}| = 20$ with 95% confidence interval. M corresponds to sample size, and D corresponds to average degree.

Graph	D	M	Method	Discrete				Linear SEM			
				F1	Precision	Recall	SHD	F1	Precision	Recall	SHD
ER	2	1000	PC-stable	0.47±0.04	**0.98**±0.02	0.32±0.03	13.83±0.64	**0.93**±0.02	**0.97**±0.02	**0.89**±0.03	**2.67**±0.84
			DF-PC	**0.55**±0.04	0.82±0.05	**0.43**±0.04	**13.60**±1.05	0.91±0.02	0.95±0.02	**0.89**±0.03	3.27±0.90
		2000	PC-stable	0.63±0.04	**0.96**±0.02	0.47±0.04	10.87±0.91	**0.95**±0.02	**0.98**±0.02	0.92±0.02	**2.10**±0.61
			DF-PC	**0.71**±0.04	0.86±0.03	**0.61**±0.04	**9.83**±1.07	0.93±0.02	0.95±0.02	**0.92**±0.02	2.63±0.67
		5000	PC-stable	0.80±0.04	**0.99**±0.01	0.68±0.05	6.43±0.95	**0.96**±0.02	**0.98**±0.02	**0.94**±0.02	**1.57**±0.67
			DF-PC	**0.85**±0.03	0.92±0.02	**0.79**±0.04	**5.57**±0.89	0.95±0.02	0.96±0.02	**0.94**±0.02	1.97±0.72
		10000	PC-stable	0.87±0.03	**1.00**±0.01	0.78±0.04	4.37±0.85	**0.97**±0.01	**0.99**±0.01	0.95±0.02	**1.03**±0.44
			DF-PC	**0.89**±0.03	0.93±0.02	**0.86**±0.04	**4.10**±0.95	0.96±0.01	0.97±0.01	**0.96**±0.02	1.43±0.47
	4	1000	PC-stable	0.26±0.02	**0.92**±0.04	0.15±0.02	34.37±0.77	0.64±0.02	**0.88**±0.03	0.50±0.02	**22.63**±1.32
			DF-PC	**0.35**±0.03	0.84±0.05	**0.22**±0.02	**32.73**±1.14	**0.64**±0.02	0.87±0.03	**0.51**±0.02	22.67±1.33
		2000	PC-stable	0.37±0.03	**0.99**±0.01	0.23±0.02	30.90±0.80	**0.66**±0.02	**0.87**±0.03	0.54±0.03	**21.73**±1.38
			DF-PC	**0.48**±0.03	0.90±0.03	**0.33**±0.03	**28.13**±1.19	0.66±0.02	0.86±0.02	**0.54**±0.03	21.83±1.39
		5000	PC-stable	0.51±0.02	**0.99**±0.01	0.35±0.02	26.23±0.85	0.69±0.03	**0.87**±0.03	0.58±0.03	20.37±1.66
			DF-PC	**0.64**±0.02	0.96±0.01	**0.48**±0.02	**21.67**±1.03	**0.69**±0.03	0.87±0.03	**0.58**±0.03	**20.33**±1.66
		10000	PC-stable	0.62±0.02	**1.00**±0.00	0.45±0.02	21.83±0.76	0.70±0.03	0.84±0.03	0.61±0.03	20.43±1.80
			DF-PC	**0.73**±0.02	0.95±0.02	**0.60**±0.03	**17.50**±1.25	**0.70**±0.03	**0.84**±0.03	**0.61**±0.03	**20.33**±1.80
SF	2	1000	PC-stable	0.32±0.05	**0.84**±0.06	0.21±0.04	**15.70**±0.73	**0.91**±0.03	**0.98**±0.01	0.87±0.05	**2.87**±0.96
			DF-PC	**0.34**±0.06	0.67±0.07	**0.24**±0.04	16.50±1.01	0.91±0.03	0.95±0.02	**0.88**±0.05	3.10±1.11
		2000	PC-stable	0.42±0.05	**0.93**±0.05	0.28±0.04	**14.07**±0.85	**0.95**±0.03	**0.98**±0.01	0.92±0.04	**1.83**±0.84
			DF-PC	**0.44**±0.05	0.81±0.06	**0.31**±0.04	14.50±1.09	0.93±0.03	0.95±0.02	**0.92**±0.04	2.40±0.89
		5000	PC-stable	0.55±0.05	**0.94**±0.04	0.40±0.05	**11.83**±0.95	**0.97**±0.02	**0.99**±0.01	**0.95**±0.03	**1.13**±0.60
			DF-PC	**0.55**±0.05	0.84±0.04	**0.43**±0.05	12.50±0.91	0.96±0.02	0.97±0.02	**0.95**±0.03	1.53±0.67
		10000	PC-stable	0.62±0.05	**0.98**±0.02	0.46±0.05	**10.33**±0.90	**0.98**±0.01	**0.99**±0.01	**0.98**±0.02	**0.63**±0.46
			DF-PC	**0.63**±0.05	0.86±0.03	**0.52**±0.06	10.73±1.08	0.97±0.01	0.97±0.01	**0.98**±0.02	0.97±0.48
	4	1000	PC-stable	0.20±0.03	**0.83**±0.09	0.12±0.02	31.60±0.84	0.66±0.04	**0.87**±0.03	0.54±0.04	**18.90**±1.99
			DF-PC	**0.24**±0.04	0.73±0.08	**0.15**±0.02	**31.53**±0.95	**0.67**±0.04	0.85±0.03	**0.55**±0.04	19.07±2.04
		2000	PC-stable	0.30±0.03	**0.92**±0.04	0.18±0.02	29.20±0.78	0.70±0.03	**0.86**±0.03	0.59±0.04	**17.80**±1.90
			DF-PC	**0.33**±0.03	0.83±0.04	**0.21**±0.02	**29.13**±0.81	**0.70**±0.03	0.85±0.03	**0.59**±0.04	17.90±1.91
		5000	PC-stable	0.41±0.03	**0.99**±0.01	0.26±0.02	**26.07**±0.81	0.73±0.03	**0.86**±0.03	0.64±0.04	**16.50**±2.02
			DF-PC	**0.44**±0.03	0.87±0.03	**0.30**±0.03	26.20±1.04	**0.73**±0.03	0.85±0.03	**0.64**±0.04	16.53±1.99
		10000	PC-stable	0.47±0.03	**0.99**±0.01	0.31±0.03	24.23±0.91	**0.75**±0.04	**0.85**±0.03	0.67±0.04	**15.83**±2.23
			DF-PC	**0.51**±0.03	0.88±0.03	**0.37**±0.03	**23.83**±1.23	0.75±0.04	0.85±0.03	**0.67**±0.04	15.93±2.23

Table 4: Computational Efficiency for $|\mathbf{V}| = 20$ with 95% confidence interval. Cond. CIT denotes conditional independence tests ($|Z| > 0$). Req. denotes the total number of query requests made by the algorithm. OH Time denotes the overhead time excluding CIT time. Time metrics are in seconds.

Graph	D	M	Method	Discrete						Linear SEM					
				Act. CIT	Cond. CIT	Req.	Tot. Time	CIT Time	OH Time	Act. CIT	Cond. CIT	Req.	Tot. Time	CIT Time	OH Time
ER	2	1000	PC-stable	208.30±4.24	18.30±4.24	**398.83**±4.51	0.03±0.00	0.03±0.00	0.00±0.00	648.20±63.62	458.20±63.62	**1067.30**±119.24	0.08±0.01	0.07±0.01	0.01±0.00
			DF-PC	**190.67**±0.50	**0.67**±0.50	407.63±8.76	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**493.13**±63.45	**303.13**±63.45	1080.40±118.08	**0.06**±0.01	**0.05**±0.01	0.01±0.00
		2000	PC-stable	222.27±5.15	32.27±5.15	**413.23**±5.46	0.03±0.00	0.03±0.00	0.00±0.00	710.57±74.55	520.57±74.55	**1152.10**±134.90	0.09±0.01	0.08±0.01	0.01±0.00
			DF-PC	**191.10**±0.66	**1.10**±0.66	429.80±9.70	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**531.53**±73.37	**341.53**±73.37	1172.03±135.74	**0.08**±0.01	**0.06**±0.01	0.01±0.00
		5000	PC-stable	252.67±7.83	62.67±7.83	**445.47**±8.34	0.05±0.00	0.04±0.00	0.00±0.00	756.67±88.68	566.67±88.68	**1216.13**±147.01	0.09±0.01	0.08±0.01	0.01±0.00
			DF-PC	**194.13**±2.20	**4.13**±2.20	473.57±15.74	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**560.40**±79.50	**370.40**±79.50	1223.87±146.88	**0.07**±0.01	**0.06**±0.01	0.01±0.00
		10000	PC-stable	277.07±9.51	87.07±9.51	**472.53**±10.30	0.06±0.01	0.06±0.00	0.00±0.00	795.13±101.83	605.13±101.83	**1266.43**±160.74	0.09±0.01	0.08±0.01	0.01±0.00
			DF-PC	**198.43**±2.42	**8.43**±2.42	505.47±20.03	**0.04**±0.00	**0.04**±0.00	0.01±0.00	**589.07**±85.08	**399.07**±85.08	1275.07±161.74	**0.07**±0.01	**0.06**±0.01	0.01±0.00
	4	1000	PC-stable	209.90±5.12	19.90±5.12	**400.50**±5.40	0.04±0.00	0.03±0.00	0.00±0.00	2289.20±123.01	2099.20±123.01	**3511.13**±212.25	0.31±0.02	0.29±0.02	0.02±0.00
			DF-PC	**190.70**±0.61	**0.70**±0.61	417.47±15.29	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**1831.33**±132.25	**1641.33**±132.25	3536.07±208.67	**0.28**±0.02	**0.23**±0.02	0.05±0.00
		2000	PC-stable	238.10±10.20	48.10±10.20	**430.17**±10.82	0.04±0.00	0.04±0.00	0.00±0.00	3004.07±183.90	2814.07±183.90	**4441.23**±287.01	**0.38**±0.03	0.36±0.03	0.02±0.00
			DF-PC	**193.43**±2.13	**3.43**±2.13	485.30±43.66	**0.03**±0.00	**0.03**±0.00	0.01±0.00	**2428.27**±189.53	**2238.27**±189.53	4463.03±285.54	0.39±0.04	**0.32**±0.03	0.07±0.01
		5000	PC-stable	292.47±20.45	102.47±20.45	**487.00**±21.54	0.06±0.01	0.05±0.01	0.00±0.00	4054.10±237.16	3864.10±237.16	**5818.97**±371.36	0.51±0.04	0.49±0.04	0.02±0.00
			DF-PC	**198.40**±4.70	**8.40**±4.70	585.50±55.86	**0.04**±0.00	**0.03**±0.00	0.01±0.00	**3310.83**±260.27	**3120.83**±260.27	5829.43±369.69	**0.49**±0.05	**0.40**±0.04	0.08±0.01
		10000	PC-stable	388.27±33.73	198.27±33.73	**588.60**±34.96	0.09±0.01	0.09±0.01	0.00±0.00	4947.00±299.57	4757.00±299.57	**6995.67**±453.83	0.61±0.05	0.58±0.05	0.03±0.00
			DF-PC	**210.03**±6.37	**20.03**±6.37	859.30±186.20	**0.05**±0.01	**0.04**±0.00	0.01±0.00	**4080.80**±319.57	**3890.80**±319.57	7016.90±452.46	**0.58**±0.05	**0.48**±0.05	0.10±0.01
SF	2	1000	PC-stable	196.13±1.88	6.13±1.88	**386.13**±1.88	0.03±0.00	0.03±0.00	0.00±0.00	1274.83±738.95	1084.83±738.95	**1544.73**±734.53	0.15±0.08	0.14±0.07	0.01±0.00
			DF-PC	**190.00**±0.00	**0.00**±0.00	388.40±3.25	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**597.13**±393.79	**407.13**±393.79	1614.53±738.42	**0.11**±0.09	**0.07**±0.05	0.04±0.04
		2000	PC-stable	198.10±2.40	8.10±2.40	**388.20**±2.43	0.03±0.00	0.03±0.00	0.00±0.00	1620.93±793.67	1430.93±793.67	**1895.77**±788.33	0.19±0.11	0.18±0.11	0.01±0.00
			DF-PC	**190.10**±0.20	**0.10**±0.20	390.47±3.71	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**754.20**±429.96	**564.20**±429.96	1918.27±788.50	**0.14**±0.11	**0.08**±0.06	0.06±0.05
		5000	PC-stable	207.77±4.78	17.77±4.78	**398.07**±4.81	0.03±0.00	0.03±0.00	0.00±0.00	2019.80±1022.08	1829.80±1022.08	**2296.20**±1016.97	0.24±0.11	0.22±0.11	0.01±0.00
			DF-PC	**190.30**±0.33	**0.30**±0.33	404.97±8.79	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**977.43**±564.14	**787.43**±564.14	2307.20±1018.36	**0.18**±0.11	**0.11**±0.06	0.07±0.05
		10000	PC-stable	216.47±4.99	26.47±4.99	**406.93**±4.95	0.04±0.00	0.04±0.00	0.00±0.00	2168.87±1009.04	1978.87±1009.04	**2445.23**±1004.08	0.25±0.12	0.24±0.12	0.01±0.00
			DF-PC	**190.60**±0.52	**0.60**±0.52	415.93±7.67	**0.04**±0.00	**0.03**±0.00	0.00±0.00	**1091.77**±588.17	**901.77**±588.17	2546.80±1064.03	**0.18**±0.10	**0.11**±0.06	0.07±0.04
	4	1000	PC-stable	197.80±1.95	7.80±1.95	**387.80**±1.95	0.03±0.00	0.03±0.00	0.00±0.00	1672.40±260.50	1482.40±260.50	**2201.60**±256.92	0.20±0.03	0.19±0.03	0.01±0.00
			DF-PC	**190.00**±0.00	**0.00**±0.00	390.13±3.26	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**871.43**±74.54	**681.43**±74.54	2321.50±367.25	**0.13**±0.01	**0.10**±0.01	0.04±0.01
		2000	PC-stable	203.20±2.95	13.20±2.95	**393.43**±3.08	0.03±0.00	0.03±0.00	0.00±0.00	2549.90±498.91	2359.90±498.91	**3143.47**±490.05	0.31±0.09	0.29±0.08	0.01±0.00
			DF-PC	**190.40**±0.37	**0.40**±0.37	397.93±4.84	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**1191.57**±118.03	**1001.57**±118.03	3410.33±766.15	**0.20**±0.04	**0.14**±0.02	0.06±0.03
		5000	PC-stable	218.33±5.20	28.33±5.20	**409.47**±5.48	0.04±0.00	0.04±0.00	0.00±0.00	3642.80±514.09	3452.80±514.09	**4345.53**±509.31	0.44±0.06	0.42±0.06	0.02±0.00
			DF-PC	**191.20**±0.67	**1.20**±0.67	419.90±9.18	**0.03**±0.00	**0.03**±0.00	0.00±0.00	**1755.93**±202.46	**1565.93**±202.46	4550.67±555.72	**0.31**±0.04	**0.21**±0.02	0.09±0.02
		10000	PC-stable	234.13±8.59	44.13±8.59	**425.70**±9.00	0.05±0.00	0.05±0.00	0.00±0.00	5090.30±898.59	4900.30±898.59	**5895.37**±888.86	0.63±0.10	0.60±0.10	0.02±0.00
			DF-PC	**191.80**±0.92	**1.80**±0.92	448.13±15.20	**0.04**±0.00	**0.04**±0.00	0.01±0.00	**2345.27**±332.86	**2155.27**±332.86	5992.80±884.81	**0.42**±0.07	**0.28**±0.04	0.14±0.04

Table 5: Learning Performance for $|\mathbf{V}| = 30$ with 95% confidence interval. M corresponds to sample size, and D corresponds to average degree.

Graph	D	M	Method	Discrete				Linear SEM			
				F1	Precision	Recall	SHD	F1	Precision	Recall	SHD
ER	2	1000	PC-stable	0.47±0.04	**0.93**±0.04	0.32±0.03	**21.10**±1.13	**0.93**±0.02	**0.97**±0.01	0.89±0.02	**4.03**±0.94
			DF-PC	**0.54**±0.04	0.75±0.04	**0.43**±0.04	21.37±1.50	0.92±0.02	0.94±0.02	**0.90**±0.02	4.73±0.99
		2000	PC-stable	0.61±0.04	**0.97**±0.02	0.46±0.04	16.67±1.22	**0.94**±0.01	**0.97**±0.01	0.92±0.02	**3.23**±0.82
			DF-PC	**0.70**±0.04	0.84±0.03	**0.61**±0.04	**15.20**±1.65	0.93±0.02	0.94±0.01	**0.92**±0.02	4.30±0.92
		5000	PC-stable	0.78±0.03	**0.99**±0.01	0.65±0.04	**10.87**±1.08	**0.96**±0.01	**0.98**±0.01	0.95±0.02	**2.03**±0.83
			DF-PC	**0.80**±0.03	0.86±0.02	**0.76**±0.04	11.10±1.29	0.95±0.01	0.96±0.01	**0.95**±0.02	2.70±0.84
		10000	PC-stable	0.86±0.02	**0.99**±0.01	0.77±0.03	**7.13**±1.00	**0.97**±0.01	**0.99**±0.01	**0.96**±0.02	**1.57**±0.72
			DF-PC	**0.87**±0.02	0.88±0.02	**0.86**±0.03	7.70±1.17	0.95±0.01	0.95±0.02	**0.96**±0.02	2.73±0.79
	4	1000	PC-stable	0.24±0.02	**0.94**±0.03	0.14±0.01	52.20±0.79	0.65±0.02	**0.87**±0.02	0.52±0.02	**33.57**±1.96
			DF-PC	**0.34**±0.03	0.80±0.03	**0.22**±0.02	**50.23**±1.18	**0.65**±0.02	0.86±0.02	**0.52**±0.02	33.60±2.01
		2000	PC-stable	0.35±0.02	**0.97**±0.02	0.21±0.02	47.63±1.03	**0.67**±0.02	**0.87**±0.02	0.55±0.02	**32.13**±2.01
			DF-PC	**0.47**±0.03	0.85±0.03	**0.33**±0.02	**43.80**±1.64	0.67±0.02	0.86±0.02	**0.55**±0.02	32.37±2.01
		5000	PC-stable	0.50±0.02	**0.99**±0.01	0.33±0.02	40.23±1.15	**0.69**±0.02	**0.85**±0.02	0.58±0.03	**30.90**±2.08
			DF-PC	**0.61**±0.02	0.89±0.02	**0.47**±0.03	**35.20**±1.64	0.69±0.02	0.85±0.02	**0.59**±0.03	31.07±2.10
		10000	PC-stable	0.59±0.02	**1.00**±0.00	0.42±0.02	34.87±1.17	**0.70**±0.02	**0.84**±0.02	**0.60**±0.03	**30.90**±2.18
			DF-PC	**0.71**±0.02	0.93±0.01	**0.58**±0.02	**28.03**±1.66	0.70±0.02	0.84±0.02	**0.60**±0.03	31.00±2.22
SF	2	1000	PC-stable	0.28±0.03	**0.78**±0.07	0.17±0.02	**24.80**±0.84	**0.90**±0.03	**0.98**±0.01	0.84±0.05	**5.07**±1.42
			DF-PC	**0.32**±0.04	0.61±0.06	**0.22**±0.03	26.17±1.19	0.89±0.03	0.93±0.02	**0.85**±0.05	5.90±1.61
		2000	PC-stable	0.40±0.03	**0.87**±0.03	0.26±0.03	**21.90**±0.77	**0.93**±0.03	**0.97**±0.01	0.89±0.04	**3.87**±1.32
			DF-PC	**0.43**±0.04	0.70±0.05	**0.32**±0.03	23.20±1.06	0.91±0.03	0.93±0.02	**0.90**±0.04	4.83±1.47
		5000	PC-stable	**0.51**±0.05	**0.91**±0.04	0.36±0.04	**19.00**±1.24	**0.95**±0.03	**0.98**±0.01	**0.93**±0.04	**2.63**±1.28
			DF-PC	0.51±0.04	0.72±0.04	**0.40**±0.04	21.40±1.26	0.93±0.03	0.95±0.02	**0.93**±0.04	3.50±1.29
		10000	PC-stable	**0.57**±0.04	**0.92**±0.03	0.42±0.04	**17.33**±1.25	**0.96**±0.02	**0.99**±0.01	**0.94**±0.04	**1.97**±1.11
			DF-PC	0.56±0.04	0.75±0.05	**0.45**±0.04	19.73±1.56	0.94±0.02	0.95±0.01	**0.94**±0.04	3.20±0.96
	4	1000	PC-stable	0.24±0.02	**0.89**±0.04	0.14±0.01	**45.97**±1.19	0.69±0.03	**0.90**±0.02	0.56±0.03	**26.03**±2.20
			DF-PC	**0.28**±0.02	0.70±0.04	**0.17**±0.02	47.23±1.42	**0.69**±0.03	0.88±0.03	**0.58**±0.03	26.47±2.39
		2000	PC-stable	0.32±0.03	**0.93**±0.03	0.19±0.02	**43.03**±1.40	**0.71**±0.03	**0.89**±0.03	0.60±0.03	**25.03**±2.20
			DF-PC	**0.34**±0.03	0.72±0.04	**0.23**±0.02	44.93±1.75	0.71±0.03	0.87±0.03	**0.61**±0.03	25.57±2.25
		5000	PC-stable	0.41±0.03	**0.97**±0.02	0.27±0.03	**38.83**±1.78	0.75±0.03	**0.88**±0.03	0.65±0.03	**22.93**±2.59
			DF-PC	**0.44**±0.03	0.80±0.03	**0.31**±0.03	40.20±2.06	**0.75**±0.03	0.88±0.03	**0.66**±0.03	**22.93**±2.47
		10000	PC-stable	0.47±0.03	**0.98**±0.01	0.32±0.03	**36.13**±1.64	**0.76**±0.03	**0.87**±0.03	0.67±0.03	**22.80**±2.67
			DF-PC	**0.51**±0.03	0.85±0.03	**0.37**±0.03	36.37±1.82	0.75±0.03	0.86±0.03	**0.67**±0.03	22.93±2.60

Table 6: Computational Efficiency for $|\mathbf{V}| = 30$ with 95% confidence interval. Cond. CIT denotes conditional independence tests ($|Z| > 0$). Req. denotes the total number of query requests made by the algorithm. OH Time denotes the overhead time excluding CIT time. Time metrics are in seconds.

Graph	D	M	Method	Discrete						Linear SEM					
				Act. CIT	Cond. CIT	Req.	Tot. Time	CIT Time	OH Time	Act. CIT	Cond. CIT	Req.	Tot. Time	CIT Time	OH Time
ER	2	1000	PC-stable	466.37±4.37	31.37±4.37	**902.07**±4.63	0.08±0.00	0.07±0.00	0.01±0.00	1335.50±124.52	900.50±124.52	**2221.60**±211.63	0.16±0.02	0.15±0.02	0.01±0.00
			DF-PC	**435.90**±0.57	**0.90**±0.57	924.57±13.97	**0.07**±0.01	**0.06**±0.01	0.01±0.00	**1018.43**±115.78	**583.43**±115.78	2246.57±209.12	**0.14**±0.02	**0.11**±0.01	0.03±0.00
		2000	PC-stable	492.63±7.71	57.63±7.71	**929.57**±8.22	0.08±0.01	0.08±0.01	0.01±0.00	1492.20±156.33	1057.20±156.33	**2424.17**±250.09	0.18±0.03	0.17±0.02	0.01±0.00
			DF-PC	**437.30**±0.96	**2.30**±0.96	992.43±35.94	**0.07**±0.00	**0.06**±0.00	0.01±0.00	**1108.93**±143.84	**673.93**±143.84	2455.20±250.77	**0.15**±0.02	**0.12**±0.02	0.03±0.01
		5000	PC-stable	540.97±14.76	105.97±14.76	**980.03**±15.43	0.10±0.01	0.09±0.01	0.01±0.00	1621.73±169.90	1186.73±169.90	**2606.37**±270.32	0.19±0.02	0.17±0.02	0.01±0.00
			DF-PC	**440.77**±1.84	**5.77**±1.84	1058.07±39.39	**0.08**±0.01	**0.07**±0.00	0.01±0.00	**1209.10**±154.36	**774.10**±154.36	2635.23±269.15	**0.15**±0.02	**0.13**±0.02	0.03±0.00
		10000	PC-stable	588.73±24.12	153.73±24.12	**1031.07**±25.15	0.13±0.01	0.12±0.01	0.01±0.00	1709.10±202.67	1274.10±202.67	**2712.70**±309.38	0.19±0.02	0.17±0.02	0.01±0.00
			DF-PC	**445.00**±3.22	**10.00**±3.22	1135.13±58.89	**0.10**±0.01	**0.08**±0.00	0.01±0.00	**1258.37**±179.95	**823.37**±179.95	2739.33±314.55	**0.16**±0.02	**0.13**±0.02	0.03±0.00
	4	1000	PC-stable	469.90±6.62	34.90±6.62	**905.73**±6.80	0.08±0.01	0.07±0.00	0.01±0.00	5076.37±272.66	4641.37±272.66	**7826.20**±476.07	0.70±0.08	0.66±0.07	0.04±0.00
			DF-PC	**436.00**±0.65	**1.00**±0.65	932.77±15.01	**0.07**±0.01	**0.06**±0.00	0.01±0.00	**3840.13**±270.80	**3405.13**±270.80	7875.17±470.43	**0.62**±0.07	**0.52**±0.06	0.10±0.01
		2000	PC-stable	505.83±10.97	70.83±10.97	**942.27**±11.30	0.09±0.01	0.08±0.01	0.01±0.00	6533.43±329.86	6098.43±329.86	**9787.90**±547.65	0.90±0.06	0.86±0.06	0.05±0.00
			DF-PC	**436.80**±1.08	**1.80**±1.08	1038.50±46.88	**0.08**±0.01	**0.07**±0.00	0.01±0.00	**5032.63**±335.65	**4597.63**±335.65	9834.03±544.27	**0.81**±0.07	**0.67**±0.06	0.15±0.01
		5000	PC-stable	612.57±37.60	177.57±37.60	**1053.27**±39.14	0.12±0.01	0.11±0.01	0.01±0.00	8969.40±415.59	8534.40±415.59	**12934.87**±661.71	1.18±0.10	1.12±0.10	0.06±0.00
			DF-PC	**445.50**±5.75	**10.50**±5.75	1401.87±206.83	**0.09**±0.01	**0.07**±0.01	0.02±0.01	**7031.07**±443.94	**6596.07**±443.94	12985.00±662.99	**1.07**±0.09	**0.88**±0.08	0.19±0.02
		10000	PC-stable	754.80±82.97	319.80±82.97	**1201.43**±85.88	0.19±0.03	0.18±0.03	0.01±0.00	11143.53±521.00	10708.53±521.00	**15691.77**±812.16	1.32±0.12	1.26±0.11	0.06±0.00
			DF-PC	**462.87**±17.79	**27.87**±17.79	1579.07±183.81	**0.11**±0.01	**0.09**±0.01	0.02±0.00	**8806.87**±565.83	**8371.87**±565.83	15735.40±810.87	**1.22**±0.11	**1.01**±0.09	0.21±0.02
SF	2	1000	PC-stable	445.13±2.47	10.13±2.47	**880.13**±2.47	0.07±0.01	0.07±0.01	0.01±0.00	2556.57±872.10	2121.57±872.10	**3111.93**±867.08	0.30±0.11	0.29±0.10	0.02±0.00
			DF-PC	**435.00**±0.00	**0.00**±0.00	884.53±4.47	**0.07**±0.01	**0.06**±0.00	0.01±0.00	**1150.57**±440.04	**715.57**±440.04	3693.63±1217.71	**0.26**±0.14	**0.15**±0.08	0.11±0.06
		2000	PC-stable	450.80±3.23	15.80±3.23	**885.87**±3.27	0.07±0.00	0.07±0.00	0.01±0.00	3476.93±1079.76	3041.93±1079.76	**4045.20**±1072.29	0.40±0.12	0.38±0.11	0.02±0.00
			DF-PC	**435.10**±0.20	**0.10**±0.20	895.00±7.21	**0.07**±0.00	**0.06**±0.00	0.01±0.00	**1430.67**±561.36	**995.67**±561.36	4476.40±1199.95	**0.30**±0.12	**0.16**±0.06	0.14±0.06
		5000	PC-stable	464.23±5.87	29.23±5.87	**899.80**±5.90	0.08±0.01	0.08±0.00	0.01±0.00	4497.67±1212.21	4062.67±1212.21	**5074.87**±1203.44	0.55±0.15	0.52±0.14	0.02±0.00
			DF-PC	**435.60**±0.44	**0.60**±0.44	921.83±16.40	**0.08**±0.00	**0.07**±0.00	0.01±0.00	**1919.70**±671.22	**1484.70**±671.22	5334.67±1313.96	**0.42**±0.15	**0.23**±0.08	0.19±0.07
		10000	PC-stable	471.80±6.10	36.80±6.10	**908.20**±6.26	0.09±0.00	0.08±0.00	0.01±0.00	4663.20±1193.91	4228.20±1193.91	**5241.93**±1183.43	0.53±0.14	0.51±0.13	0.02±0.00
			DF-PC	**436.60**±0.68	**1.60**±0.68	922.93±8.97	**0.09**±0.00	**0.08**±0.00	0.01±0.00	**2097.53**±737.54	**1662.53**±737.54	5718.47±1373.37	**0.41**±0.16	**0.23**±0.09	0.18±0.07
	4	1000	PC-stable	453.60±4.06	18.60±4.06	**888.83**±4.17	**0.07**±0.01	0.07±0.00	0.01±0.00	3421.53±759.34	2986.53±759.34	**4538.50**±770.31	0.41±0.10	0.39±0.09	0.02±0.00
			DF-PC	**435.30**±0.33	**0.30**±0.33	896.97±7.54	0.08±0.01	**0.07**±0.01	0.01±0.00	**1714.07**±136.20	**1279.07**±136.20	4973.37±856.83	**0.29**±0.05	**0.20**±0.02	0.09±0.04
		2000	PC-stable	464.33±5.65	29.33±5.65	**899.90**±5.85	0.08±0.01	0.07±0.00	0.01±0.00	5415.07±1627.60	4980.07±1627.60	**6694.00**±1633.82	0.65±0.19	0.62±0.18	0.03±0.01
			DF-PC	**435.57**±0.48	**0.57**±0.48	909.83±8.82	**0.08**±0.01	**0.07**±0.01	0.01±0.00	**2417.70**±194.84	**1982.70**±194.84	6994.80±1636.73	**0.43**±0.10	**0.28**±0.03	0.15±0.09
		5000	PC-stable	484.37±9.76	49.37±9.76	**920.43**±10.09	0.09±0.01	0.08±0.01	0.01±0.00	8081.00±1263.05	7646.00±1263.05	**9605.27**±1296.76	1.03±0.20	0.99±0.19	0.04±0.01
			DF-PC	**436.50**±0.78	**1.50**±0.78	952.33±18.24	**0.08**±0.01	**0.07**±0.01	0.01±0.00	**3810.20**±452.20	**3375.20**±452.20	9850.43±1334.75	**0.66**±0.12	**0.46**±0.07	0.21±0.07
		10000	PC-stable	505.67±11.16	70.67±11.16	**942.80**±11.73	0.11±0.01	0.10±0.01	0.01±0.00	11660.00±2018.18	11225.00±2018.18	**13368.53**±2023.36	1.41±0.25	1.36±0.24	0.06±0.01
			DF-PC	**438.27**±1.81	**3.27**±1.81	990.77±27.99	**0.09**±0.01	**0.08**±0.01	0.01±0.00	**5202.20**±691.91	**4767.20**±691.91	13791.93±2011.78	**0.92**±0.15	**0.61**±0.09	0.30±0.08

C.2.2 Nonlinear Experiments Results

Table 7: Nonlinear Causal Discovery Performance using KCI Test ($|\mathbf{V}| = 10, M = 1000$). Mean with 95% confidence interval.

Graph	Method	F1	Precision	Recall	SHD	Cond. CIT	Tot. Time (s)
ER	PC-stable	0.85±0.05	0.97±0.02	0.77±0.06	2.50±0.61	77.33±14.44	66.37±11.70
	DF-PC	**0.86**±0.05	0.94±0.02	0.80±0.06	2.47±0.61	**43.90**±11.56	**39.00**±9.48
SF	PC-stable	**0.85**±0.05	0.99±0.01	0.77±0.07	2.07±0.59	52.50±10.56	45.26±8.97
	DF-PC	0.85±0.05	0.97±0.02	0.78±0.07	2.10±0.65	**15.47**±5.72	**12.95**±4.20

Efficiency in Computationally Expensive (Nonlinear) Testing Table 7 presents the results for nonlinear additive noise models using the KCI test. The high computational cost of KCI highlights the practical advantage of DF-PC's deductive framework. In these high-latency CIT regimes, the total execution time is predominantly determined by the number of statistical tests rather than algorithmic complexity. DF-PC reduced the total execution time by up to 71% in Scale-Free networks (from 45.26s to 12.95s), directly corresponding to a proportional reduction in KCI calls. Notably, the computational overhead introduced by the recursive logical inference remains negligible compared to the massive efficiency gains achieved by pruning redundant CITs. This makes DF-PC particularly promising for modern causal discovery tasks where complex data structures necessitate the use of heavy kernel-based or exact tests, as it maintains structural accuracy while drastically lowering the computational barrier.

C.2.3 Best- and Worst- Case Scenario Analysis

(a) Collider Hub (Best) (b) Source Hub (Worst)

Figure 5: Structural motifs used for boundary analysis.

Table 8: Performance on Extreme Causal Hubs ($|\mathbf{V}| = 11, M = 2000$).

Structure	Method	F1	Total CIT	Time (s)
Collider-Hub	PC-stable	1.00	5165.10	0.60
	DF-PC	1.00	**55.30**	0.13
Source-Hub	PC-stable	1.00	**5210.80**	0.57
	DF-PC	1.00	**5210.80**	0.87

Performance under Structural Extremes To identify the theoretical and empirical boundary conditions of DF-PC, we evaluated the algorithm on two extreme topologies (Fig. 5 and Table 8).

Collider Hub (Best Case): As theoretically predicted, DF-PC achieves a near-total reduction in statistical query complexity. While PC-stable requires 5,165 tests to exhaustively search all conditioning sets in this dense structure, DF-PC identifies the identical skeleton using only 55.3 tests—a 98.9% reduction.

Source Hub (Failure Case): In environments where the discovered independence relations are structurally insufficient to trigger deductive inference (i.e., the absence of prerequisite outcomes for our deduction rules), DF-PC's execution naturally converges to that of PC-stable. In this Source-Hub scenario, both algorithms perform the same number of statistical tests (5,210.8). Consequently, as shown in Table 8, DF-PC incurs a marginal increase in total execution time (from 0.57s to 0.87s). This reflects the inherent logical overhead of evaluating potential deductions before falling back to statistical testing in a query-exhausted environment. This demonstrates that while DF-PC provides massive gains in deducible environments, it maintains robust performance by gracefully reverting to standard procedures, albeit with a minor computational cost for the logic engine.

C.2.4 Sensitivity to Significance Level

Table 9: Sensitivity Analysis of Significance Level α ($|\mathbf{V}| = 40, M = 5000$).

α	Method	F1	Total CIT	Time (s)
0.05	PC-stable	**0.93**	3090.00	0.44
	DF-PC	0.87	**1976.40**	0.34
0.01	PC-stable	**0.95**	2674.90	0.42
	DF-PC	0.94	**1769.40**	0.29
0.001	PC-stable	**0.96**	2485.60	0.32
	DF-PC	0.95	**1701.20**	0.24
0.0001	PC-stable	**0.96**	2373.40	0.33
	DF-PC	**0.96**	**1651.00**	0.21

Stability across Significance Levels (α) The sensitivity analysis (Table 9) reveals that DF-PC consistently maintains its efficiency advantage across all tested levels of α. While both PC-stable and DF-PC are susceptible to False Positive (FP) errors under finite samples, the impact of FPs can be pronounced in a deductive framework [Faller and Janzing, 2025]. In DF-PC, each individual CIT outcome carries significantly more "weight" as it also serves as a logical premise for multiple downstream inferences; thus, a single spurious result can potentially "poison" the deductive chain. Consequently, applying conservative significance levels (e.g., $\alpha = 0.001$ or 0.0001) is not only reasonable in the context of multiple testing in structure learning [Li and Wang, 2009, Strobl et al., 2019] but also critical for maintaining logical consistency of deductive framework. As shown in Table 9, at the most stringent level ($\alpha = 0.0001$), DF-PC achieves its peak F1-score (0.96) while delivering a 30.4% reduction in CIT calls. These observations suggest that the deductive approach of DF-PC can be effectively paired with strict statistical thresholds to improve both performance and efficiency of discovery tasks in various settings.

C.2.5 Scalability Experiments

Table 10: Scalability Benchmark for $|\mathbf{V}| = 100$ (Linear SEM, $M = 1000, \alpha = 0.001$). Mean with 95% confidence interval over 30 trials.

Graph	D	Method	F1	Precision	Recall	SHD	Act. CIT	Cond. CIT	Req.	Tot. Time
ER	2	PC-stable	**0.94**±0.01	0.99±0.00	0.90±0.02	**11.20**±1.66	9212.93±377.98	4262.93±377.98	15822.53±571.06	1.18±0.08
		DF-PC	0.94±0.01	0.97±0.01	0.90±0.01	12.13±1.56	**7268.70**±272.59	**2318.70**±272.59	15930.43±571.73	**1.11**±0.08
	4	PC-stable	0.68±0.01	0.91±0.01	0.55±0.02	101.53±3.82	40839.73±2016.52	35889.73±2016.52	62272.03±3331.70	5.62±0.51
		DF-PC	**0.69**±0.01	0.90±0.01	0.56±0.02	**100.77**±3.87	**26069.37**±1543.18	**21119.37**±1543.18	62614.47±3337.61	**3.79**±0.31
SF	2	PC-stable	0.87±0.02	0.99±0.00	0.78±0.03	19.87±2.69	9643.13±1311.04	4693.13±1311.04	14941.07±1300.14	1.52±0.25
		DF-PC	**0.88**±0.02	0.96±0.01	0.81±0.03	**19.83**±2.61	**5798.17**±219.72	**848.17**±219.72	16401.40±1482.66	**1.28**±0.23
	4	PC-stable	0.71±0.02	0.96±0.01	0.56±0.02	77.47±4.44	14566.50±899.23	9616.50±899.23	22448.10±969.26	1.78±0.11
		DF-PC	**0.72**±0.02	0.94±0.01	0.58±0.02	**76.67**±4.42	**9179.07**±377.23	**4229.07**±377.23	23143.07±1058.00	**1.31**±0.07

As the number of variables increases to $N = 100$, the search space for conditioning sets grows exponentially, posing a significant challenge to both the efficiency and accuracy of constraint-based algorithms. Table 10 summarizes the results for $|\mathbf{V}| = 100$ variables under a strict significance level ($\alpha = 0.001$).

Sustained Efficiency Gains in High Dimensions DF-PC demonstrates remarkable scalability, particularly in denser graph regimes. For Erdős-Rényi (ER) graphs with $D = 4$, DF-PC reduced the total number of actual CIT calls from $40,839$ to $26,069$, representing a **36.1% reduction**. Similar gains were observed in Scale-Free (SF) networks, with a **37.0% reduction** in CIT calls ($14,566 \rightarrow 9,179$) for $D = 4$. This reduction is primarily driven by the deduction engine's ability to prune redundant high-order conditional queries, which are computationally expensive.

Execution Time and Computational Overhead Despite the increased complexity of managing the deduction cache and logical rules for 100 variables, the total execution time of DF-PC remained consistently lower than that of the baseline. In the $D = 4$ ER case, the total wall-clock time was reduced by **32.6%** (from $5.62s$ to $3.79s$). The marginal overhead of the deduction logic is effectively amortized by the significant savings in statistical testing time, showing that DF-PC can be better suited for large-scale causal discovery.

Structural Accuracy and Stability In terms of structural recovery, DF-PC maintains near-perfect parity with or slightly outperforms the baseline. For $D = 4$ ER graphs, DF-PC yielded an F1-score of 0.69 compared to the baseline's 0.68, and achieved a similar SHD (100.77 vs 101.53). At the strict significance level of $\alpha = 0.001$, the deduced-first strategy successfully avoided the accumulation of statistical errors that typically plague large-scale discovery, demonstrating that logical consistency remains a reliable constraint even in high-dimensional search spaces.

C.2.6 Semi-Synthetic Data Experiments

Table 11: Semi-Synthetic Network Benchmarks: Barley ($|\mathbf{V}| = 48$) and Mildew ($|\mathbf{V}| = 35$). Mean with 95% confidence interval over 30 trials.

Dataset	M	Method	F1	Precision	Recall	SHD	Act. CIT	Cond. CIT	Req.	Tot. Time
Barley	2000	PC-stable	**0.68**$\pm$0.01	0.97$\pm$0.01	0.52$\pm$0.01	**41.60**$\pm$0.77	6675.07$\pm$103.29	5547.07$\pm$103.29	10018.27$\pm$137.85	1.37$\pm$0.04
		DF-PC	0.67$\pm$0.01	0.91$\pm$0.01	0.53$\pm$0.01	44.07$\pm$1.00	**4904.90**$\pm$77.78	**3776.90**$\pm$77.78	10669.93$\pm$156.13	**1.14**$\pm$0.02
	5000	PC-stable	**0.76**$\pm$0.01	0.97$\pm$0.00	0.62$\pm$0.01	**33.27**$\pm$0.78	11007.63$\pm$173.99	9879.63$\pm$173.99	16162.27$\pm$236.19	2.63$\pm$0.06
		DF-PC	0.75$\pm$0.01	0.93$\pm$0.01	0.63$\pm$0.01	35.13$\pm$0.85	**8404.03**$\pm$151.42	**7276.03**$\pm$151.42	16892.50$\pm$266.95	**2.17**$\pm$0.06
Mildew	2000	PC-stable	0.77$\pm$0.01	0.94$\pm$0.01	0.65$\pm$0.01	17.83$\pm$0.79	2504.03$\pm$65.44	1909.03$\pm$65.44	4248.80$\pm$86.13	2.65$\pm$0.64
		DF-PC	**0.79**$\pm$0.01	0.89$\pm$0.02	0.71$\pm$0.01	**17.27**$\pm$1.09	**2105.13**$\pm$54.55	**1510.13**$\pm$54.55	4553.83$\pm$98.40	**2.21**$\pm$0.55
	5000	PC-stable	0.85$\pm$0.01	0.97$\pm$0.01	0.75$\pm$0.01	**12.30**$\pm$0.53	3521.70$\pm$60.58	2926.70$\pm$60.58	5670.33$\pm$78.76	4.70$\pm$0.88
		DF-PC	**0.85**$\pm$0.01	0.93$\pm$0.01	0.79$\pm$0.01	12.37$\pm$0.67	**2838.97**$\pm$49.74	**2243.97**$\pm$49.74	5964.33$\pm$94.79	**4.10**$\pm$0.76

We evaluate the practical performance of DF-PC on two well-known benchmarks with complex structural dependencies: *Barley* (48 nodes, 84 edges) and *Mildew* (35 nodes, 46 edges). Table 11 summarizes the performance across different sample sizes ($M \in \{2000, 5000\}$).

Efficiency in Semi-Synthetic Environments Across all real-world configurations, DF-PC consistently reduces the computational burden of causal discovery. In the *Barley* benchmark, DF-PC achieved a significant reduction in statistical query complexity, dropping the actual CIT calls by 26.5% for $M = 2000$ (from 6,675 to 4,904) and 23.6% for $M = 5000$. This reduction in statistical overhead translated into a 16–17% improvement in total execution time. Similarly, in the *Mildew* dataset, the number of statistical tests was reduced by approximately 16% to 19%. These results confirm that the "Deduced-First" strategy remains effective in pruning redundant tests even when variable cardinalities and structural complexities increase.

Structural Accuracy under Realistic Dependencies The structural discovery performance of DF-PC is generally comparable to that of the baseline PC-stable. In the *Mildew* dataset ($M = 2000$), DF-PC showed a marginal improvement in F1-score (0.79 vs 0.77) and SHD (17.27 vs 17.83), suggesting that logical consistency constraints can occasionally prevent premature edge deletions under certain sample distributions. However, in the *Barley* dataset, DF-PC yielded slightly higher SHD values compared to the baseline. These observations indicate that while logical deduction efficiently reduces the test budget, its accuracy is closely tied to the reliability of the initial statistical findings in complex categorical domains. Importantly, DF-PC maintains a narrow performance gap relative to the baseline across both datasets, demonstrating its utility as a more efficient alternative that reaches a similar structural maturity with a significantly lower CIT budget.

C.2.7 Baseline Comparison with PC with deduce-dep Results

The baseline comparison results involving PC-stable, PC with DEDUCE-DEP, and DF-PC are illustrated in Figure 6 (F1 score comparison) and Figure 7 (count of performed CITs).

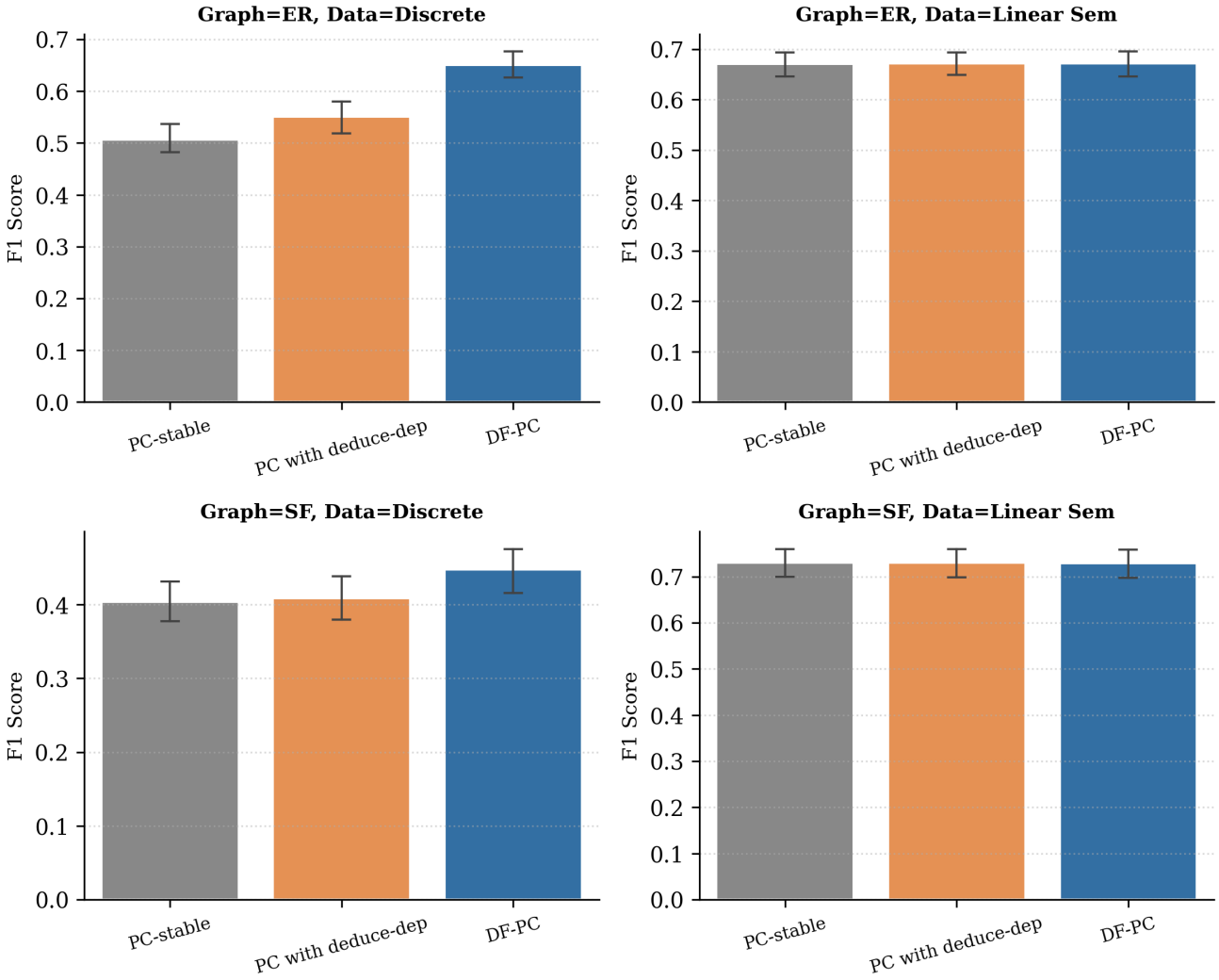

Figure 6: **F1 Score Comparison.** F1 score for various algorithms across ER and SF graph topologies with discrete and linear Gaussian data. The experiments are conducted with 20 nodes, 5000 samples, and an average degree of 4. Bars represent the mean across 30 independent trials with unique random seeds, and error bars indicate the 95% confidence interval (CI).

Analysis of Baseline Comparison As shown in Figures 6 and 7, DF-PC significantly outperforms both PC-stable and PC with DEDUCE-DEP in terms of the number of bypassed CITs. Specifically, for discrete data, DF-PC bypasses 315.15 CITs on average, which is about 70% more than the 185.13 CITs bypassed by PC with DEDUCE-DEP. In linear SEM data, DF-PC bypasses 2,396.88 CITs, nearly doubling the savings of PC with DEDUCE-DEP (1,286.95). In terms of accuracy (Figure 6), DF-PC demonstrates a higher F1-score (0.549) on discrete datasets compared to PC-stable (0.455) and PC with DEDUCE-DEP (0.480), showcasing the robustness of the "Deduced-First" logic against the cascade of statistical errors.

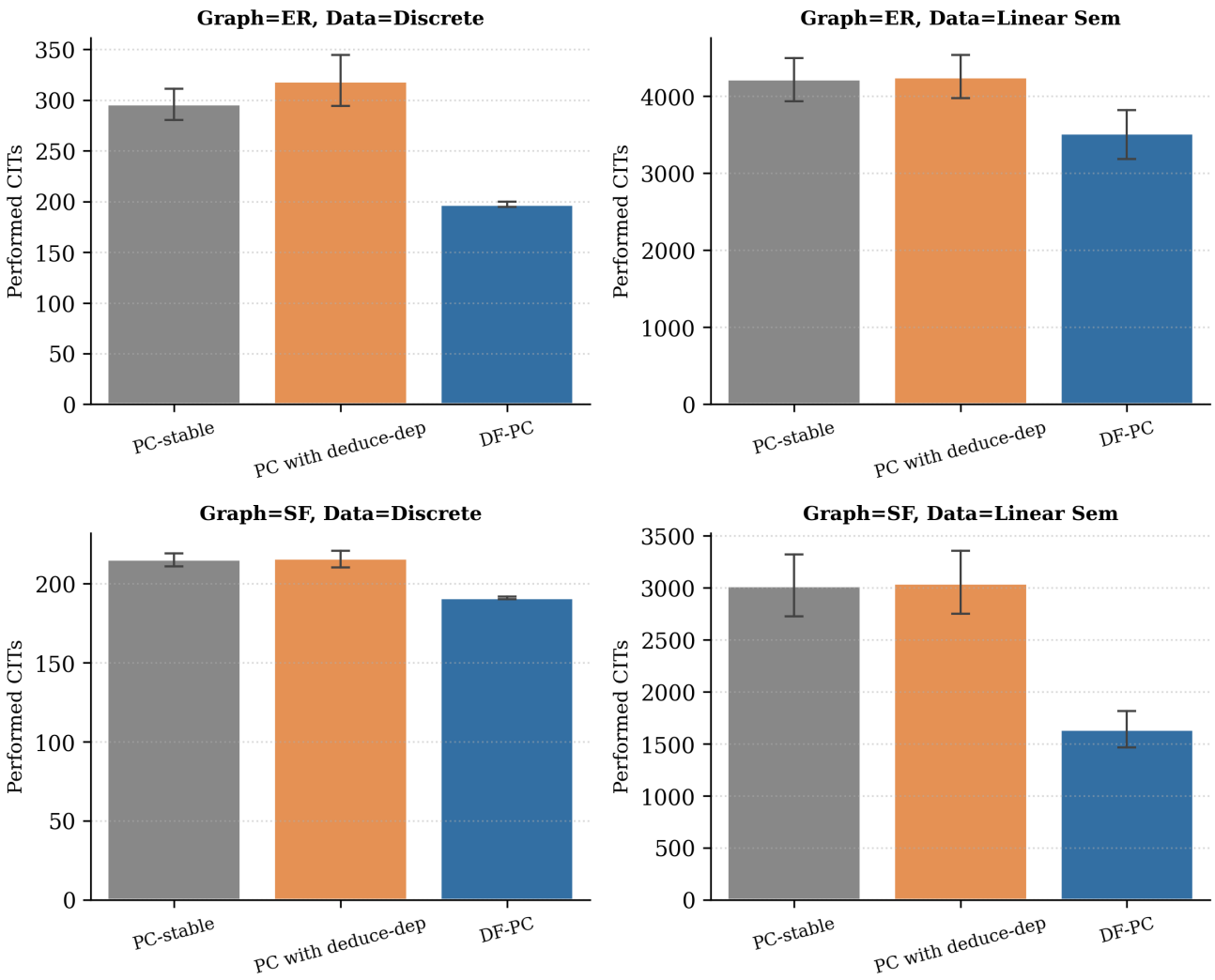

Figure 7: **Count of Performed CITs.** The number of Conditional Independence Tests (CITs) executed by each algorithm. The setup uses 20 nodes, 5000 samples, and an average degree of 4. Error bars indicate the 95% CI calculated over 30 trials.

C.2.8 High-Density Stress Test Results

To investigate how DF-PC performs under high-density regimes, the F1 score and CIT counts for average degree $D = 6$ are visualized in Figure 8 and Figure 9.

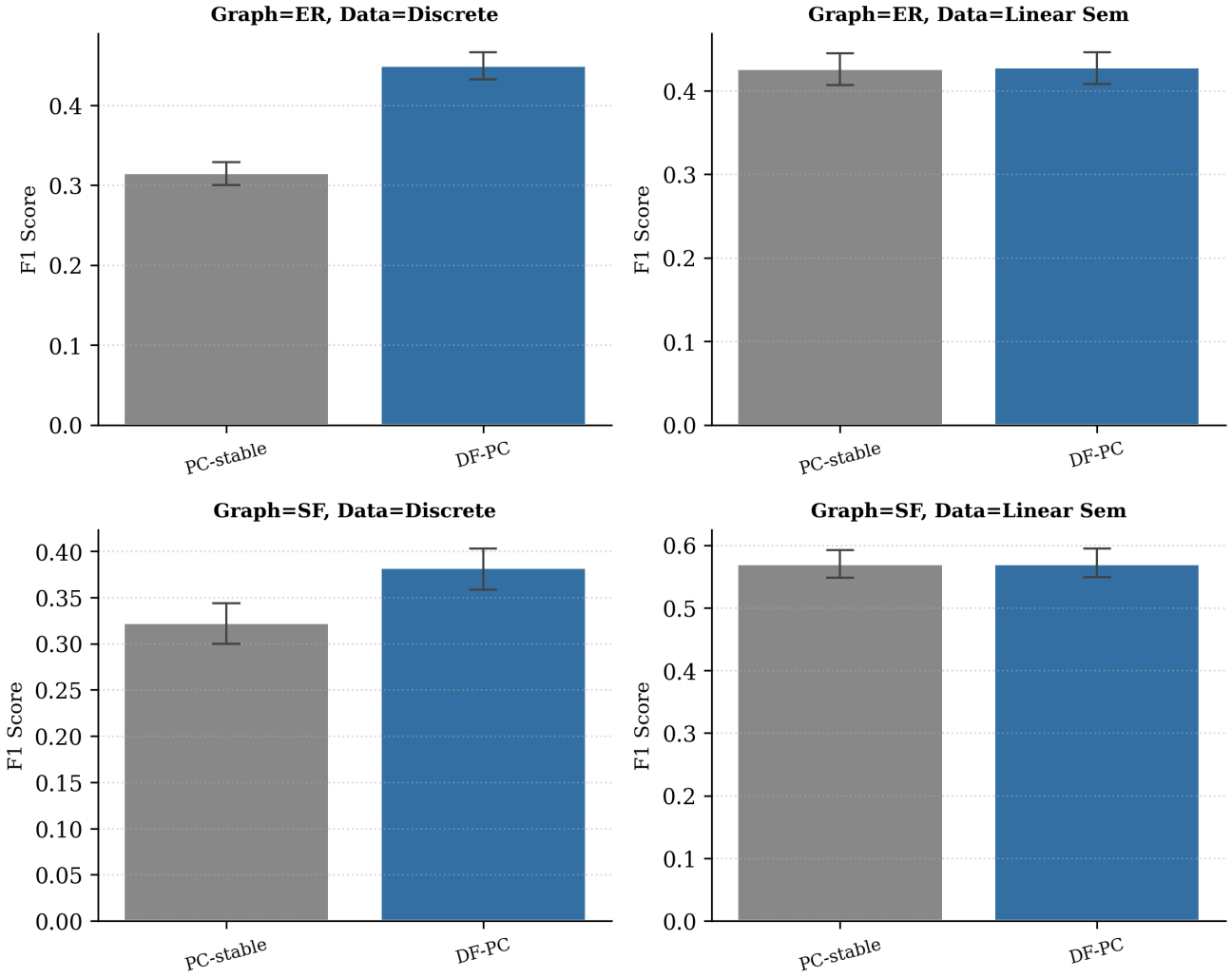

Figure 8: **Performance in High-Density Settings (F1).** F1 score on graphs with 30 nodes, 5000 samples, and an average degree of 6. Error bars represent 95% CI across 30 independent trials.

High-Density Performance Analysis Under high graph density ($D = 6$), DF-PC exhibits a significant advantage in query complexity (Figure 9). In linear SEM datasets, DF-PC performed 14,891.90 CITs on average, bypassing 9,269.33 tests, whereas PC-stable performed 16,568.23 CITs, bypassing only 7,546.70. Moreover, in discrete datasets, DF-PC secures a substantial F1-score improvement (0.416 vs. 0.319) along with a lower SHD (62.617 vs. 66.767), as shown in Figure 8. This indicates that the logical pruning mechanism scales well and maintains robustness even when the causal graph is dense and highly connected.

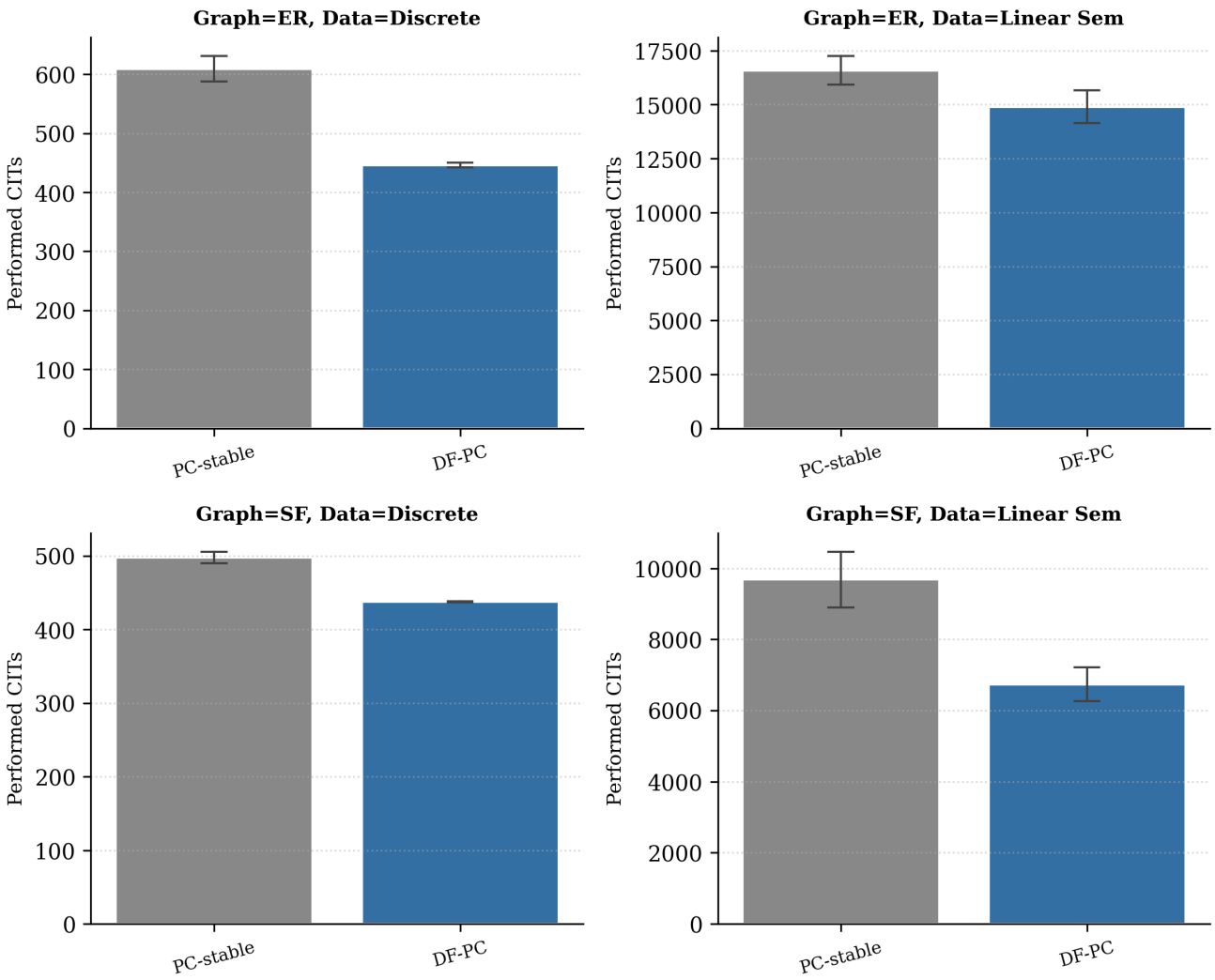

Figure 9: **CIT Counts in High-Density Settings.** Total performed CITs for 30 nodes and 5000 samples with an average degree of 6. Error bars denote 95% CI based on 30 trials.

C.2.9 Runtime Scaling Results

The execution time scaling across node counts and graph density is plotted in Figure 10 and Figure 11.

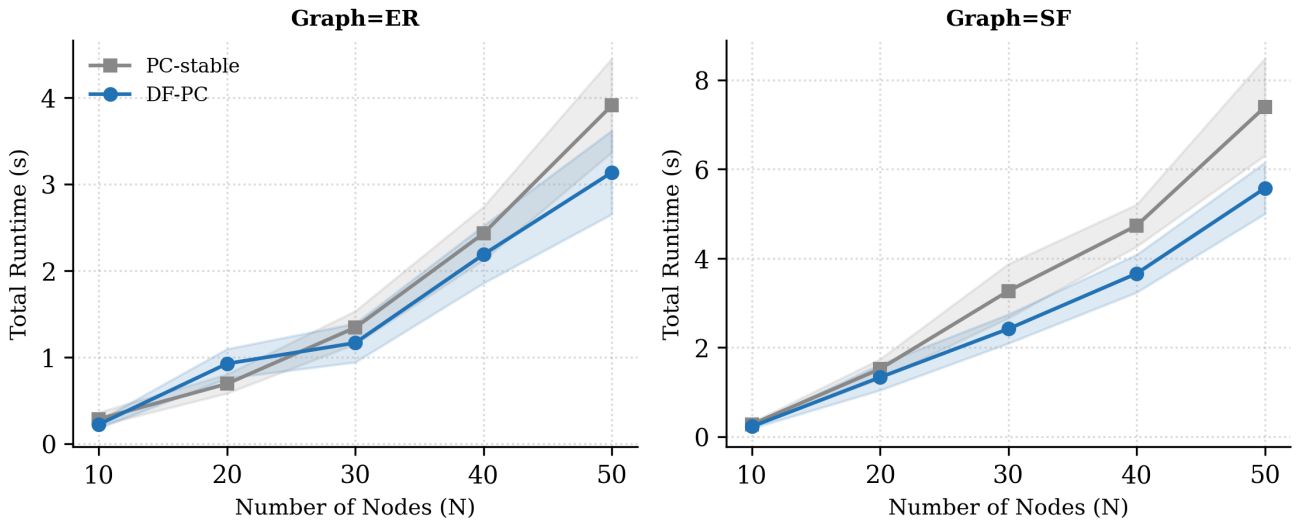

Figure 10: **Runtime Scaling across Node Counts.** Total execution time in seconds as the number of nodes N scales from 20 to 50, with density parameter $D = 3$ and 5000 samples across Linear SEM data. Here D is the ER target average degree and the SF attachment parameter. Markers indicate the mean, and error bars represent the 95% CI over 30 trials.

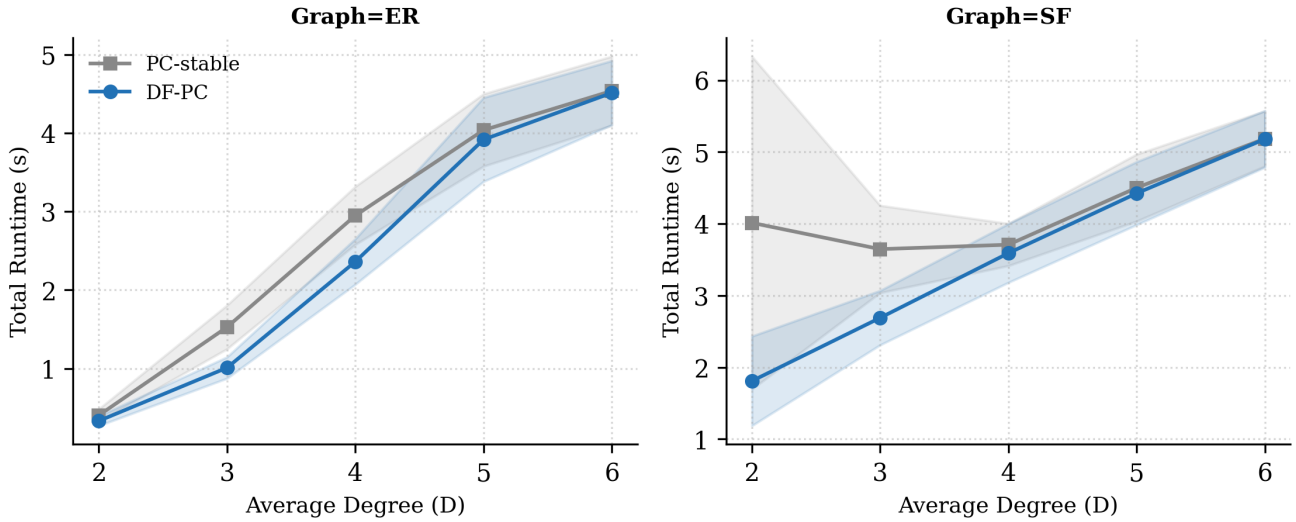

Figure 11: **Runtime Scaling across Graph Density.** Execution time as the density parameter D scales from 2 to 6 for a 30-node graph with 5000 samples in Linear SEM data. The horizontal-axis parameter is the ER target average degree and the SF attachment parameter, not the realized SF average degree. Error bars indicate 95% CI across 30 independent trials.

Runtime Efficiency and Scalability As shown in Figures 10 and 11, DF-PC achieves lower execution times compared to PC-stable in most tested settings. Under the node scaling scenario (Setup A), DF-PC reduces the total runtime to 2.082 seconds on average (a 19.4% speedup) while maintaining a nearly identical F1-score (0.726 vs. 0.729). Under the density scaling scenario (Setup B), DF-PC scales to 2.982 seconds on average (a 13.6% speedup) while PC-stable requires 3.451 seconds.

C.2.10 Oracle CIT Skip Analysis Results

To evaluate the theoretical limit of DF-PC's deduction rules without the interference of statistical errors, we analyzed the bypass rates across conditioning set sizes ($|\mathbf{Z}|$). The skip analysis results across node scaling and density scaling are plotted in Figure 12 and Figure 13.

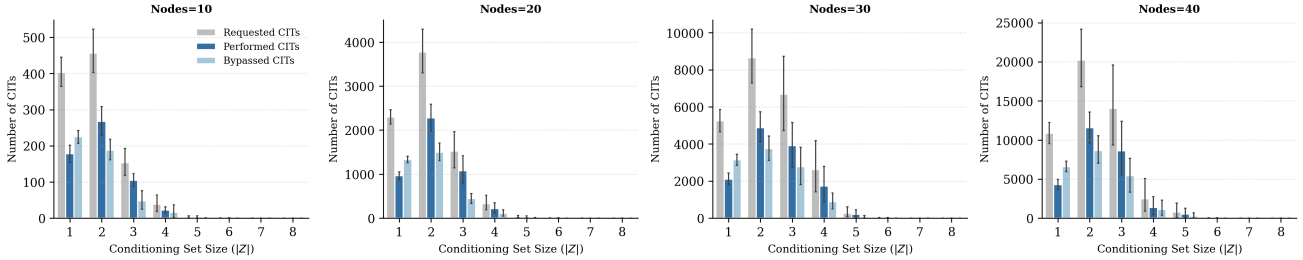

Figure 12: **CIT Skip Analysis across Node Counts.** Decomposition of CIT requests categorized by conditioning set size $|Z|$ for ER graphs with an average degree of 3 using an oracle CIT. *Requested CITs* denotes total tests requested by the algorithm, *Performed CITs* denotes tests actually run by DF-PC, and *Bypassed CITs* denotes CI requests handled without a new statistical computation (this count includes both cache reuse and deduction). Error bars indicate 95% CI based on 30 seeds.

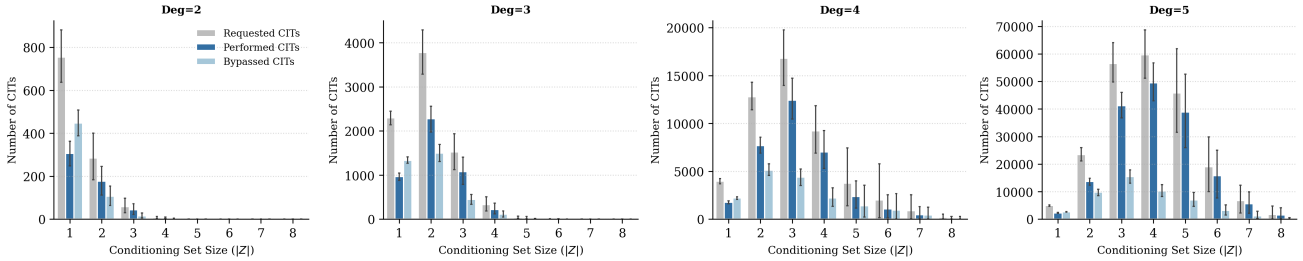

Figure 13: **CIT Skip Analysis across Graph Density.** Categorized CIT counts as graph density increases from average degree 2 to 5 for an ER graph with 20 nodes using an oracle CIT. Error bars indicate 95% CI across 30 trials.

Savings across Conditioning Set Sizes Under the node scaling scenario (Setup A with $N = 40, D = 3$, see Figure 12), the oracle tracker reveals that out of 10,867.8 requested queries at level $|\mathbf{Z}| = 1$, DF-PC bypasses 6,565.93 tests. At level $|\mathbf{Z}| = 2$, out of 20,228.77 requested queries, DF-PC bypasses 8,668.03 tests. Under the density scaling scenario (Setup B with $N = 20, D = 5$, see Figure 13), DF-PC bypasses 2,664.97 tests at level $|\mathbf{Z}| = 1$ (out of 4,914.73 requested), 9,781.27 tests at level $|\mathbf{Z}| = 2$ (out of 23,403.3 requested), and 15,406.03 tests at level $|\mathbf{Z}| = 3$ (out of 56,552.4 requested). These findings demonstrate that the pattern of bypassed tests shifts noticeably depending on the size and density of graph.

C.2.11 Robustness against Low-Order CIT Errors Results

We evaluate the sensitivity of DF-PC's logical propagation by injecting artificial errors into low-order oracle CIT results, as illustrated in Figure 14.

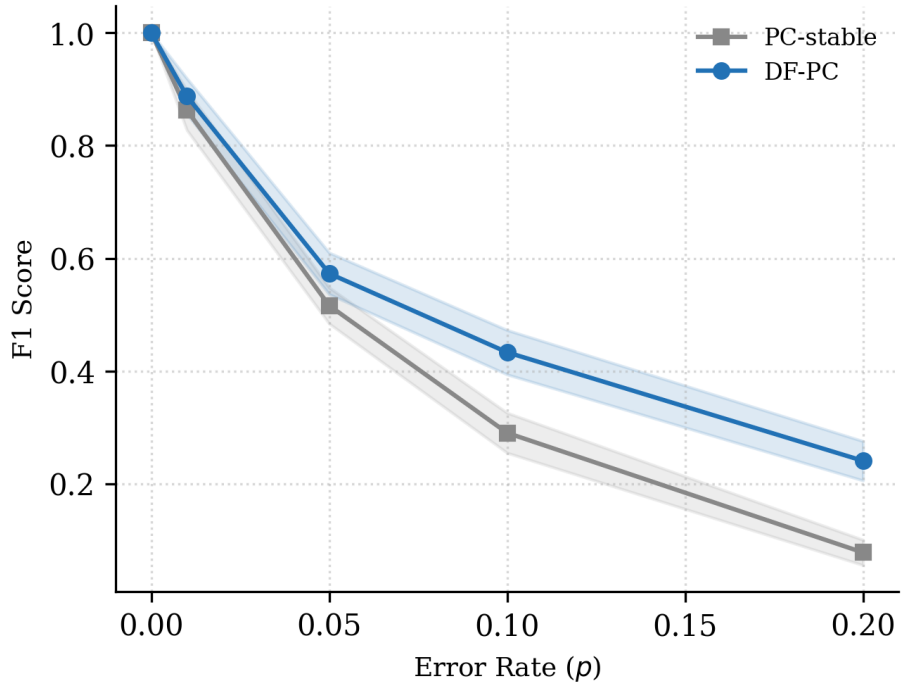

Figure 14: **Robustness against Low-Order CIT Errors.** F1 score when random errors with rate p are injected into low-order oracle CITs ($|Z| \leq 1$) for a graph with 20 nodes and an average degree of 3 in an ER topology. Error bars indicate 95% CI derived from 30 independent trials.

Analysis of Logical Stability Figure 14 highlights a key trade-off of logical deduction propagation. At low error rates ($p \leq 0.05$), DF-PC exhibits superior F1-scores compared to PC-stable. For example, at $p = 0.01$, DF-PC yields an F1-score of 0.888 (vs. 0.863) and a lower SHD of 6.933 (vs. 8.133). Under high error rates ($p \geq 0.10$), DF-PC has higher SHD (32.433 vs. 29.500 at $p = 0.10$). PC-stable produces a sparser network under high noise, leading to slightly lower SHD but very low F1-scores. These results characterize sensitivity to low-order premise errors, with higher-order oracle CIT answers remaining exact. The benefits are metric-dependent: at $p = 0.05$, DF-PC has higher F1 but also higher SHD than PC-stable.