SCALABLE CAUSAL DISCOVERY VIA VERTICAL PARTITIONING OF DATA FEATURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Causal discovery (CD) methods often incur exponential complexity in the number of variables, limiting their scalability. To address this, recent research has proposed to leverage divide-and-conquer strategies to boost the scalability of a wide variety of existing CD algorithms. However, these strategies require significant overhead to either *divide* the original problem into subproblems or aggregate the *conquered* local solutions into a global solution. Furthermore, failing to assert locally statistical properties such as *correctness-and-completeness* often impairs the performance of many algorithms. In this paper, we present a novel divide-and-conquer framework that provably performs both problem decomposition and solution aggregation in quadratic time complexity. Furthermore, our framework theoretically guarantees that the local solutions are consistent with the global solution. Finally, extensive evaluation on a wide variety of benchmark datasets show that our framework consistently and significantly improves the scalability and performance of various CD algorithms. Our implementation is accessible at anonymous.4open.science/r/DEFT.

1 INTRODUCTION

Causal discovery (CD) is the problem of uncovering the causal structure between observed variables from data (Spirtes & Glymour, 1991; Pearl, 2009). Traditionally, the finding of such a causal structure requires an exponential time complexity with respect to the number of variables observed (Spirtes & Glymour, 1991), which makes causal discovery an intractable problem. As a result, many subsequent studies have been dedicated to reduce the complexity of this procedure, either by incorporating heuristics (Chickering, 2002; Hauser & Bühlmann, 2012), or adopting assumptions about the data generating process (Zheng et al., 2018; Ng et al., 2020; Bello et al., 2022).

To further improve the scalability of existing CD methods, a recent divide-and-conquer paradigm has been developed (Cai et al., 2013; Zhang et al., 2022). In essence, higher computational efficiency can be achieved by formulating and solving smaller local problems, then aggregating their solutions. These frameworks are generally expressed as

$$G = \mathcal{A}(\mathcal{D}) \mapsto \begin{cases} \{\mathcal{D}_i\}_{i=1}^{K} = \text{Decompose}\left(\mathcal{D}\right), \\ G = \text{Aggregate}\left(\{\mathcal{A}(\mathcal{D}_i)\}_{i=1}^{K}\right), \end{cases} \tag{1}$$

where G is the output graph, $\mathcal{D}$ is the input data, $\{\mathcal{D}_i\}$ are sub-datasets from $\mathcal{D} = \bigcup_{i=1}^{K} \mathcal{D}_i$, $\mathcal{A}$ is any CD algorithm, $\text{Decompose}(\cdot)$ and $\text{Aggregate}(\cdot)$ denote the data partitioning and solution merging functions, respectively. We formally name the partitioning of the original data into sub-data, i.e., $\mathcal{D} \mapsto \{\mathcal{D}_i\}$, the problem decomposition process. Likewise, the merging of local solutions to achieve the global solution, i.e., $\{\mathcal{A}(\mathcal{D}_i)\} \mapsto G$, is called the aggregation process. Depending on the design of the decomposition and aggregation processes, divide-and-conquer frameworks can be characterized as constraint-based or score-based [1].

Existing Works. Constraint-based approach focuses on strict statistical independence among subproblems. These methods recursively find the cut set to divide the original problem into subproblems (Xie & Geng, 2008). However, it involves solving an additional optimization problem to find a cut set

[1]This is an informal taxonomy for convenience: constraint-based methods use statistical independence tests for problem decomposition whereas score-based methods optimize a score function for the same task.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

at every recursive step (Zhang et al., 2022). Furthermore, for aggregation, extra computational complexity is required to resolve local conflicts (Cai et al., 2013). To remedy these weaknesses, Zhang et al. (2024) introduced an undirected superstructure to guide partitioning, but constructing this superstructure remains computationally expensive. A subsequent work (Shah et al., 2025) assumes this superstructure is known in advance. On the other hand, score-based approach reduces the computational complexity by relaxing exact statistical conditions with estimates (Gu & Zhou, 2020; Huang & Zhou, 2022). For this, they adopt additional assumptions such as the Structural Equations Model (Pearl, 2009). These methods excel at computation speed at the cost of theoretical correctness: they often produce spurious conclusions. In this work, we focus on causal discovery from observational data alone, thus excluding works that rely on prior knowledge as auxiliary information (Wang et al., 2025). Further details regarding related works are provided in Section 5.

High-Level Solution. In this paper, we propose a constraint-based framework. Overcoming limitations of prior methods (Xie & Geng, 2008; Cai et al., 2013; Zhang et al., 2022), we avoid the recursive computational complexity by leveraging a global causal structure whose construction scales quadratically in the number of variables. Intuitively, this structure allows us to allocate groups of variables that are statistically (conditionally) independent without brute-force. As a result, we can directly formulate subproblems and identify their positions within the global solution. Furthermore, we prove that they satisfy *correctness-and-completeness* (Zhang et al., 2022) without additional assumptions (Gu & Zhou, 2020; Huang & Zhou, 2022). This theoretically guarantees the consistency of our local solutions with the global solution. Finally, by proving that our edge-wise aggregated graph is always acyclic, we sidestep a costly procedure of orienting edges and resolving conflicts during aggregation (Xie & Geng, 2008; Cai et al., 2013).

Contributions. Key contributions of this paper are:

1. We construct an additional causal structure that overlays the observational one, enabling identification of statistically independent variable groups without examining individual variables. For this, we introduce groups of variables (Definition 2) and define their statistical relations (Definition 4).

2. We discover that the positions of causally-related groups on the topological order can be established via examining their characteristics (Theorem 1). This allows us to devise a computationally inexpensive identifiable algorithm (Algorithm 1) to build the global causal structure. Upon this structure, we formulate local problems whose *correctness-and-completeness* is assured (Theorem 2).

3. Theoretically, we show that our framework requires a quadratic computational complexity w.r.t. the number of observed variables to perform both decomposition and aggregation processes (Theorem 4). Empirically, we record stark improvements over existing state-of-the-art frameworks in all metrics.

2 PRELIMINARIES

Graphical Models. A graphical model is defined as $G = (\boldsymbol{X}, \boldsymbol{E})$ where $\boldsymbol{X} := \{X_1, X_2, \ldots, X_d\}$ is the set of vertices (a.k.a. nodes, variables, or features), and $\boldsymbol{E}$ is the set of directed edges. Any edge $X_i \to X_j \in \boldsymbol{E}$ forms a causal relationship between X_i and X_j which we say X_i is a parent of X_j. Let $\mathrm{Pa}(X_j)$ denote the parent set of X_j, the joint distribution P entailed by G over $\boldsymbol{X}$ can be factorized accordingly (Pearl, 2009): $P(\boldsymbol{X}) = \prod_{k=1}^{d} P(X_k \mid \mathrm{Pa(X_k)})$. If there exists a causal chain $X_i \to \cdots \to X_k$, then X_i is said to be an ancestor of X_k, and conversely, X_k is a descendant of X_i. In causal discovery, we are interested in a particular type of graphical model: directed acyclic graphs (DAGs). That means, no node X_k is an ancestor of itself.

Statistical Relations. X_i is said to be (statistically) independent of X_j given the set $\boldsymbol{Z} \subset \boldsymbol{X} \setminus \{X_i, X_j\}$, denoted as $X_i \perp\!\!\!\perp_G X_j \mid \boldsymbol{Z}$, if the $\boldsymbol{Z}$ d-separates X_i and X_j on G (Pearl et al., 2016). Otherwise, X_i and X_j are said to be dependent given $\boldsymbol{Z}$, denoted as $X_i \not\perp\!\!\!\perp_G X_j \mid \boldsymbol{Z}$. However, since G is unknown, and that only the data $\mathcal{D} \sim P(\boldsymbol{X})$ is observed, we need to determine $X_i \perp\!\!\!\perp_G X_j \mid \boldsymbol{Z}$ from the data $\mathcal{D}$. For this purpose, we need the following assumptions:

Assumption 1 (Faithfulness (Pearl, 2009))**.** *The distribution P is said to be faithful to the graph G iff $X_i \perp\!\!\!\perp_G X_j \mid \boldsymbol{Z} \Leftrightarrow X_i \perp\!\!\!\perp_P X_j \mid \boldsymbol{Z}$, with $X_i \perp\!\!\!\perp_P X_j \mid \boldsymbol{Z}$ being empirically measured on $\mathcal{D} \sim P$.*

Assumption 2 (Causal Sufficiency (Pearl et al., 2016))**.** *There are no unobserved common causes with respect to any pair of observed variables.*

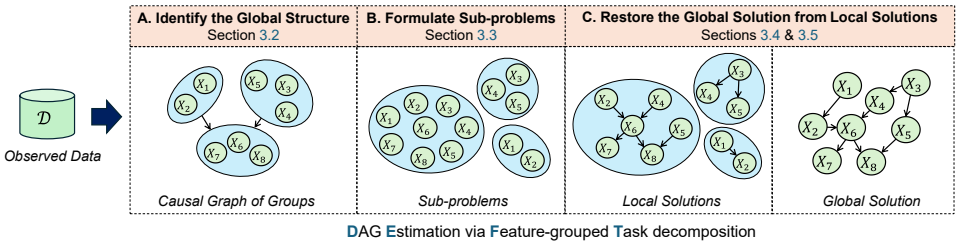


Figure 1: **Framework Overview. A.** Observed data is used to identify the global structure over the variables. **B.** The global structure facilitates problem decomposition. **C.** By solving and merging local solutions, we yield the global solution.

In practice, Assumption 1 is not guaranteed in finite-data regimes. Consequently, conditional independence tests (a.k.a., CIT) often return false conclusions: $X_i \perp\!\!\!\perp_P X_j \mid \boldsymbol{Z} \not\equiv X_i \perp\!\!\!\perp_G X_j \mid \boldsymbol{Z}$. Following existing practice (Kocaoglu, 2023; Lee et al., 2025), we verify our theoretical results by employing oracle CITs to artificially guarantee Assumption 1 ($X_i \perp\!\!\!\perp_P X_j \mid \boldsymbol{Z} \equiv X_i \perp\!\!\!\perp_G X_j \mid \boldsymbol{Z}$). This oracle is used for all baselines for fair comparison. However, oracle CITs are turned off when solving local problems. We evaluate our proposal's performance without oracle CITs in Appendix B.2.

In our work, Assumption 1 allows us to define statistical dependencies between groups (Definition 4). Assumption 2 assures that found dependencies correspond to the true causal graph and are not spuriously concluded (Spirtes et al., 2000). In Section 3.3, subproblems are formulated so that each subproblem is proven to satisfy the *correctness-and-completeness* property (Theorem 2).

3 DAG Estimation via Feature-grouped Task Decomposition

This section presents the workflow of **DEFT** (Figure 1). To commence with, we provide an intuitive overview of the design of our proposed framework, DAG Estimation via Feature-grouped Task decomposition (or **DEFT**), in Section 3.1. Technical details are provided from Sections 3.2 to 3.5.

3.1 An Overview

A. Observed Data to Global Structure. Our proposed framework is built on an insight that if two or more variables share the same set of dependent variables, they can be simultaneously represented by a group. Accordingly, we can easily extend the notion of statistical independence between variables to statistical independence between groups which allows us to provably identify the causal graph of groups (Section 3.2). Unlike the global structure in (Zhang et al., 2024), ours is a fully identifiable causal graph and can be constructed efficiently using an innate topological order.

B. Global Structure to Local Problems. Having identified the causal graph of groups, in Section 3.3, we define a subproblem as a structure learning problem on every set of child-parents groups within the global structure. Such a formulation of a subproblem is proven to be both theoretically and empirically beneficial for a wide range of causal learning algorithms (see Section 4).

C. Local Solutions to Global Solution. As we have constructed a causal graph for each subproblem using any designated structure learning algorithm (Section 3.4), we need to aggregate these component graphs to restore the global causal graph. Fortunately, in Section 3.5, we show that this aggregation can be achieved by an edge-wise merging operator, without the need to directly enforce the global acyclicity constraint.

D. Computational Complexity and Identifiability. We provably show in Section 3.6 that **DEFT**'s decomposition and aggregation procedures scale quadratically with the number of observed variables which is much more preferable over existing frameworks. Regarding identifiability, **DEFT** has the same identifiability as the designated structure learning algorithm it uses to solve subproblems.

3.2 Groups and Causal Graph of Groups

Our decomposability relies on constructing the causal graph of groups. This section details the construction from observational data to the resulting group-level causal structure (Figure 2). Firstly,

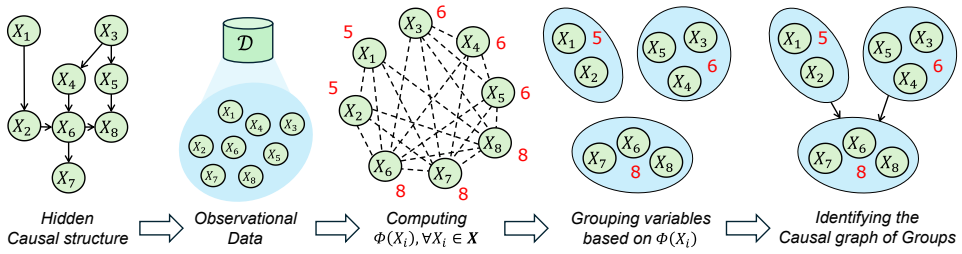


Figure 2: **Identification of the Causal Graph of Groups.** From observed data, we compute the characteristic number (Definition 1) for each variable X_i. These characteristic numbers are then used to define groups. These groups, along with their characteristic numbers, form a topological order which helps construct the causal graph of groups.

we formally define groups and the causal graph of groups, then establish the connection between group-level and variable-level causal relations (Definition 4). Lastly, we show that an intrinsic topological order over groups exists (Theorem 1) that significantly reduces the computational complexity required to construct the global structure.

We start by introducing the notion of characteristic numbers:

Definition 1 (Characteristic Number)**.** Let $\Phi(X) := \{Y \mid Y \in \boldsymbol{X}, Y \not\perp\!\!\!\perp X\}$ be the set of variables that are dependent on X (including X). $|\Phi(X)|$ is called the characteristic number of variable X.

It takes $\mathcal{O}(d^2T)$ to compute $\Phi(\boldsymbol{X}) := \{\Phi(X)\}, \forall X \in \boldsymbol{X}$, where T is the time complexity required for an unconditional independence test between two random variables. Intuitively, the characteristic number of each variable reveals its relative position in the topological order with respect to other variables. From $\Phi(\boldsymbol{X})$ and the characteristic numbers, we define a group as follows:

Definition 2 (Group of Variables)**.** Variables X and Y are said to belong to the same group if $\Phi(X) \equiv \Phi(Y)$, and $\tau = |\Phi(X)| \in \mathbb{N}$ is called the characteristic number of the group containing X.

A simple $\mathcal{O}(d^2)$ procedure is sufficient to partition all nodes into groups according to Definition 2. Furthermore, our variable grouping is deterministic with respect to $\Phi(\boldsymbol{X})$. Therefore, given a causal graph, its grouping is uniquely determined. We now define the causal graph of groups:

Definition 3 (Causal Graph of Groups)**.** $G_{\mathrm{R}} = \left(\{R_k\}_{k=1}^K, \boldsymbol{E}_{\mathrm{R}}\right)$ is called the causal graph of groups if $\forall i, j \in \{1, 2, \ldots, K\} : \left(\exists X \in R_i \wedge \exists Y \in R_j \wedge X \to Y\right) \Leftrightarrow R_i \to R_j \in \boldsymbol{E}_{\mathrm{R}}$. If $R_i \to R_j \in \boldsymbol{E}_{\mathrm{R}}$, we say R_i is a parent of R_j.

Intuitively, if a variable $X \in R_i$ is a parent of another variable $Y \in R_j$ $(j \neq i)$ then R_i is a parent of R_j in the causal graph of groups. In order to construct G_{R}, it is necessary to define and to measure (conditional) independencies among these groups. For this purpose, we define:

Definition 4 (Group-level Relations)**.** Let R_1 and R_2 be two different groups, then R_1 and R_2 are said to be:
(i) dependent iff $\exists X_1 \in R_1, \exists X_2 \in R_2 : X_1 \not\perp\!\!\!\perp X_2$. We write $R_1 \not\perp\!\!\!\perp R_2$.
(ii) independent iff $\forall X_1 \in R_1, \forall X_2 \in R_2 : X_1 \perp\!\!\!\perp X_2$. We write $R_1 \perp\!\!\!\perp R_2$.
(iii) independent given R_3 iff $\forall X_1 \in R_1, \forall X_2 \in R_2 : X_1 \perp\!\!\!\perp X_2 \mid R_3$. We write $R_1 \perp\!\!\!\perp R_2 \mid R_3$.

Now that the statistical relationships between groups are defined, we can build the causal graph of groups using any causal discovery method, such as the Peter-Clark algorithm (Spirtes & Glymour, 1991). However, naively applying such a method incurs a computational complexity that scales exponentially with the number of groups. Fortunately, Theorem 1 offers an insight that the increasing order of the characteristic number of groups $\{\tau_k\}_{k=1}^K$ forms a topological order over their corresponding groups $\{R_k\}_{k=1}^K$:

Theorem 1 (Topological Order of Groups)**.** *If $R_i \not\perp\!\!\!\perp R_j$ and $\tau_i < \tau_j$, then R_i is a non-descendant group of R_j.*

We defer the proof of Theorem 1 to Appendix A.1. Since $\Phi(\boldsymbol{X})$ is precomputed, uncovering the topological order over K groups requires no additional independence tests. Such a topological order is used to construct G_{R} within $\mathcal{O}(d^2)$ (see Algorithm 1, Theorem 6) as follows:

Table 1: **Comparison of computational complexity between baselines.** σ and k_{par} are parameters used by CAPA and CPA, respectively. K is the number of groups defined in Definition 2.

		SADA (2013)	CAPA (2022)	CPA (2024)	HCD (2025)	**DEFT** (ours)
Computational complexity	Decomposition	$\mathcal{O}(d^2 2^{d-2})$	$\mathcal{O}(d^{2+\sigma})$	$\mathcal{O}(d^{2+k_{\text{par}}})$	$\mathcal{O}(d^3)$	$\mathcal{O}(d^2)$
	Aggregation	$\mathcal{O}(2^d)$	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$

Causal graph of groups. Let $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_K\}$ be a topological order over the groups $\{R_1, R_2, \ldots, R_K\}$ following Theorem 1. Then, for any group R_{π_u} $(1 \leq u \leq K)$, we need only to find its parent groups within the set of preceding groups $\{R_{\pi_1}, \ldots, R_{\pi_{u-1}}\}$. Intuitively, to examine whether an edge $R_{\pi_p} \to R_{\pi_u}$ $(1 \leq p < u)$ is spurious or not, we perform a CIT between R_{π_p} and R_{π_u}, conditioning on $[\text{Pa}(R_{\pi_p}) \cup \text{Ch}(R_{\pi_p}) \cup \text{Sp}(R_{\pi_p})] \cap \{R_{\pi_i}\}_{i=1}^{u-1}$, where $\text{Ch}(\cdot)$ and $\text{Sp}(\cdot)$ denote the set of child and spouse groups, respectively. As we condition on this set, the causal effect of R_{π_p} to R_{π_u} can be exactly examined by the CIT. If not independent, it follows that $R_{\pi_p} \to R_{\pi_u}$. This logic is implemented in Algorithm 1, along with the identifiability of G_{R} (Appendix A.4).

Now that G_{R} is constructed, the next step is to acquire the causal graph within each group. However, formulating a subproblem as a group is problematic since it inherently gives rise to situations where assumption 2 is violated. Indeed, suppose that $R_i \to R_j$ is known, it is not possible for many algorithms to build a causal graph for R_j given the data consisting of nodes in R_j alone, since some nodes can be confounded by R_i.

3.3 Problem Decomposition

Here, we present the method to divide the original dataset $\mathcal{D}$ into sub-datasets $\{\mathcal{D}_k\}_{k=1}^K$ such that each $\mathcal{D}_k$ can be solved effectively.

Notations: For convenience, we introduce the notation for vertical data segmentation as $\mathcal{D}[\,\cdot\,]$, where $\mathcal{D}[R_i]$ denotes a vertical subset of the original data $\mathcal{D}$ with columns being variables in R_i. Furthermore, we use $\hat{G}_i$ for the causal graph built from variables of R_i, and $\hat{G}_{ij}$ for the causal graph connecting variables between R_i and R_j, and not within R_i or R_j. In other words, $\{\hat{G}_i\}$ concerns about the causal graph inside each group whereas $\{\hat{G}_{ij}\}$ gives information of the causal relations between groups. Lastly, we overload the notation $\text{Pa}(R_i)$ to denote the set of parent groups of R_i, based on the causal relations defined by G_{R}.

As we have discussed in Section 3.2, the construction of $\hat{G}_i$ requires not only $\mathcal{D}[R_i]$ but rather $\mathcal{D}[R_i \cup \text{Pa}(R_i)]$. Figure 3 captures this idea perfectly. Our Theorem 2 asserts that if two variables are conditionally independent given the global context, they are also conditionally independent given the local context. That is equivalent to the *correctness and completeness* property (Shen et al., 2025; Zhang et al., 2024) of local solutions:

Remark 1. $\mathcal{D}[R_i \cup \text{Pa}(R_i)]$ is used to perform causal discovery for $\bigcup_{j \in \text{Pa}(R_i)} \hat{G}_{ij} \cup \hat{G}_i$.

Theorem 2. $\forall (X_i, X_j) \in \bigcup_{j \in \text{Pa}(R_i)} \hat{G}_{ij} \cup \hat{G}_i, \exists \mathcal{Z} \subset \boldsymbol{X}$: $X_i \perp\!\!\!\perp_G X_j | \mathcal{Z} \Leftrightarrow \exists \mathcal{Z}' \subset R_i \cup \text{Pa}(R_i) : X_j \perp\!\!\!\perp_G X_i | \mathcal{Z}'$.

Proofs for Theorem 2 is deferred to Appendices A.2. In essence, Theorem 2 indicates that every independence found in G can also be found in corresponding groups. This theoretically guaranteed that no relations are spuriously concluded. In summary, a subproblem corresponds to a vertical subset of the original dataset whose columns are variables of a group and its causal parent groups, as dictated by G_{R}.

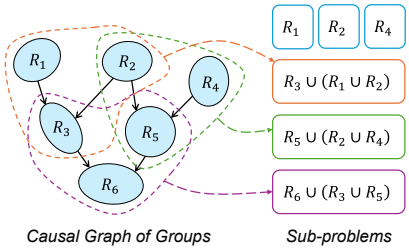


Figure 3: **Problem Decomposition.** $R_1, R_2,$ and R_4 are each considered a subproblem as they have no parent groups.

Remark 2. The number of subproblems equals the number of groups, which is K.

3.4 Local Solution Identification

With subproblems formulated, we can apply any designated algorithm (subroutine) $\mathcal{A}$ to solve these subproblems. Ideally, when $\mathcal{A}$ is an identifiability-guaranteed, precision-oriented algorithm that struggles with scalability, we can apply $\mathcal{A}$ on compact subsets of variables to achieve high precision at a significantly reduced time complexity. We examine a variety of subroutines $\mathcal{A}$ in Section 4.

As shown by Remark 1 and Theorem 2, $\bigcup_{j\in \mathrm{Pa}(R_i)} \hat{G}_{ij} \cup \hat{G}_i = \mathcal{A}\left(\mathcal{D}[R_i \cup \mathrm{Pa}(R_i)]\right)$ is theoretically and practically feasible for algorithm $\mathcal{A}$. Furthermore, $\mathcal{A}$ is now applied on a potentially much smaller feature space compared to the full feature space of the original data, which implies that computational complexity can be reduced significantly as evidently shown by empirical studies (Section 4) and theoretical analysis (Theorem 4). Notice that as per Definition 3, every edge in $\hat{G}_{ij}, \forall i, j$, is automatically oriented following the topological order between R_i and R_j.

Remark 3. While the identifiability of G_{R} is guaranteed by Theorem 5, that of $\hat{G}_k$ depends on algorithm $\mathcal{A}$ and its assumptions. Thus, the identifiability of the complete graph G depends on $\mathcal{A}$.

3.5 Global Solution Aggregation

Now, we have a set of component graphs $\{\hat{G}_k\}_{k=1}^K$ at hand. The task left is to assemble these graphs to obtain the complete causal graph. Fortunately, we have already established the causal relations between groups via $\{\hat{G}_{ij}\}$ and they do not conflict with causal relations within each group $\{\hat{G}_i\}$. Therefore, it is natural to aggregate these graphs edge-wise. Furthermore, the aggregation of component graphs is a valid directed acyclic graph:

Theorem 3 (Guaranteed Global Acyclicity)**.** *Let G denote the edge-wise aggregated graph of $\{\hat{G}_k\}_{k=1}^K$. If $\forall i \in \{1, \ldots, K\} : \hat{G}_i$ is a DAG, then G is also a DAG.*

The proof for Theorem 3 is deferred to Appendix A.3. One of the main reasons why most frameworks struggle with end-to-end parallelism is the requirement of acyclicity (Cai et al., 2013; Zhang et al., 2022). Indeed, one needs the entire view of the graph to enforce acyclicity. However, as per Theorem 3, as long as each individual $\hat{G}_i$ is acyclic, the aggregated G is guaranteed to be acyclic. Thus, **DEFT** can run in parallel.

3.6 Time Complexity Analysis

Here, we conduct a theoretical analysis on the computational complexity of **DEFT**.

Theorem 4 (Complexity Analysis)**.** *Let $T_{\mathcal{A}}$ be the worst-case computational complexity required by algorithm $\mathcal{A}$ for a local subproblem, T be the computational complexity required to perform a statistical independence test, and K be the number of groups on G, following Definition 2. Then, the total complexity required by our framework is $\mathcal{O}(d^2T + d^2T + KT_{\mathcal{A}} + d^2)$.*

Proof. Our grouping procedure is dominated by the computation of $\Phi(\boldsymbol{X})$, which is $\mathcal{O}(d^2T)$. The groups are used to build the causal graph of groups, which costs another $\mathcal{O}(d^2T)$. Then, we apply routine $\mathcal{A}$ on each group, hence $\mathcal{O}(KT_{\mathcal{A}})$. Finally, solutions are aggregated in $\mathcal{O}(d^2)$. Therefore, the total computational complexity amounts to $\mathcal{O}(d^2T + d^2T + KT_{\mathcal{A}} + d^2)$. □

An ablation study regarding the values of K is deferred to Appendix B.6. Please note that $K < d$ is not a hyperparameter but the result of applying our grouping strategy (Def. 2) to the input data. For example, K values for datasets Win95pts, Andes, Diabetes are $61, 179$, and 197, respectively. As shown in Table 1, our framework exceeds the most recent state-of-the-art HCD (Shen et al., 2025) in terms of the computational complexity as well as other metrics of interest (Table 3). We also conduct ablation studies on the size of subproblems produced by **DEFT** in Appendix B.7.

4 Empirical Studies

Following the practice in Zhang et al. (2024), we conduct experiments on large-sized real-world graphs provided by the bnlearn standard package (Scutari, 2010), including Win95pts (76 nodes), Andes (223), Diabetes (413), Link (724), and Munin (1041). Synthetic graphs are also used for evaluation: we randomly generate Erdos-Renyi graphs consisting of 400 and 500 nodes, reusing the

Table 2: **Performance boosting for different subroutines**. **Green** emphasizes better.

		Precision ↑	Recall ↑	F1-score ↑	Spurious ↓	SHD ↓	Runtime(s) ↓
(A) $d = 400$; $e = 800$	PC	0.34 ± 0.03	0.36 ± 0.03	0.35 ± 0.02	0.49 ± 0.02	922.3 ± 44.05	6467.21 ± 1276.23
	+DEFT (ours)	0.79 ± 0.02	0.9 ± 0.02	0.84 ± 0.02	0.13 ± 0.02	264.4 ± 30.77	11.9 ± 2.49
	ICALiNGAM	0.55 ± 0.03	0.77 ± 0.02	0.64 ± 0.03	0.36 ± 0.03	590.1 ± 58.6	1162.25 ± 377.81
	+DEFT (ours)	0.93 ± 0.02	0.94 ± 0.01	0.94 ± 0.01	0.04 ± 0.01	77.8 ± 18.62	33.27 ± 2.7
	GOLEM	0.81 ± 0.2	0.8 ± 0.24	0.8 ± 0.22	0.17 ± 0.19	284.5 ± 310.05	552.58 ± 85.13
	+DEFT (ours)	0.93 ± 0.02	0.94 ± 0.01	0.93 ± 0.02	0.07 ± 0.02	108.2 ± 25.23	235.23 ± 6.34
	GraNDAG	0.63 ± 0.21	0.04 ± 0.02	0.08 ± 0.04	0.32 ± 0.24	798.6 ± 35.0	1459.76 ± 122.3
	+DEFT (ours)	0.99 ± 0.01	0.59 ± 0.02	0.74 ± 0.01	0.0 ± 0.0	325.8 ± 15.13	2129.46 ± 219.54
	SCORE	0.02 ± 0.01	0.03 ± 0.01	0.03 ± 0.01	0.64 ± 0.04	1521.7 ± 62.71	1004.07 ± 142.0
	+DEFT (ours)	0.9 ± 0.02	0.78 ± 0.02	0.83 ± 0.02	0.1 ± 0.02	245.3 ± 28.78	492.05 ± 37.62
	Varsort	0.68 ± 0.06	0.96 ± 0.01	0.79 ± 0.05	0.29 ± 0.06	369.9 ± 100.47	303.05 ± 148.88
	+DEFT (ours)	0.93 ± 0.03	0.98 ± 0.01	0.95 ± 0.02	0.05 ± 0.02	62.9 ± 26.37	7.53 ± 0.94
(B) $d = 500$; $e = 1000$	PC	0.3 ± 0.03	0.36 ± 0.02	0.33 ± 0.02	0.54 ± 0.03	1284.1 ± 90.65	16227.8 ± 3220.32
	+DEFT (ours)	0.78 ± 0.02	0.9 ± 0.02	0.84 ± 0.02	0.13 ± 0.01	331.8 ± 35.35	17.72 ± 4.05
	ICALiNGAM	0.56 ± 0.03	0.78 ± 0.02	0.65 ± 0.03	0.36 ± 0.03	727.7 ± 79.27	2173.43 ± 390.1
	+DEFT (ours)	0.93 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.04 ± 0.01	99.7 ± 14.89	45.21 ± 5.34
	GOLEM	0.84 ± 0.17	0.83 ± 0.23	0.83 ± 0.2	0.15 ± 0.16	310.4 ± 342.37	608.85 ± 55.61
	+DEFT (ours)	0.92 ± 0.02	0.94 ± 0.02	0.93 ± 0.02	0.08 ± 0.02	143.0 ± 34.2	297.17 ± 6.02
	GraNDAG	0.55 ± 0.25	0.02 ± 0.01	0.04 ± 0.02	0.36 ± 0.31	1030.5 ± 95.35	1357.16 ± 148.92
	+DEFT (ours)	0.99 ± 0.0	0.59 ± 0.01	0.74 ± 0.01	0.0 ± 0.0	407.8 ± 14.14	2570.49 ± 255.05
	SCORE	0.02 ± 0.0	0.03 ± 0.01	0.03 ± 0.0	0.64 ± 0.02	1876.4 ± 35.62	1426.32 ± 132.09
	+DEFT (ours)	0.89 ± 0.03	0.76 ± 0.02	0.82 ± 0.02	0.1 ± 0.03	322.9 ± 40.61	623.34 ± 59.29
	Varsort	0.65 ± 0.03	0.95 ± 0.01	0.77 ± 0.02	0.32 ± 0.03	514.8 ± 66.24	757.55 ± 406.83
	+DEFT (ours)	0.93 ± 0.02	0.98 ± 0.01	0.95 ± 0.01	0.05 ± 0.01	78.7 ± 17.62	12.33 ± 1.69

Table 3: **Comparison with state-of-the-arts baseline frameworks.** Experiments that do not return results within 24 hours are denoted as "NA". Subroutine is the PC algorithm. nCIT denotes the number of conditional independence tests recorded. **Green** emphasizes best.

		DEFT (ours)	HCD	CPA	CAPA	SADA
Win95pts 76 nodes	↑ Precision	0.83 ± 0.02	0.34 ± 0.04	0.37 ± 0.03	0.51 ± 0.03	0.44 ± 0.05
	↑ Recall	0.92 ± 0.02	0.47 ± 0.04	0.57 ± 0.05	0.75 ± 0.04	0.55 ± 0.06
	↑ F1-score	0.87 ± 0.01	0.40 ± 0.04	0.44 ± 0.04	0.60 ± 0.04	0.49 ± 0.05
	↓ Spurious	0.05 ± 0.01	0.28 ± 0.03	0.49 ± 0.03	0.47 ± 0.03	0.36 ± 0.03
	↓ SHD	29.0 ± 2.49	129.4 ± 9.4	136.6 ± 9.82	106.89 ± 11.25	101.6 ± 9.88
	↓ nCIT	3156	23127	27386	63506	$(5.09 \pm 1.48) \times 10^6$
Andes 223 nodes	↑ Precision	0.80 ± 0.02	0.28 ± 0.02	0.16 ± 0.01	0.21 ± 0.01	0.28 ± 0.03
	↑ Recall	0.72 ± 0.02	0.36 ± 0.03	0.34 ± 0.03	0.46 ± 0.02	0.23 ± 0.02
	↑ F1-score	0.76 ± 0.02	0.32 ± 0.02	0.22 ± 0.02	0.29 ± 0.01	0.25 ± 0.02
	↓ Spurious	0.14 ± 0.02	0.52 ± 0.02	0.74 ± 0.01	0.79 ± 0.01	0.57 ± 0.02
	↓ SHD	148.6 ± 10.48	466.8 ± 14.55	766.2 ± 15.63	758.75 ± 4.79	418.62 ± 16.37
	↓ nCIT	34260	1997832	2061087	5893967	$(95.6 \pm 11.1) \times 10^6$
Diabetes 413 nodes	↑ Precision	0.45 ± 0.03	0.29 ± 0.07	0.28 ± 0.03	NA	NA
	↑ Recall	0.60 ± 0.03	0.33 ± 0.05	0.38 ± 0.02	NA	NA
	↑ F1-score	0.51 ± 0.03	0.31 ± 0.06	0.32 ± 0.03	NA	NA
	↓ Spurious	0.36 ± 0.04	0.51 ± 0.13	0.54 ± 0.05	NA	NA
	↓ SHD	614.17 ± 51.55	834.88 ± 167.62	880.67 ± 67.57	NA	NA
	↓ nCIT	136059	21.21×10^6	19.36×10^6	NA	NA
Link 724 nodes	↑ Precision	0.81 ± 0.01	0.23 ± 0.01	0.18 ± 0.01	NA	NA
	↑ Recall	0.91 ± 0.01	0.28 ± 0.01	0.29 ± 0.02	NA	NA
	↑ F1-score	0.86 ± 0.01	0.25 ± 0.01	0.22 ± 0.01	NA	NA
	↓ Spurious	0.05 ± 0.01	0.47 ± 0.01	0.60 ± 0.01	NA	NA
	↓ SHD	335.8 ± 16.85	1625.0 ± 22.06	1945.4 ± 55.89	NA	NA
	↓ nCIT	277397	25.97×10^6	26.41×10^6	NA	NA
Munin 1041 nodes	↑ Precision	0.64 ± 0.01	0.36 ± 0.01	NA	NA	NA
	↑ Recall	0.78 ± 0.01	0.53 ± 0.01	NA	NA	NA
	↑ F1-score	0.70 ± 0.01	0.43 ± 0.01	NA	NA	NA
	↓ Spurious	0.27 ± 0.01	0.38 ± 0.02	NA	NA	NA
	↓ SHD	875.0 ± 24.89	1728.33 ± 16.17	NA	NA	NA
	↓ nCIT	591197	63.53×10^6	NA	NA	NA

practice in Zheng et al. (2018). For other classes of graphs (bipartite, scale-free), see Appendix B.5. Here, we present the results on data that are synthetically generated following the linear Gaussian-noise additive model. For non-Gaussian noise, we defer the results to Appendix B.3. Baselines we are to compare consist of state-of-the-art supporting frameworks: HCD (Shen et al., 2025), CPA (Zhang et al., 2024), CAPA (Zhang et al., 2022), and SADA (Cai et al., 2013). Experiments presented in the main-text only oracle CITs for decomposition and aggregation, not local solving. For finite-data regimes, see Appendix B.2. The metrics of interest are Precision, Recall, F1-score, Spurious rate, Structure Hamming Distance (SHD), and Runtime. Code for subroutines is openly accessible via the Gcastle package (Zhang et al., 2021) and their original papers (Reisach et al., 2021; Rolland et al., 2022). Hyper-parameters settings for subroutines are detailed in Table 4 (Appendix B.1).

4.1 IMPROVEMENT OVER SUBROUTINE $\mathcal{A}$

We first examine the effectiveness of our proposal as performance booster for different subroutines $\mathcal{A}$. The results are shown in Table 2 where the chosen subroutines include PC (Spirtes & Glymour, 1991), ICALiNGAM (Shimizu et al., 2006), GraNDAG (Lachapelle et al., 2019), GOLEM (Ng et al., 2020), Varsort (Reisach et al., 2021), and SCORE (Rolland et al., 2022). In addition to other evaluation metrics, we also compare direct runtime in seconds. For each scenario, we randomly generate 10 graphs using different seeds, and report means and standard deviations. Since some improvements in percentages can go arbitrarily high, we find it more suitable to report them in absolute differences.

In general, all metrics of interest are significantly boosted. Regarding positive scores (higher is better), we record an average improvement in Precision across subroutines by $+0.31$ in scenario (A) and $+0.33$ in scenario (B). Noticeably, SCORE benefits a staggering $+0.88$ in Precision when boosted by our framework, **DEFT**. The figures for the Recall metric are $+0.22$ and $+0.21$, respectively in scenarios (A) and (B). Consequentially, we record a surge in the F1-score metric with an average of $+0.42$ in the first scenario and 0.43 in the other. Besides SCORE, which improves on average by $+0.8$, GraNDAG also receives a sizable $+0.68$ as well. Regarding negative scores (lower is better), our framework consistently reduces spurious rates of all subroutines, which, in turn, substantially benefits the SHD metric. Specifically, an average of 0.32 spurious rate is reduced when subroutines are integrated within **DEFT**. In scenario (A), this results in an average 567.11 lower in SHD whereas in scenario (B), the figure is 726.67. Such reductions are equivalent to 2 up to 8 times lower SHD compared to the standalone version for some subroutines.

Finally, our framework enables causal discovery at scale, such that even the least-scalable PC algorithm becomes the second fastest subroutine. Indeed, among subroutines that benefit from the complexity reduction, is shortened on average by a factor of 543.46 in scenario (A) and 915.79 in scenario (B). Varsort and ICALiNGAM are also significantly hastened by factors of 61.44 and 48.01, respectively. However, in the rare case of GraNDAG, the re-initialization (i.e., of neural networks and solvers) takes extra runtime – scales with the number of subproblems – which results in a slightly higher end-to-end runtime when integrated into our framework.

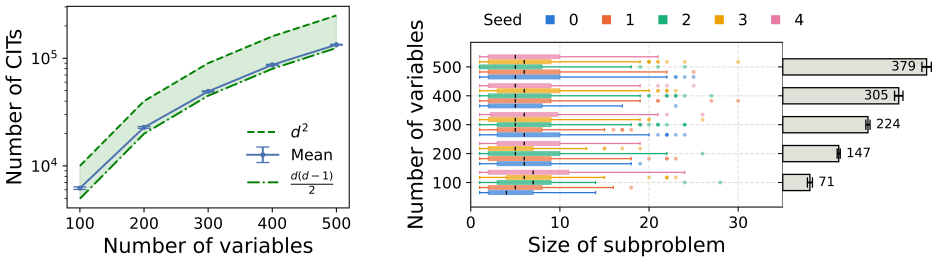
4.2 SUPERIORITY OVER STATE-OF-THE-ART FRAMEWORKS

Here, we present a thorough comparison between our proposal with state-of-the-art divide-and-conquer frameworks. To compare scalability, the number of CI tests (exclusively for task decomposition) required by each baseline and our proposal are used instead of runtime.

Table 3 presents empirical evaluations of **DEFT** and other frameworks on the five real-world, increasingly larger graphs. Overall, it can be seen that our framework consistently outperforms other frameworks by a large margin. Most noticeably, **DEFT** almost doubles the F1-score of the second best baseline framework across experiments. Indeed, the F1-score ratios between our framework and the second best framework are 1.45 (Win95pts), 2.37 (Andes), 1.6 (Diabetes), 3.44 (Link), and 1.63 (Munin). The reduction in terms of SHD is equally impressive as **DEFT** achieves averagely 75.17% lower SHD in the Win95pts dataset, 73.42% in Andes, 28.34% in Diabetes, 81.04% in Link, and 49.37% in the Munin dataset. Finally, empirical results show that our framework requires significantly fewer conditional independence tests (nCIT) than existing state-of-the-art frameworks. While HCD is the most efficient baseline in terms of nCIT, its $\mathcal{O}(d^3)$ complexity inevitably incurs up to $107.46\times$ more nCIT than **DEFT** on the largest dataset (Munin, 1041 variables). This effectively shows the scalability of our framework compared to state-of-the-art baselines.

4.3 EMPIRICAL COMPLEXITY ANALYSIS

Figure 4.(a) shows the records of number of conditional independence tests (CITs) across our synthetic experiments on Erdos-Renyi graphs that are used in Table 2, summarized over 10 independent runs. It can be seen that **DEFT** requires no more than $O(d^2)$ CITs across scenarios, adhering to our theoretical analysis (Theorem 4). Regarding the local complexity, 4.(b) depicts the distribution of subproblems' size with increasingly larger graphs. Interestingly, we consistently observe relatively small subproblem sizes (mean ~ 5, maximum 30) across all experiments regardless of input graph sizes. Consequently, the number of subproblems K grows almost linearly with respect to graph size. More results regarding the complexity analysis of real world graphs are presented in Appendix B.7.



(a) Complexity records (b) Subproblem records

Figure 4: **Analysis on synthetic datasets.** The graphs used in this experiment are the same as reported in Section 4.1. We visualize only 5 random seeds in Figure 4.(b). The number of subproblems K is also depicted in the gray barplot on the right of figure 4.(b).

5 Related Works

Constraint-based approach concerns with strict statistical independence among subproblems, and has been investigated thoroughly. Xie & Geng (2008) pioneered a framework that recursively partitions the variable set into three distinct sets $A, B,$ and C such that A is independent of B conditioned on C. As a result, both decomposition and aggregation procedures incur a heavy overhead. Since there are exponentially many candidates for C, Cai et al. (2013) argued that the most desirable C is the one maximizes $\min\{|A|, |B|\}$. They then applied a traversal strategy to find such a set. A later work, *Causal Partitioning* (**CAPA**, Zhang et al. (2022)), instead maximized $\max\{|A|, |B|\}$. More recently, Zhang et al. (2024) proposed *Causal Partitioning by Edge Cutting of Adjoint Graph* (**CPA**) and formulated a more complicated objective function for the cut set. However, to facilitate the finding of the optimal cut, a heavy procedure is taken to first uncover the skeleton (superstructure) of the true causal graph. A following research (Shah et al., 2025) assumed to have this superstructure in advance. Most recently, Shen et al. (2025) devised the framework *Hierarchical Causal Discovery from large-scale observed variables* (**HCD**) that determines the size of the cut set before optimizing for C. Dong et al. (2025) relied on the linear Structural Equation Model (SEM) assumption to perform problem decomposition and linear programming for their aggregation. In summary, this line of works requires additional computational complexity to (i) solve the optimal cut set problem or construct the superstructure, and (ii) reliably aggregating graphs recursively while resolving local conflicts.

Score-based approach reduces the computational complexity by relaxing exact statistical conditions with estimates. Gu & Zhou (2020) proposed clustering local structures using a SEM-based distance and hierarchical clustering to produce partial directed acyclic graphs (PDAGs). Merging these PDAGs requires an additional score-based optimization, relying on linear SEM assumptions. Later, Huang & Zhou (2022) introduced normalized distances to improve robustness during graph aggregation. However, these works excel at computation speed at the cost of theoretical correctness — global assumptions (e.g., causal sufficiency (Pearl, 2009)) are not guaranteed to hold locally which results in spurious conclusions in graph.

Theoretically, *Correctness-and-completeness* (Zhang et al., 2022) is the most important property for divide-and-conquer methods as it theoretically guarantees that the local solutions are consistent with the hidden causal graph. While some constraint-based methods (Zhang et al., 2024; Shen et al., 2025) can deliver such a result, others either cannot or require further assumptions (Huang & Zhou, 2022).

6 Conclusion

This paper presented a framework, **DEFT**, that effectively performs divide-and-conquer for scalable causal discovery. Our method is grounded on independence tests, and theoretically supported by rigorous proofs. Through extensive empirical evaluation, **DEFT** is highly favorable compared to recent state-of-the-art frameworks with markedly large margins of improvement, both in terms of performance and scalability.

REFERENCES

Kevin Bello, Bryon Aragam, and Pradeep Ravikumar. Dagma: Learning dags via m-matrices and a log-determinant acyclicity characterization. *Advances in Neural Information Processing Systems*, 35:8226–8239, 2022.

Ruichu Cai, Zhenjie Zhang, and Zhifeng Hao. Sada: A general framework to support robust causation discovery. In *International conference on machine learning*, pp. 208–216. PMLR, 2013.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

Shuyu Dong, Michele Sebag, Kento Uemura, Akito Fujii, Shuang Chang, Yusuke Koyanagi, and Koji Maruhashi. Dcilp: A distributed approach for large-scale causal structure learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 16345–16353, 2025.

Jiaying Gu and Qing Zhou. Learning big gaussian bayesian networks: Partition, estimation and fusion. *Journal of Machine Learning Research*, 21(158):1–31, 2020. URL http://jmlr.org/papers/v21/19-318.html.

Alain Hauser and Peter Bühlmann. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *The Journal of Machine Learning Research*, 13(1): 2409–2464, 2012.

Jireh Huang and Qing Zhou. Partitioned hybrid learning of bayesian network structures. *Machine Learning*, 111(5):1695–1738, 2022.

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

Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*, 2019.

Kenneth Lee, Bruno Ribeiro, and Murat Kocaoglu. Constraint-based causal discovery from a collection of conditioning sets. In *The 41st Conference on Uncertainty in Artificial Intelligence*, 2025.

Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the role of sparsity and dag constraints for learning linear dags. *Advances in Neural Information Processing Systems*, 33:17943–17954, 2020.

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

Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

Alexander Reisach, Christof Seiler, and Sebastian Weichwald. Beware of the simulated dag! causal discovery benchmarks may be easy to game. *Advances in Neural Information Processing Systems*, 34:27772–27784, 2021.

Paul Rolland, Volkan Cevher, Matthäus Kleindessner, Chris Russell, Dominik Janzing, Bernhard Schölkopf, and Francesco Locatello. Score matching enables causal discovery of nonlinear additive noise models. In *International Conference on Machine Learning*, pp. 18741–18753. PMLR, 2022.

Marco Scutari. Learning bayesian networks with the bnlearn r package. *Journal of Statistical Software*, 35(3):1–22, 2010. doi: 10.18637/jss.v035.i03. URL https://www.jstatsoft.org/index.php/jss/article/view/v035i03.

Ashka Shah, Adela Frances DePavia, Nathaniel C Hudson, Ian Foster, and Rick Stevens. Causal discovery over high-dimensional structured hypothesis spaces with causal graph partitioning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=FecsgPCOHk.

Rujia Shen, Muhan Li, Chao Zhao, Boran Wang, Yi Guan, Jie Liu, and Jingchi Jiang. Hierarchical causal discovery from large-scale observed variables. *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2626–2639, 2025. doi: 10.1109/TKDE.2025.3539788.

Shohei Shimizu, Patrik O. Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-gaussian acyclic model for causal discovery. *J. Mach. Learn. Res.*, 7:2003–2030, December 2006. ISSN 1532-4435.

Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72, 1991.

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

Xiangyu Wang, Taiyu Ban, Lyuzhou Chen, Derui Lyu, Qinrui Zhu, and Huanhuan Chen. Large-scale hierarchical causal discovery via weak prior knowledge. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Xianchao Xie and Zhi Geng. A recursive method for structural learning of directed acyclic graphs. *Journal of Machine Learning Research*, 9(14):459–483, 2008. URL `http://jmlr.org/papers/v9/xie08a.html`.

Zhang, Shuigeng Zhou, Chuanxu Yan, Jihong Guan, Xin Wang, Ji Zhang, and Jun Huan. Learning causal structures based on divide and conquer. *IEEE transactions on cybernetics*, 52(5):3232–3243, 2022.

Zhang, Yixin Ren, Yewei Xia, Shuigeng Zhou, and Jihong Guan. Towards effective causal partitioning by edge cutting of adjoint graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Keli Zhang, Shengyu Zhu, Marcus Kalander, Ignavier Ng, Junjian Ye, Zhitang Chen, and Lujia Pan. gcastle: A python toolbox for causal discovery, 2021.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31, 2018.

A PROOFS

A.1 PROOF FOR THEOREM 1

Theorem 1: If $R_i \not\perp\!\!\!\perp R_j$ and $\tau_i < \tau_j$, then R_i is a non-descendant group of R_j.

To prove Theorem 1, we first introduce and prove Lemma 1:

Lemma 1. *If $A \to B$ then $|\Phi(A)| \leq |\Phi(B)|$. Equality occurs if all parents of B are dependent on A.*

Proof. For any $C \in \Phi(A)$, there exists an unblocked path $\mathfrak{P}$ connecting C to A. Then, the path $\mathfrak{P} \cup \{B\}$ is also unblocked because $A \to B$. Therefore, there exists the unblocked path $\mathfrak{P} \cup \{B\}$ connecting C to B, thus $C \not\perp\!\!\!\perp B$ or $C \in \Phi(B)$. In conclusion, $\forall C \in \Phi(A) : C \in \Phi(B) \Leftrightarrow \Phi(A) \subseteq \Phi(B) \Leftrightarrow |\Phi(A)| \leq |\Phi(B)|$. If B has a parent D that is independent of A, then it follows that $|\Phi(A)| < |\Phi(A) \cup \Phi(D)| = |\Phi(B)|$. □

Proof of Theorem 1. By **contradiction**. If R_i is a descendant group of R_j, then by Lemma 1, it must follow that $\tau_i \geq \tau_j$, which contradicts the assumption. □

A.2 PROOF FOR THEOREM 2

Theorem 2: $\forall (X_i, X_j) \in \bigcup_{j \in \mathrm{Pa}(R_i)} \hat{G}_{ij} \cup \hat{G}_i, \exists \mathcal{Z} \subset \boldsymbol{X} : X_i \perp\!\!\!\perp_G X_j | \mathcal{Z} \Leftrightarrow \exists \mathcal{Z}' \subset R_k \cup \mathrm{Pa}(R_k) : X_j \perp\!\!\!\perp_G X_i | \mathcal{Z}'$.

Proof. As per the definition of $\hat{G}_i$ and $\hat{G}_{ij}$, for any pair (X_i, X_j), there are two cases:
(i) $X_i \in R_k, X_j \in \mathrm{Pa}(R_k)$. Then $\exists \mathcal{Z} \subset \boldsymbol{X} : X_i \perp\!\!\!\perp_G X_j \mid \mathcal{Z} \Leftrightarrow [X_j \notin \mathrm{Pa}(X_i)] \wedge [X_j \notin \mathrm{Desc}(X_i)] \Leftrightarrow X_i \perp\!\!\!\perp X_j \mid \mathrm{Pa}(X_i)$ (Markov Property). Since $\mathrm{Pa}(X_i) \subset [R_k \cup \mathrm{Pa}(R_k)] \Rightarrow \mathcal{Z}' = \mathrm{Pa}(X_i)$ satisfies.
(ii) $X_i, X_j \in R_k$. Then, with similar arguments, either $\mathcal{Z}' = \mathrm{Pa}(X_i)$ or $\mathcal{Z}' = \mathrm{Pa}(X_j)$ satisfies. □
The fact that both $\mathrm{Pa}(X_i)$ and $\mathrm{Pa}(X_j)$ are available in the subproblem is formulation of the subproblem (Remark 1).

A.3 PROOF FOR THEOREM 3

Theorem 3: Let G denote the edge-wise aggregated graph of $\{\hat{G}_k\}_{k=1}^K$. If $\forall k : \hat{G}_k$ is a DAG, then G is also a DAG.

Proof. By contradiction. Let us examine $X_i \in \hat{G}_k$. If there exists a path $\mathfrak{P} := \{X_i \to \cdots \to X_i\}$, there are two cases:

(i) Path $\mathfrak{P}$ is fully contained in $\hat{G}_k$, which contradicts the assumption that $\hat{G}_k$ is acyclic.

(ii) Path $\mathfrak{P}$ contains nodes from other groups than R_k. Then, there must exist a path from R_k to other groups and returns to R_k. Then G_{R} contains a cycle, which contradicts the fact that algorithm 1 always returns an identifiable DAG (Theorem 5). □

A.4 BUILDING G_{R}

In this section, we present Algorithm 1 and prove the identifiability of G_{R}.

Theorem 5 (Identifiability)**.** *G_{R} is fully identifiable using Algorithm 1.*

Proof. **By Induction.** Predicate $P(n)$: $n \geq 1$ first groups in the topological order π correctly recover their causal parent groups using Algorithm 1.

Base case. Let us examine $P(1)$: The first group has no causal parent group as it has no preceding group in the topological order. Therefore, $P(1)$ is trivial.

Induction step. Assume $P(k)$, we want to show that $P(k+1)$ is true. Indeed, let us examine $R_{\pi_{k+1}}$ whose parents are among $\{R_{\pi_i}\}_{i=1}^k$ (**Line 9**). We then analyze every possible edge $(R_{\pi_i}, R_{\pi_{k+1}})_{i=1}^k$ to see if such an edge is spurious or not (**Line 10**). If not, we conclude that $R_{\pi_i} \in \mathrm{Pa}(R_{\pi_{k+1}})$.

Algorithm 1 Build causal graph of groups G_{R}

Input: data x_i, size m
function EXAMINE(R_{π_i}, R_{π_k})
  Get current $\mathrm{Pa}(R_{\pi_i})$
  Get current $\mathrm{Ch}(R_{\pi_i})$ from $\{R_{\pi_i}\}_{i=1}^{k}$
  Get current $\mathrm{Sp}(R_{\pi_i})$ from $\{R_{\pi_i}\}_{i=1}^{k}$
  return CIT(R_{π_i}, R_{π_k}, $\mathrm{Pa}(R_{\pi_i}) \cup \mathrm{Ch}(R_{\pi_i}) \cup \mathrm{Sp}(R_{\pi_i})$)
end function
Find topological order π (Theorem 1)
Initialize parent set of all groups $\mathrm{Pa}(R_{\pi_k}) \leftarrow \{\emptyset\}$
for $k = 2$ to K **do**
  for $i = 1$ to $k-1$ **do**
    if $\neg$EXAMINE(R_{π_i}, R_{π_k}) **then**
      $\mathrm{Pa}(R_{\pi_k}) = \mathrm{Pa}(R_{\pi_k}) \cup \{R_{\pi_i}\}$
    end if
  end for
end for
Build G_{R} from $\mathrm{Pa}(R_k), \forall k \in [1, K]$
return G_{R}

To isolate the causal effect between R_{π_i} and $R_{\pi_{k+1}}$, we need to block on all active[2] paths (backdoor[3] and front-door[4] paths). For backdoor paths, it suffices to conditioned on $\mathrm{Pa}(R_{\pi_i})$, which is known per assumption $P(k), k \geq i$ (**Line 2**). For front-door paths, it suffices to conditioned on known child and spouse groups of R_{π_i}. Because unknown (not yet examined by the algorithm) child groups of R_{π_i} lie behind R_{π_k} in the topological order, such groups are not in any backdoor or front-door path. For known child groups of R_{π_i}, as we conditioned on them (**Line 3**), we potentially create active paths via spouse groups of R_{π_i}, which is why we also need to condition those groups (**Line 4**). Notice that all of these groups are examined, and causally uncovered as assumed by $P(k)$.

To this end, we have completely isolated the causal effect between R_{π_i} and $R_{\pi_{k+1}}$. A simple statisicial test [$R_{\pi_i} \not\perp\!\!\!\perp R_{\pi_{k+1}} \mid$ known $\mathrm{Pa}(R_{\pi_i}) \cup \mathrm{Ch}(R_{\pi_i}) \cup \mathrm{Sp}(R_{\pi_i})$] will reveal if $(R_{\pi_i}, R_{\pi_{k+1}})$ is spurious or causal. Repeat the above procedure for all pairs, we find $\mathrm{Pa}(R_{\pi_{k+1}})$, hence $P(k+1)$. **Induction completes**. □

Theorem 6 (G_{R} Computational Complexity)**.** *Algorithm 1 requires $\mathcal{O}(d^2)$ conditional independence tests (CITs).*

Proof. Let S denote the total number of CITs performed by Algorithm 1. According to the procedure, we have: $S = \sum_{k=2}^{K} \sum_{i=1}^{k-1} |R_{\pi_i}||R_{\pi_k}|$. Since π is only a permutation, without the loss of generality, S can be rewritten without π:

$$S = \sum_{1 \leq i < k \leq K} |R_i||R_k| \quad \text{s.t.} \sum_{j=1}^{K} |R_j| = d. \tag{2}$$

From the constraint, we derive that:

$$d^2 = \left(\sum_{j=1}^{K} |R_j| \right)^2 = \sum_{j=1}^{K} |R_j|^2 + 2S \Rightarrow S = \frac{1}{2} \left(d^2 - \sum_{j=1}^{K} |R_j|^2 \right). \tag{3}$$

Moreover, Cauchy-Schwarz inequality writes:

$$d^2 = \left(\sum_{j=1}^{K} |R_j| \right)^2 \leq \left(\sum_{j=1}^{K} |R_j|^2 \right) \left(\sum_{j=1}^{K} 1^2 \right) = K \left(\sum_{j=1}^{K} |R_j|^2 \right) \Rightarrow \sum_{j=1}^{K} |R_j|^2 \geq \frac{d^2}{K} \tag{4}$$

[2] An active path between two variables is a path containing no colliders with respect to the two variables.
[3] A path $X \leftarrow \cdots \rightarrow Y$ is called a backdoor path from X to Y.
[4] A path $X \rightarrow \cdots \rightarrow Y$ is called a front-door path from X to Y.

From equation 3 and equation 4, we bound S as:

$$S \leq \frac{1}{2}\left(d^2 - \frac{d^2}{K}\right) = \frac{K-1}{2K}d^2 = \mathcal{O}(d^2) \quad \square \tag{5}$$

B EXPERIMENTAL SETTINGS, EXTRA RESULTS

B.1 DETAILS ON EXPERIMENTAL SETTINGS

Table 4: **Parameters settings.**

Routine	Parameters	Standalone	Integrated
PC	Fisher's test threshold	0.3	
	Chi-square test threshold	0.05	
GOLEM	Number of iterations	10000	500
	λ_1	0.02	
	λ_2	5	
	Weight dropping threshold	0.3	
	Learning rate	0.005	0.1
GraNDAG	Number of hidden layers	2	
	Number of iterations	1000	
	Learning rate	0.005	0.01
SCORE	η_H	0.01	
	η_G	0.01	
	Threshold	0.01	
Varsort	No parameters		

For all experiments, we set the sample size to 1000. Table 4 details our parameters and hyperparameters setting for our reported empirical studies. These parameters are applied to ours as well as all baseline frameworks. In most cases, we reuse the recommended parameters in the subroutines' original paper. For deep learning subroutines, such as GOLEM and GraNDAG, we significantly increase the learning rate. In the case of GOLEM, however, we also decrease the number of iterations by a proportionally significant amount since subproblems are supposedly much easier to solve than the original. We run all experiments on a single machine, using CPU resources only. Hyperparameters σ and k_{par} used by CPA and CAPA, respectively, are set at 1.

B.2 PERFORMANCE UNDER FINITE-DATA REGIME

Here, we validate **DEFT** under the finite-data regime by replacing the oracle CI tests by sample-based CI tests. We first investigate the effect of imperfect CI tests on the formation of groups and its consequences on the final performance of our framework.

Grouping effectiveness. We evaluate to what extent grouping under finite-sample CI is consistent with grouping under oracle CI. We treat the finite-sample grouping as a prediction of whether each pair of variables (X, Y) belongs to the same group, and evaluate this prediction over all variable pairs using standard binary classification metrics. Results are summarized in Table 5, reported over multiple random graphs with $d = 100$ nodes each, amounting to approximately 50000 test cases.

Table 5: Grouping correctness under finite-sample CI regime.

Metric (%)	**Value**
TP	40.33 ± 3.66
TN	51.20 ± 3.44
FP	2.64 ± 0.66
FN	5.83 ± 1.67
Precision	93.84 ± 1.49
Recall	87.33 ± 3.53
F1-score	90.43 ± 2.02

These results indicate that grouping remains relatively robust under finite-sample CI. While performance decreases (as expected) under finite-sample CI (Table 6), DEFT continues to consistently outperform baseline methods, indicating significantly more robustness to imperfect finite-sample CI over prior SOTA methods.

Performance against state-of-the-art baselines. We compare **DEFT** with the best baseline framework HCD on synthetic Erdos-Renyi graphs with graph size increasing from 100 to 500. The results are summarized over 10 independent runs, and are reported in Table 6.

Table 6: **Performance comparison between DEFT and HCD for increasingly larger graphs.** PC as the local solver.

d	Framework	Precision	Recall	F1-score	Spurious	SHD
100	**DEFT**	0.48 ± 0.04	0.63 ± 0.06	0.54 ± 0.05	0.43 ± 0.05	189.75 ± 25.94
	HCD	0.20 ± 0.04	0.17 ± 0.03	0.18 ± 0.03	0.48 ± 0.07	271.00 ± 29.93
200	**DEFT**	0.50 ± 0.04	0.65 ± 0.06	0.57 ± 0.04	0.39 ± 0.03	357.60 ± 34.92
	HCD	0.15 ± 0.03	0.05 ± 0.02	0.07 ± 0.02	0.44 ± 0.06	459.60 ± 20.98
300	**DEFT**	0.42 ± 0.05	0.58 ± 0.04	0.48 ± 0.05	0.45 ± 0.05	649.60 ± 31.03
	HCD	0.16 ± 0.03	0.04 ± 0.01	0.06 ± 0.01	0.44 ± 0.08	661.30 ± 29.50
400	**DEFT**	0.35 ± 0.03	0.48 ± 0.03	0.40 ± 0.03	0.49 ± 0.03	966.60 ± 58.11
	HCD	0.15 ± 0.04	0.02 ± 0.01	0.04 ± 0.01	0.42 ± 0.07	846.60 ± 12.89
500	**DEFT**	0.33 ± 0.02	0.44 ± 0.04	0.38 ± 0.03	0.50 ± 0.01	1242.00 ± 39.68
	HCD	0.12 ± 0.04	0.01 ± 0.01	0.03 ± 0.01	0.44 ± 0.06	1049.90 ± 14.03

The results show that **DEFT** consistently achieves much stronger overall performance than HCD across all dimensions. For example, **DEFT** improves the F1-score from 0.18 to 0.54 at $d = 100$, from 0.07 to 0.57 at $d = 200$, and remains far better even at $d = 500$, where HCD drops to only 0.03 while **DEFT** still obtains 0.38. This gain is largely driven by recall: **DEFT** maintains recall between 0.44 and 0.65, whereas HCD rapidly collapses from 0.17 to 0.01 as d increases, indicating that HCD misses most true edges in high dimensions. Although the SHD of **DEFT** starts to rise over that of HCD for larger d values, due to the effect of an imperfect grouping strategy – only 1000 samples are observed for d variables. Overall, **DEFT** provides a much better accuracy–recall trade-off and remains substantially more reliable than HCD as dimensionality increases.

B.3 Linear non-Gaussian data-generating model

Table 7: **Performance boosting for different subroutines.** Data-generating model is linear additive non-Gaussian noise. "NA" denotes experiments that fail to produce results within 24 hours.

		Precision ↑	Recall ↑	F1-score ↑	Spurious ↓	SHD ↓	Runtime(s) ↓
(A) $d = 400$; $e = 800$	PC	NA	NA	NA	NA	NA	NA
	+DEFT	0.77 ± 0.02	0.95 ± 0.02	0.85 ± 0.02	0.15 ± 0.02	236.6 ± 24.41	26.72 ± 6.65
	GOLEM	0.31 ± 0.02	0.09 ± 0.01	0.14 ± 0.01	0.09 ± 0.03	750.56 ± 8.37	285.54 ± 5.96
	+DEFT	0.65 ± 0.04	0.45 ± 0.02	0.53 ± 0.03	0.34 ± 0.04	625.2 ± 39.97	1054.7 ± 30.46
	GraNDAG	0.36 ± 0.17	0.02 ± 0.01	0.03 ± 0.02	0.14 ± 0.28	796.1 ± 24.03	703.24 ± 14.99
	+DEFT	0.85 ± 0.03	0.32 ± 0.03	0.46 ± 0.03	0.04 ± 0.01	556.1 ± 22.56	5535.5 ± 1464.9
	ICALiNGAM	0.43 ± 0.07	0.06 ± 0.01	0.1 ± 0.01	0.0 ± 0.0	754.5 ± 6.52	508.5 ± 80.59
	+DEFT	0.93 ± 0.03	0.16 ± 0.01	0.28 ± 0.02	0.0 ± 0.0	669.3 ± 11.05	24.2 ± 4.11
	Varsort	0.2 ± 0.02	0.26 ± 0.02	0.23 ± 0.02	0.31 ± 0.01	913.6 ± 20.67	865.05 ± 93.23
	+DEFT	0.81 ± 0.02	0.87 ± 0.01	0.84 ± 0.01	0.1 ± 0.01	194.7 ± 18.06	19.24 ± 3.82
(B) $d = 500$; $e = 1000$	PC	NA	NA	NA	NA	NA	NA
	+DEFT	0.76 ± 0.03	0.95 ± 0.01	0.84 ± 0.02	0.16 ± 0.02	317.3 ± 43.43	46.65 ± 13.49
	GOLEM	0.34 ± 0.03	0.1 ± 0.02	0.15 ± 0.02	0.1 ± 0.04	929.8 ± 18.67	603.91 ± 29.69
	+DEFT	0.66 ± 0.03	0.45 ± 0.01	0.54 ± 0.02	0.33 ± 0.03	771.4 ± 34.06	740.07 ± 52.47
	GraNDAG	0.41 ± 0.19	0.01 ± 0.01	0.02 ± 0.01	0.17 ± 0.33	1012.0 ± 39.78	1441.24 ± 184.44
	+DEFT	0.83 ± 0.03	0.31 ± 0.04	0.45 ± 0.04	0.05 ± 0.01	708.3 ± 37.19	4810.28 ± 135.72
	ICALiNGAM	0.46 ± 0.03	0.06 ± 0.01	0.1 ± 0.01	0.0 ± 0.0	942.2 ± 5.98	1413.09 ± 300.97
	+DEFT	0.93 ± 0.02	0.16 ± 0.01	0.27 ± 0.02	0.0 ± 0.0	839.9 ± 11.1	47.71 ± 11.37
	Varsort	0.21 ± 0.02	0.27 ± 0.02	0.23 ± 0.02	0.31 ± 0.01	1134.2 ± 46.77	656.93 ± 120.5
	+DEFT	0.81 ± 0.02	0.86 ± 0.01	0.84 ± 0.01	0.11 ± 0.01	251.1 ± 24.24	29.63 ± 5.23

Table 7 shows the performance of our framework in boosting subroutines against scenarios where the data is generated from non-Gaussian additive noises. We omit SCORE from this section as the algorithm is not designed for non-Gaussian additive noise models. Consequentially, the solver of SCORE fails to return results for this study.

Overall, **DEFT** boosts all subroutines in terms of Precision, Recall, and F1-score. Indeed, the average increment in Precision is $+0.56$ in scenario (A) and $+0.492$ in scenario (B). At the same time, we also see an equally significant increase in Recall: $+0.36$ and $+0.466$ in scenarios (A) and (B), respectively. Together, we find that **DEFT** systematically levels the performance of all subroutines by large margins, with the average of $+0.452$. This, in turn, allows our framework to achieve much lower SHD compared to all subroutines.

Regarding spurious rates, we also witness general improvements with a singular exception for the subroutine GOLEM. It appears that GOLEM **+DEFT** resulted in much higher spurious relations despite having higher Recall score. Our conjecture is that, for non-Gaussian additive noise models,

GOLEM's neural network solver requires extra information to better estimating the graph – which agrees with existing model-based methods that generally do not require Causal Sufficiency.

As for runtime, we observe that deep learning subroutines (i.e., GOLEM, GraNDAG) when integrated into **DEFT** often suffer from increased runtime. As we explained in the main text, this is mostly due to the re-initialization of neural networks and their solvers. Except from these subroutines, others benefit from an average of 14.49 times shorter runtime.

Remark 4. **DEFT** is most beneficial when the base solver's cost is dominated by graph size. For neural/optimization-heavy solvers such as GraNDAG and GOLEM, repeated re-initialization and optimization over many subproblems can offset the gains from smaller problem size, especially when convergence requires many iterations. When non-Gaussian noise is introduced, the optimization of NOTEARS-like methods becomes significantly harder (e.g., slower convergence, noisier loss landscape). When such optimization must be repeated across multiple subproblems in **DEFT**, the cumulative complexity can outweigh the benefits of decomposition.

B.4 Performance under non-linear Additive Noise Models (ANMs) with non-Gaussian noise.

In addition to the linear non-Gaussian results in Appendix B.3, we conduct experiments on nonlinear ANMs with Tanh, Sigmoid, and Sine additive noise on the dataset WIN95PTS. For the local solver, we choose GOLEM as the representative for NOTEARS family of methods, and Varsort as the strongest representative for other methods. We report the results from 10 independent runs in Table 8.

Table 8: Performance comparison across ANM types and subroutines.

ANM type	Subroutines	Precision↑	Recall↑	F1-score↑	Spurious↓	SHD↓
Tanh	Varsort	0.551 ± 0.058	0.887 ± 0.027	0.678 ± 0.052	0.380 ± 0.053	82.6 ± 18.530
	+DEFT	0.874 ± 0.019	0.969 ± 0.011	0.919 ± 0.015	0.098 ± 0.012	15.7 ± 2.584
	GOLEM	0.997 ± 0.006	0.632 ± 0.021	0.774 ± 0.017	0.00 ± 0.00	41.20 ± 2.394
	+DEFT	0.988 ± 0.010	0.641 ± 0.024	0.777 ± 0.019	0.00 ± 0.00	40.20 ± 2.658
Sigmoid	Varsort	0.745 ± 0.033	0.843 ± 0.033	0.791 ± 0.032	0.121 ± 0.024	33.0 ± 4.830
	+DEFT	0.938 ± 0.022	0.957 ± 0.019	0.947 ± 0.019	0.025 ± 0.018	7.7 ± 2.983
	GOLEM	0.600 ± 0.516	0.008 ± 0.011	0.016 ± 0.021	0.00 ± 0.00	111.10 ± 1.197
	+DEFT	0.960 ± 0.040	0.176 ± 0.039	0.296 ± 0.057	0.032 ± 0.040	93.00 ± 4.320
Sine	Varsort	0.677 ± 0.072	0.435 ± 0.046	0.527 ± 0.040	0.212 ± 0.060	79.2 ± 5.884
	+DEFT	0.833 ± 0.044	0.512 ± 0.058	0.632 ± 0.045	0.122 ± 0.043	63.3 ± 6.075
	GOLEM	0.888 ± 0.076	0.093 ± 0.027	0.167 ± 0.045	0.00 ± 0.00	101.60 ± 3.062
	+DEFT	0.938 ± 0.080	0.097 ± 0.029	0.176 ± 0.048	0.00 ± 0.00	101.10 ± 3.247

It can be seen that **DEFT** is the main factor driving the performance improvements across the different ANM types. When applied to Varsort, **DEFT**'s impact is especially strong in the Sigmoid setting, where the SHD decreases from 33.0 to 7.7, indicating a much closer match to the true graph. When combined with GOLEM, **DEFT** gives more moderate but still noticeable improvements, particularly for Sigmoid and Sine cases where GOLEM's performance is weak. Overall, these results suggest that **DEFT** acts as an effective booster across non-Gaussian noises.

B.5 Performance under different graph generators

Beyond the Erdos-Renyi class of graphs, we also tested bipartite (BP) and scale-free (SF) graphs on the same settings as in Section 4.1. The experiment is conducted and summarized over 5 independent generated graphs of each class.

Table 9 shows the performance boost on bipartite graph settings. Overall, **DEFT** substantially improves most methods, generally increasing precision, recall, and F1-score while reducing spurious edges and SHD, although the effect on runtime varies across algorithms. The most striking improvement is observed for PC, whose F1-score rises from 0.405 to 0.890 in setting A and from 0.395 to 0.894 in setting B. GOLEM and VarSort also benefit considerably, with their **DEFT**-augmented versions achieving near-perfect F1-scores and much lower SHD, while **DEFT** enables SCORE to obtain F1-scores of 0.942 and 0.944 despite the original SCORE failing to return meaningful results within 24 hours. However, **DEFT** slightly reduces the F1-score of ICALiNGAM in this setting, although it still makes ICALiNGAM substantially faster in both settings.

The same observation is also seen in the scale-free settings (see Table 10). The largest gains are observed for ICALiNGAM, whose F1-score increases from 0.388 to 0.877 in setting A and from 0.405 to 0.869 in setting B, accompanied by dramatic runtime reductions up to $10\times$ shorter. VarSort also performs particularly well with **DEFT**, achieving F1-scores of 0.985 and 0.956 and reducing SHD from 144.8 to 20.8 and from 284.2 to 88.0, while SCORE improves from extremely low F1-scores of 0.047 and 0.041 to 0.580 and 0.584, respectively. Although the original PC fails to produce results within 24 hours in both settings, **DEFT** enables it to achieve F1-scores of 0.753 and 0.743.

Extra runtime is observed in the case of GOLEM and GraNDAG albeit with significantly higher performance metrics. Again, this is consistent with our explanation provided in Appendix B.3: cumulative delay caused by re-initialization of neural network solvers.

Table 9: **Performance boosting for different subroutines on Bipartite graphs.** SCORE fails to return meaningful results despite running for 24 hours, thus is reported as "—" in this experiment.

		Precision ↑	Recall ↑	F1-score ↑	Spurious ↓	SHD ↓	Runtime(s) ↓
(A) $d = 400$; $e = 800$	PC	0.36 ± 0.027	0.464 ± 0.052	0.405 ± 0.037	0.481 ± 0.014	923.4 ± 35.949	2908.44 ± 258.88
	+DEFT	0.803 ± 0.016	0.998 ± 0.002	0.89 ± 0.011	0.131 ± 0.013	193.8 ± 19.588	5.59 ± 0.64
	GOLEM	0.955 ± 0.01	0.982 ± 0.005	0.969 ± 0.006	0.044 ± 0.01	50.4 ± 10.407	92.21 ± 5.67
	+DEFT	1.0 ± 0.0	0.999 ± 0.002	0.999 ± 0.001	0.0 ± 0.0	1.0 ± 1.225	350.04 ± 9.79
	GraNDAG	0.967 ± 0.03	0.12 ± 0.044	0.211 ± 0.073	0.028 ± 0.024	707.4 ± 33.216	372.69 ± 6.43
	+DEFT	0.998 ± 0.001	0.733 ± 0.01	0.845 ± 0.007	0.0 ± 0.001	214.0 ± 8.276	2060.61 ± 49.12
	ICALiNGAM	0.992 ± 0.005	0.996 ± 0.003	0.994 ± 0.004	0.006 ± 0.005	8.0 ± 5.568	143.51 ± 19.51
	+DEFT	0.977 ± 0.004	0.978 ± 0.005	0.977 ± 0.005	0.001 ± 0.001	18.8 ± 3.834	33.99 ± 1.24
	SCORE	—	—	—	—	—	—
	+DEFT	0.986 ± 0.012	0.901 ± 0.012	0.942 ± 0.012	0.013 ± 0.012	88.4 ± 16.994	717.16 ± 13.41
	Varsort	0.945 ± 0.011	1.0 ± 0.0	0.972 ± 0.006	0.055 ± 0.011	46.6 ± 9.633	24.89 ± 0.44
	+DEFT	0.995 ± 0.002	1.0 ± 0.0	0.998 ± 0.001	0.005 ± 0.002	3.8 ± 1.789	3.25 ± 0.12
(B) $d = 500$; $e = 1000$	PC	0.337 ± 0.032	0.476 ± 0.041	0.395 ± 0.035	0.508 ± 0.014	1243.25 ± 59.084	8355.11 ± 1132.42
	+DEFT	0.809 ± 0.013	0.999 ± 0.001	0.894 ± 0.008	0.122 ± 0.006	227.8 ± 17.079	7.15 ± 0.67
	GOLEM	0.962 ± 0.01	0.985 ± 0.002	0.974 ± 0.005	0.038 ± 0.01	53.4 ± 11.104	158.13 ± 0.85
	+DEFT	1.0 ± 0.0	1.0 ± 0.001	1.0 ± 0.0	0.0 ± 0.0	0.4 ± 0.548	430.89 ± 5.83
	GraNDAG	0.898 ± 0.057	0.084 ± 0.027	0.153 ± 0.046	0.096 ± 0.054	926.4 ± 20.132	607.08 ± 15.12
	+DEFT	0.999 ± 0.001	0.731 ± 0.017	0.845 ± 0.011	0.0 ± 0.0	268.6 ± 17.184	2593.68 ± 98.89
	ICALiNGAM	0.983 ± 0.007	0.994 ± 0.002	0.988 ± 0.005	0.014 ± 0.006	20.4 ± 8.173	420.34 ± 70.48
	+DEFT	0.978 ± 0.003	0.978 ± 0.003	0.978 ± 0.003	0.001 ± 0.001	22.2 ± 3.033	68.12 ± 57.58
	SCORE	—	—	—	—	—	—
	+DEFT	0.989 ± 0.005	0.903 ± 0.012	0.944 ± 0.008	0.009 ± 0.004	105.4 ± 14.859	896.47 ± 13.81
	Varsort	0.941 ± 0.004	1.0 ± 0.0	0.97 ± 0.002	0.059 ± 0.004	62.2 ± 4.494	46.73 ± 0.13
	+DEFT	0.993 ± 0.002	1.0 ± 0.0	0.997 ± 0.001	0.007 ± 0.002	6.8 ± 1.643	4.28 ± 0.12

Table 10: **Performance boosting for different subroutines on Scale-free graphs.** "NA" denotes experiments that fail to produce results within 24 hours.

		Precision ↑	Recall ↑	F1-score ↑	Spurious ↓	SHD ↓	Runtime(s) ↓
(A) $d = 400$; $e = 800$	PC	NA	NA	NA	NA	NA	NA
	+DEFT	0.826 ± 0.02	0.692 ± 0.027	0.753 ± 0.023	0.072 ± 0.008	356.333 ± 31.134	16823.50 ± 1404.72
	GOLEM	0.594 ± 0.082	0.828 ± 0.128	0.69 ± 0.096	0.387 ± 0.073	568.4 ± 161.483	89.50 ± 7.02
	+DEFT	0.954 ± 0.061	0.708 ± 0.052	0.81 ± 0.009	0.045 ± 0.062	263.8 ± 9.149	188.76 ± 13.74
	GraNDAG	0.403 ± 0.179	0.141 ± 0.049	0.198 ± 0.057	0.589 ± 0.18	894.8 ± 122.06	370.49 ± 7.70
	+DEFT	0.94 ± 0.078	0.404 ± 0.047	0.563 ± 0.048	0.054 ± 0.08	495.8 ± 49.256	1205.87 ± 43.72
	ICALiNGAM	0.304 ± 0.022	0.537 ± 0.041	0.388 ± 0.028	0.619 ± 0.022	1241.0 ± 60.428	168.84 ± 18.53
	+DEFT	0.905 ± 0.038	0.853 ± 0.07	0.877 ± 0.047	0.054 ± 0.031	157.6 ± 65.194	29.85 ± 2.05
	SCORE	0.055 ± 0.011	0.041 ± 0.008	0.047 ± 0.009	0.361 ± 0.033	977.0 ± 18.138	1472.16 ± 30.19
	+DEFT	0.958 ± 0.012	0.416 ± 0.017	0.58 ± 0.016	0.04 ± 0.012	479.2 ± 14.36	734.83 ± 39.45
	Varsort	0.846 ± 0.033	0.99 ± 0.003	0.913 ± 0.02	0.147 ± 0.031	144.8 ± 35.996	25.08 ± 0.39
	+DEFT	0.98 ± 0.006	0.99 ± 0.009	0.985 ± 0.005	0.016 ± 0.005	20.8 ± 6.834	6.80 ± 0.45
(B) $d = 500$; $e = 1000$	PC	NA	NA	NA	NA	NA	NA
	+DEFT	0.838 ± 0.03	0.668 ± 0.02	0.743 ± 0.02	0.044 ± 0.004	452.0 ± 24.15	32286.99 ± 2143.15
	GOLEM	0.577 ± 0.125	0.777 ± 0.176	0.659 ± 0.136	0.404 ± 0.123	776.0 ± 311.098	160.78 ± 4.19
	+DEFT	0.938 ± 0.056	0.743 ± 0.024	0.828 ± 0.013	0.062 ± 0.056	308.2 ± 33.147	229.13 ± 7.19
	GraNDAG	0.42 ± 0.18	0.101 ± 0.026	0.158 ± 0.038	0.574 ± 0.18	1070.2 ± 115.09	611.12 ± 16.11
	+DEFT	0.949 ± 0.043	0.417 ± 0.032	0.578 ± 0.03	0.042 ± 0.044	601.2 ± 32.36	1526.35 ± 41.38
	ICALiNGAM	0.323 ± 0.027	0.545 ± 0.024	0.405 ± 0.026	0.603 ± 0.028	1476.4 ± 118.919	449.52 ± 89.64
	+DEFT	0.891 ± 0.037	0.853 ± 0.061	0.869 ± 0.025	0.075 ± 0.039	221.2 ± 38.408	41.92 ± 4.08
	SCORE	0.05 ± 0.024	0.035 ± 0.013	0.041 ± 0.017	0.355 ± 0.04	1218.25 ± 55.108	2302.58 ± 23.62
	+DEFT	0.953 ± 0.011	0.421 ± 0.021	0.584 ± 0.02	0.042 ± 0.013	595.8 ± 19.241	978.33 ± 46.10
	Varsort	0.779 ± 0.046	0.991 ± 0.002	0.872 ± 0.03	0.214 ± 0.046	284.2 ± 77.393	47.04 ± 0.36
	+DEFT	0.936 ± 0.065	0.979 ± 0.007	0.956 ± 0.037	0.06 ± 0.065	88.0 ± 79.234	10.06 ± 0.59

B.6 Ablation study on the number of groups K

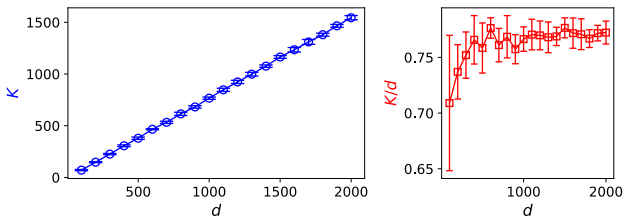


Figure 5: Statistics of K on Erdos-Renyi class of graphs. Left: K w.r.t. d. Right: K/d w.r.t. d.

Theorem 4 indicates that the decomposition procedure of **DEFT** scales quadratically with respect to the number of variables d and linearly to the number of groups K. It is worth noting that $K \leq d$ due to the pigeon hole principle. In this ablation study, we want to establish the scale of K with respect to d.

For this, we randomly generate 10 Erdos-Renyi graphs of size $d \in \{100, 200, \ldots, 2000\}$ and using Definition 2 to formulate the groups and count for K. Figure 5 shows the mean value of K along with its standard deviation. In this particular class of graphs (Erdos-Renyi), we observe an almost linear relation between K and d (i.e, $K = 0.77d - 6.5$ with $R^2 \approx 0.99$).

However, when it comes to real-world graphs as provided by the bnlearn package Scutari (2010), K can be of arbitrarily lower order than d. For this, it is important to note that the K is graph-dependent and thus does not invalidate our theoretical analysis showing that **DEFT**'s decomposition process scales quadratically with respect to the number of variables d.

Remark 5. We emphasize that **DEFT** provides a structure-adaptive scalability improvement. In the worst-case scenarios:

(i) All variables in one group: **DEFT** reduces to the base routine, incurring no benefits nor loss.
(ii) Each variable forms its own group: the group graph coincides with the variable graph, and **DEFT** reduces to constructing this graph in $O(d^2)$, comparable to existing state-of-the-art approaches.

Therefore, **DEFT** is never worse (in order) than standard methods in the worst case, while providing provable improvements when non-trivial grouping exists. Empirically, across both synthetic and real-world graphs (Tables 2–3), we consistently observe the presence of non-trivial grouping, under which **DEFT** delivers substantial scalability and performance gains.

B.7 Size of Subproblems

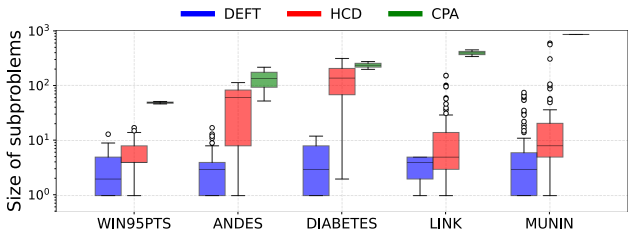


Figure 6: **Size of subproblems on Bnlearn datasets.** Y-axis is in logarithmic scale. The largest subproblem size of HCD is 597, whereas that of our framework is 74.

The size of subproblems defines the workload for the subroutine $\mathcal{A}$, which affects the framework's overall time complexity. Trivially, the size of subproblems is inversely correlated with the number of subproblems. However, since the total computational complexity scales linearly with the number of subproblems but exponentially with the size of subproblems (Theorem 4), we prefer smaller subproblems over less subproblems. Figure 6 shows the distribution of the size of subproblems across baseline frameworks which indicates that our proposal, **DEFT**, is the most favorable. Compared to HCD, the third quartile produced by **DEFT** is almost always lower than the first quartile of HCD.

Noticeably, the maximum size of subproblems produced by **DEFT** is strictly smaller than that by CPA, up to $10\times$ smaller in large-sized graphs such as in the Link and Munin datasets. CPA, on the other hand, always produces two subproblems, which are much larger compared to both HCD's and ours.

C LIMITATIONS

Our proposal uses statistical independence tests for its variable partitioning. This poses two challenges: the reliability and computational complexity of CI tests. Through ablation studies, we show that **DEFT** still outperforms existing state-of-the-art frameworks under the finite-data regime. As for the computational complexity of CI tests, **DEFT** may demand conditioning on a larger number of variables. These limitations, in turn, open up possibilities for further improvements in the future.

Declaration of LLM usage. OpenReview's LLM-generated reviews are used for polishing the paper and correcting typos.