Journal of Machine Learning Research 26 (2025) 1-65 Submitted 3/24; Revised 11/24; Published 3/25

Recursive Causal Discovery

Ehsan Mokhtarian EHSAN.MOKHTARIAN@EPFL.CH
School of Computer and Communication Sciences
EPFL
1015 Lausanne, Switzerland

Sepehr Elahi SEPEHR.ELAHI@EPFL.CH
School of Computer and Communication Sciences
EPFL
1015 Lausanne, Switzerland

Sina Akbari SINA.AKBARI@EPFL.CH
School of Computer and Communication Sciences
EPFL
1015 Lausanne, Switzerland

Negar Kiyavash NEGAR.KIYAVASH@EPFL.CH
College of Management of Technology
EPFL
1015 Lausanne, Switzerland

Editor: Ilya Shpitser

Abstract

Causal discovery from observational data, i.e., learning the causal graph from a finite set of samples from the joint distribution of the variables, is often the first step toward the identification and estimation of causal effects, a key requirement in numerous scientific domains. Causal discovery is hampered by two main challenges: limited data results in errors in statistical testing and the computational complexity of the learning task is daunting. This paper builds upon and extends four of our prior publications (Mokhtarian et al., 2021; Akbari et al., 2021; Mokhtarian et al., 2022, 2023a). These works introduced the concept of *removable* variables, which are the only variables that can be removed recursively for the purpose of causal discovery. Presence and identification of removable variables allow recursive approaches for causal discovery, a promising solution that helps to address the aforementioned challenges by reducing the problem size successively. This reduction not only minimizes conditioning sets in each conditional independence (CI) test, leading to fewer errors but also significantly decreases the number of required CI tests. The worst-case performances of these methods nearly match the lower bound. In this paper, we present a unified framework for the proposed algorithms, refined with additional details and enhancements for a coherent presentation. A comprehensive literature review is also included, comparing the computational complexity of our methods with existing approaches, showcasing their state-of-the-art efficiency. Another contribution of this paper is the release of `RCD`, a Python package that efficiently implements these algorithms. This package is

designed for practitioners and researchers interested in applying these methods in practical scenarios. The package is available at `github.com/ban-epfl/rcd`, with comprehensive documentation provided at `rcdpackage.com`.

Keywords: causal discovery, removable variable, Python package, recursive, causal structure learning, constraint-based, score-based, permutation-based

Note to Readers

This paper builds upon and extends the methodologies from four of our previous publications in causal discovery. These publications are outlined as follows:

- $\mathfrak{R}_1$ Mokhtarian et al. (2021): A recursive Markov boundary-based approach to causal structure learning; published in KDD-CD 2021; introducing **MARVEL**.
- $\mathfrak{R}_2$ Akbari et al. (2021): Recursive causal structure learning in the presence of latent variables and selection bias; published in NeurIPS 2021; introducing **L-MARVEL**.
- $\mathfrak{R}_3$ Mokhtarian et al. (2022): Learning Bayesian networks with structural side information; published in AAAI 2022; introducing **RSL**.
- $\mathfrak{R}_4$ Mokhtarian et al. (2023a): Novel ordering-based approaches for causal structure learning in the presence of unobserved variables; published in AAAI 2023; introducing **ROL**.

For readers primarily interested in the practical implementation of our proposed methods, we recommend proceeding directly to Section 8, where our Python package `RCD` is introduced. Additionally, to facilitate ease of reading, we have summarized the key notations in Table 2. The main results discussed in this paper are also summarized in Table 5.

1. Introduction

A fundamental task in various fields of science is discovering the causal relations among the variables of interest in a system. These relations have broad applications across numerous fields, including economics (Angrist et al., 1996; Heckman, 2008), genetics (Friedman et al., 2000; Schadt et al., 2003), finance (Granger, 1969; Mokhtarian et al., 2024), natural language processing (Agrawal et al., 2024; Wu et al., 2024), and social sciences (Morgan and Winship, 2015). They are often encoded as a *maximal ancestral graph* (MAG), which represents the *causal structure* of the system. In cases where no hidden variables are present, a MAG simplifies to a *directed acyclic graph* (DAG). The problem of learning the causal structure of a system, known as the causal discovery problem, is notoriously challenging and is recognized as an NP-hard problem (Chickering et al., 2004).

To address this problem, a slew of methodologies, broadly categorized into *constraint-based*, *score-based*, and *ordering-based* methods, have been developed. Constraint-based methods leverage statistical tests to identify structures consistent with the *conditional independence* (CI) relations observed in data. Score-based methods learn the graph as a solution to an optimization problem that maximizes a predefined score. Ordering-based methods focus on discovering a causal ordering of variables, simplifying structure learning by reducing the search space and computational complexity. Additionally, *hybrid* methods

Algorithm		Assumptions		Completeness	#CI tests
Reference	Name	Causal sufficiency	Other		
$\mathfrak{R}_1$	MARVEL	YES	-	YES	$\mathcal{O}(n^2 + n\Delta_{in}^2 2^{\Delta_{in}})$
$\mathfrak{R}_2$	L-MARVEL	NO	-	YES	$\mathcal{O}(n^2 + n(\Delta_{in}^+)^2 2^{\Delta_{in}^+})$
$\mathfrak{R}_3$	RSL_ω	YES	$\omega(\mathcal{G}) \leq m$	YES	$\mathcal{O}(n^2 + n\Delta_{in}^{m+1})$
	RSL_D	YES	Diamond-free	YES	$\mathcal{O}(n^2 + n\Delta_{in}^3)$
$\mathfrak{R}_4$	ROL_{HC}	NO	-	NO	$\mathcal{O}(\text{maxIter} \times n^3)$
	ROL_{VI}	NO	-	YES	$\mathcal{O}(n^2 2^n)$
	ROL_{PG}	NO	-	NO	N/A
Lower Bound		YES	-	YES	$\Omega(n^2 + n\Delta_{in} 2^{\Delta_{in}})$
Lower Bound		NO	-	YES	$\mathcal{O}(n^2 + n\Delta_{in}^+ 2^{\Delta_{in}^+})$

Table 1: Summary of the assumptions, guarantees, and complexities of the recursive causal discovery methods discussed in this paper. For a comparison with the existing literature, refer to Table 3.

have emerged that combine aspects of these methodologies. This paper primarily focuses on constraint-based methods, which are most commonly used in the presence of hidden variables. In Section 7, we present a literature review of causal discovery methods.

Causal discovery is plagued by numerous challenges, most critically those pertaining to computational/time complexity and sample efficiency. In constraint-based methods, time complexity is primarily determined by the number of required CI tests. The two classical approaches, the PC algorithm, developed for DAG-learning, and its counterpart for MAG-learning, the FCI algorithm, are not scalable to large graphs (Spirtes et al., 2000). Subsequent research has focused on reducing the computational burden and improving the statistical efficiency of these seminal works. On the other hand, methods such as the RFCI (Colombo et al., 2012), specifically designed to gain computational speed, do this at the cost of possibly compromising completeness. Completeness refers to a method's asymptotic correctness, i.e., when sufficiently large numbers of samples are available so that the statistical tests used in the method are error-free.

Recent advancements, such as our four publications ($\mathfrak{R}_1$-$\mathfrak{R}_4$) in recursive causal discovery, have made significant strides in addressing time and sample complexity while maintaining completeness guarantees. Our proposed framework strategically identifies a so-called *removable* variable (Definition 16), denoted by X, and learns its neighbors. After omitting this variable, the causal structure over the remaining variables is learned using only the samples from those variables. It is essential to carefully select X to avoid any erroneous inclusions or exclusions in the causal graph, which we shall explain in detail with examples in Section 3. As these methods operate iteratively, they systematically reduce the problem size, leading to fewer CI tests and smaller conditioning sets; hence improving the statistical reliability of these tests.

Table 1 provides a concise summary of the assumptions, guarantees, and complexity of the recursive causal discovery methods discussed in this paper[1]. In this table, causal sufficiency refers to when all variables in the system are observable. The last column shows the number of total CI tests performed as a common proxy to measure the complexity of constraint-based methods. As mentioned earlier, causal discovery from observational data is NP-hard (Chickering et al., 2004). In the last two rows, we present lower bounds for the complexity of constraint-based methods under causal sufficiency and in the absence of it, as established in Section 6.1. We shall discuss in more detail in Section 6 how this table demonstrates the state-of-the-art efficiency of our proposed methods under various assumptions.

Our main contributions in this paper are as follows.

- We present a unified framework for the algorithms proposed in $\mathfrak{R}_1$-$\mathfrak{R}_4$, refined with additional details and enhancements for a coherent presentation.
- We launch the `RCD` Python package, an efficient implementation of our recursive algorithms.
- We conduct a comprehensive literature review and compare the computational efficiency of our methods with existing approaches.

The remainder of the paper is organized as follows. We provide preliminaries, including formal definitions and the goal of causal discovery from observational data in Section 2. The theoretical foundations of our proposed algorithms are established in Section 3. In this section, we cover the generic recursive framework for causal discovery, describe the concept and characteristics of removable variables, and provide a comparison of our novel removable orders with traditional approaches. In Section 4, we introduce four recursive causal discovery methods: MARVEL, L-MARVEL, RSL, and ROL, and discuss their integration within the recursive framework. In Section 5, we delve into the implementation details of these methods and provide detailed pseudocode for each of the aforementioned algorithms. In Section 6, we discuss the complexity and completeness of various causal discovery methods, with a particular emphasis on our proposed recursive approaches, alongside lower bounds that provide theoretical limits for constraint-based methods. An extensive literature review is carried out in Section 7. Finally, Section 8 is dedicated to introducing our Python package `RCD`, which efficiently implements our proposed recursive causal discovery algorithms.

2. Preliminaries

The key notations are summarized in Table 2 to enhance clarity in our presentation. Throughout the paper, we denote random variables in capital letters, sets of variables in bold letters, and graphs in calligraphic letters (e.g., $\mathcal{G}$). Further, since the graphs are defined over a set of random variables, we use the terms variable and vertex interchangeably.

2.1 Preliminary Graph Definitions

A *mixed graph* (MG) is a graph $\mathcal{G} = \langle \mathbf{W}, \mathbf{E}_1, \mathbf{E}_2 \rangle$, where $\mathbf{W}$ is a set of vertices, $\mathbf{E}_1$ is a set of directed edges, i.e., $\mathbf{E}_1 \subseteq \{(X, Y) \mid X, Y \in \mathbf{W}\}$, and $\mathbf{E}_2$ is a set of bi-directed edges, i.e.,

1. For the notations used in Table 1, please refer to Table 2.

Notation	Description	Definition
$\mathbf{V}$, $\mathbf{U}$	Sets of observed and unobserved variables	
n	Number of observed variables, i.e., $\lvert\mathbf{V}\rvert$	
MG	Mixed graph	
MAG	Maximal ancestral graph	
DAG	Directed acyclic graph	
$Pa_{\mathcal{G}}(X)$	Parents of vertex X in MG $\mathcal{G}$	
$Pa_{\mathcal{G}}(\mathbf{X})$	$\bigcup_{X\in\mathbf{X}} Pa_{\mathcal{G}}(X)$	
$Ch_{\mathcal{G}}(X)$	Children of vertex X in MG $\mathcal{G}$	
$Ne_{\mathcal{G}}(X)$	Neighbors of vertex X in MG $\mathcal{G}$	
$Ne_{\mathbf{W}}(X)$	$\{Y \in \mathbf{W}\setminus\{X\} \lvert \forall \mathbf{Z}\subseteq \mathbf{W}\setminus\{X,Y\} : (X \not\perp\!\!\!\perp Y\lvert\mathbf{Z})_{P_{\mathbf{W}}}\}$	Definition 15
$Anc_{\mathcal{G}}(X)$	Ancestors of vertex X (including X) in MG $\mathcal{G}$	
$Anc_{\mathcal{G}}(\mathbf{X})$	$\bigcup_{X\in\mathbf{X}} Anc_{\mathcal{G}}(X)$	
$\Lambda_{\mathcal{G}}(X)$	Co-parents of vertex X in DAG $\mathcal{G}$	Definition 6
$\mathrm{Mb}_{\mathbf{W}}(X)$	Markov boundary of vertex X with respect to $\mathbf{W}$	Definition 12
$\mathrm{Mb}_{\mathbf{W}}$	$\{\mathrm{Mb}_{\mathbf{W}}(X) \lvert X\in\mathbf{W}\}$	Definition 12
$\mathrm{Mb}_{\mathcal{G}}(X)$	Markov boundary of vertex X in MG $\mathcal{G}$	Definition 13
$\mathrm{VS}_{\mathcal{G}}(X)$	V-structures in DAG $\mathcal{G}$ in which vertex X is a parent	Definition 5
$Dis_{\mathcal{G}}(X)$	District set of vertex X in MG $\mathcal{G}$	Definition 3
$Pa^{+}_{\mathcal{G}}(X)$	$Pa_{\mathcal{G}}(X)\cup Dis_{\mathcal{G}}(X)\cup Pa_{\mathcal{G}}(Dis_{\mathcal{G}}(X))$	Definition 3
$\Pi(\mathbf{W})$	Orders over $\mathbf{W}$	Definition 2
$\Pi^{c}(\mathcal{G})$	c-orders of DAG $\mathcal{G}$	Definition 23
$\Pi^{r}(\mathcal{G})$	r-orders of MAG $\mathcal{G}$	Definition 24
$\Delta(\mathcal{G})$	Maximum degree of MG $\mathcal{G}$	
$\Delta_{in}(\mathcal{G})$	Maximum in-degree of DAG $\mathcal{G}$	
$\Delta^{+}_{in}(\mathcal{G})$	Maximum size of $Pa^{+}_{\mathcal{G}}(.)$ in MG $\mathcal{G}$	Definition 3
$\omega(\mathcal{G})$	Clique number of MG $\mathcal{G}$	
$\alpha(\mathcal{G})$	Maximum Markov boundary size of MAG $\mathcal{G}$	Definition 13
$\mathcal{G}[\mathbf{W}]$	Induced subgraph of MG $\mathcal{G}$ over $\mathbf{W}$	Definition 1
$\mathcal{G}_{\mathbf{W}}$	Latent projection of MAG $\mathcal{G}$ over $\mathbf{W}$	Definition 9
$[\mathcal{G}]$	Markov equivalent MAGs of MAG $\mathcal{G}$	Definition 11
$[\mathcal{G}]^{d}$	Markov equivalent DAGs of DAG $\mathcal{G}$	Definition 11
$(X \perp\!\!\!\perp Y\lvert\mathbf{Z})_{\mathcal{G}}$	An m-separation in MG $\mathcal{G}$	Definition 4
$(X \perp\!\!\!\perp Y\lvert\mathbf{Z})_{P}$	A conditional independence (CI) in distribution P	
$\mathrm{Data}(\mathbf{W})$	A collection of i.i.d. samples from distribution $P_{\mathbf{W}}$	

Table 2: Table of notations.

$\mathbf{E}_2 \subseteq \big\{\{X, Y\} \mid X, Y \in \mathbf{W}\big\}$. All graphs in this paper are assumed to be mixed graphs. If a directed edge $X \to Y$ exists in the graph, we say X is a *parent* of Y and Y is a *child* of X. Two variables are as *neighbors* if a directed or bidirected edge exists between them. The *skeleton* of $\mathcal{G}$ is an undirected graph with the same set of vertices $\mathbf{W}$ and an edge between X and Y if they are neighbors in $\mathcal{G}$. The clique number of $\mathcal{G}$, denoted by $\omega(\mathcal{G})$, is the size of the largest clique (complete subgraph) of the skeleton of $\mathcal{G}$. Further, we denote by $\Delta(\mathcal{G})$ the maximum degree of a graph $\mathcal{G}$.

Definition 1 ($\mathcal{G}[\mathbf{W}]$) *For an MG $\mathcal{G} = \langle \mathbf{W}', \mathbf{E}'_1, \mathbf{E}'_2 \rangle$ and subset $\mathbf{W} \subseteq \mathbf{W}'$, MG $\mathcal{G}[\mathbf{W}] = \langle \mathbf{W}, \mathbf{E}_1, \mathbf{E}_2 \rangle$ denotes the induced subgraph of $\mathcal{G}$ over $\mathbf{W}$, that is $\mathbf{E}_1 = \{(X, Y) \in \mathbf{E}'_1 \mid X, Y \in \mathbf{W}\}$ and $\mathbf{E}_2 = \big\{\{X, Y\} \in \mathbf{E}'_2 \mid X, Y \in \mathbf{W}\big\}$.*

A path $\mathcal{P} = (X_1, \dots, X_k)$ is called *directed* if $X_i \to X_{i+1}$ for all $1 \leq i < k$. If a directed path exists from X to Y, we say X is an *ancestor* of Y (we assume each variable is an ancestor of itself). We denote by $Pa_{\mathcal{G}}(X)$, $Ch_{\mathcal{G}}(X)$, $Ne_{\mathcal{G}}(X)$, and $Anc_{\mathcal{G}}(X)$ the set of parents, children, neighbors, and ancestors of X in graph $\mathcal{G}$, respectively. Further, for a set of vertices $\mathbf{X}$, we define

$$Anc_{\mathcal{G}}(\mathbf{X}) \coloneqq \bigcup_{X \in \mathbf{X}} Anc_{\mathcal{G}}(X), \quad Pa_{\mathcal{G}}(\mathbf{X}) \coloneqq \bigcup_{X \in \mathbf{X}} Pa_{\mathcal{G}}(X).$$

Definition 2 ($\Pi(\mathbf{W})$) *For a set $\mathbf{W} = \{W_1, W_2, \dots, W_m\}$, an order over $\mathbf{W}$ is a tuple $(W_{i_1}, W_{i_2}, \dots, W_{i_n})$, where $\{i_1, i_2, \dots, i_m\}$ is a permutation of $\{1, 2, \dots, m\}$. We denote by $\Pi(\mathbf{W})$, the set of all orders over $\mathbf{W}$.*

Definition 3 ($Pa^+_{\mathcal{G}}(X)$) *The district set of a variable X in MG $\mathcal{G}$, denoted by $Dis_{\mathcal{G}}(X)$, is the set of variables with a path to X comprised only of bi-directed edges. By $Pa^+_{\mathcal{G}}(X)$, we denote the union of parents, district set, and parents of district set. That is,*

$$Pa^+_{\mathcal{G}}(X) \coloneqq Pa_{\mathcal{G}}(X) \cup Dis_{\mathcal{G}}(X) \cup Pa_{\mathcal{G}}(Dis_{\mathcal{G}}(X)).$$

Furthermore, by $\Delta^+_{in}(\mathcal{G})$, we denote the maximum size of $Pa^+_{\mathcal{G}}(.)$ in $\mathcal{G}$.

A non-endpoint vertex X_i on a path $(X_1, X_2, \cdots, X_k)$ is called a *collider* if one of the following situations arises.

$$X_{i-1} \to X_i \leftarrow X_{i+1}, \quad X_{i-1} \leftrightarrow X_i \leftarrow X_{i+1},$$
$$X_{i-1} \to X_i \leftrightarrow X_{i+1}, \quad X_{i-1} \leftrightarrow X_i \leftrightarrow X_{i+1}.$$

When X_{i-1} and X_{i+1} are not neighbors, and X_i is a collider, the arrangement is termed as an unshielded collider. A path $\mathcal{P}$ is a *collider path* if every non-endpoint vertex on $\mathcal{P}$ is a collider on $\mathcal{P}$.

Definition 4 (m-separation) *In an MG $\mathcal{G}$ over $\mathbf{W}$, a path $\mathcal{P} = (X, W_1, \cdots, W_k, Y)$ between two distinct variables X and Y is said to be blocked by a set $\mathbf{Z} \subseteq \mathbf{W} \setminus \{X, Y\}$ in $\mathcal{G}$ if there exists $1 \leq i \leq k$ such that (i) W_i is a collider on $\mathcal{P}$ and $W_i \notin Anc_{\mathcal{G}}(\mathbf{Z} \cup \{X, Y\})$, or (ii) W_i is not a collider on $\mathcal{P}$ and $W_i \in \mathbf{Z}$. We say $\mathbf{Z}$ m-separates X and Y in $\mathcal{G}$ and denote it by $(X \perp\!\!\!\perp Y | \mathbf{Z})_{\mathcal{G}}$ if all the paths in $\mathcal{G}$ between X and Y are blocked by $\mathbf{Z}$.*

A *directed cycle* exists in an MG $\mathcal{G} = \langle \mathbf{W}, \mathbf{E}_1, \mathbf{E}_2 \rangle$ when there exists $X, Y \in \mathbf{W}$ such that $(X, Y) \in \mathbf{E}_1$ and $Y \in Anc_{\mathcal{G}}(X)$. Similarly, an *almost directed cycle* exists in $\mathcal{G}$ when there exists $X, Y \in \mathbf{W}$ such that $\{X, Y\} \in \mathbf{E}_2$ and $Y \in Anc_{\mathcal{G}}(X)$. An MG with no directed or almost-directed cycles is said to be *ancestral*. An ancestral MG is called *maximal* if every pair of non-neighbor vertices are m-separable, i.e., there exists a set of vertices that m-separates them. An MG is called a *maximal ancestral graph* (MAG) if it is both ancestral and maximal. A MAG with no bidirected edges is called a *directed acyclic graph (DAG)*. When $\mathcal{G}$ is a DAG, the definition of m-separation reduces to the so-called d-separation. Moreover, unshielded colliders in a DAG reduce to *v-structures*. For a DAG $\mathcal{G}$, we denote its maximum in-degree by $\Delta_{in}(\mathcal{G})$.

Definition 5 ($\mathbf{VS}_{\mathcal{G}}(X)$) *In a DAG $\mathcal{G}$, we denote by $VS_{\mathcal{G}}(X)$ the set of v-structures in which vertex X is one of the two parents.*

Definition 6 (Co-parent) *In a DAG, two non-neighbor vertices are co-parents if they have at least one common child. The set of co-parents of vertex X in DAG $\mathcal{G}$ is denoted by $\Lambda_{\mathcal{G}}(X)$.*

Definition 7 (Discriminating path) *In a MAG $\mathcal{G}$, a path $\mathcal{P} = (X, V_1, \ldots, V_k, Y)$, where $k \geq 1$, is a discriminating path for V_k if (i) X and Y are not neighbors, (ii) $\{V_1, \ldots V_{k-1}\} \subseteq Pa_{\mathcal{G}}(X)$, and (iii) $\{V_1, \ldots V_{k-1}\}$ are colliders on $\mathcal{P}$.*

Definition 8 (Inducing path) *Suppose $\mathcal{G}$ is a MAG over $\mathbf{W}_1 \sqcup \mathbf{W}_2$[2] and let X, Y be distinct vertices in $\mathbf{W}_1$. An inducing path between X and Y relative to $\mathbf{W}_2$ is a path on which (i) every non-collider is a member of $\mathbf{W}_2$, and (ii) every collider belongs to $Anc_{\mathcal{G}}(X, Y)$.*

We note that in Definition 8, no subset of $\mathbf{W}_1$ can block an inducing path relative to $\mathbf{W}_2$.

Definition 9 (Latent projection) *Suppose $\mathcal{G}$ is a MAG over $\mathbf{W}_1 \sqcup \mathbf{W}_2$. The latent projection of $\mathcal{G}$ over $\mathbf{W}_1$, denoted by $\mathcal{G}_{\mathbf{W}_1}$, is a MAG over $\mathbf{W}_1$ constructed as follows:*

(i) Skeleton: $X, Y \in \mathbf{W}_1$ are neighbors in $\mathcal{G}_{\mathbf{W}_1}$ if there exists an inducing path in $\mathcal{G}$ between X and Y relative to $\mathbf{W}_2$.

(ii) Orientation: For each pair of neighbors X, Y in $\mathcal{G}_{\mathbf{W}_1}$, the edge between X and Y is oriented as $X \to Y$ if $X \in Anc_{\mathcal{G}}(Y)$ and $Y \notin Anc_{\mathcal{G}}(X)$ and as $X \leftrightarrow Y$ if $X \notin Anc_{\mathcal{G}}(Y)$ and $Y \notin Anc_{\mathcal{G}}(X)$.

Remark 10 *The latent projection maintains the ancestral relationships. Furthermore, in a MAG $\mathcal{G}$ over $\mathbf{W}$, for any $\mathbf{W}_2 \subseteq \mathbf{W}_1 \subseteq \mathbf{W}$ we have $(\mathcal{G}_{\mathbf{W}_1})_{\mathbf{W}_2} = \mathcal{G}_{\mathbf{W}_2}$.*

Richardson et al. (2002) showed that the latent projection in Definition 9 is the unique projection of a MAG $\mathcal{G} = \langle \mathbf{W}_1 \sqcup \mathbf{W}_2, \mathbf{E}_1, \mathbf{E}_2 \rangle$ over $\mathbf{W}_1$ that satisfies the following: for any distinct variables X, Y in $\mathbf{W}_1$ and $\mathbf{Z} \subseteq \mathbf{W}_1 \setminus \{X, Y\}$,

$$(X \perp\!\!\!\perp Y | \mathbf{Z})_{\mathcal{G}_{\mathbf{W}_1}} \iff (X \perp\!\!\!\perp Y | \mathbf{Z})_{\mathcal{G}} \,. \tag{1}$$

2. We $\sqcup$ to denote disjoint union.

2.2 Generative Model: Structural Equation Model (SEM)

Consider a DAG $\mathcal{G}$ over $\mathbf{V} \sqcup \mathbf{U}$, where $\mathbf{V}$ and $\mathbf{U}$ denote sets of observed and unobserved variables, respectively. In a *structural equation model* (SEM) with *causal graph* $\mathcal{G}$, each variable $X \in \mathbf{V} \cup \mathbf{U}$ is generated as $X = f_X(Pa_{\mathcal{G}}(X), \epsilon_X)$, where f_X is a deterministic function and ϵ_X is the exogenous variable corresponding to X with an additional assumption that the exogenous variables are jointly independent (Pearl, 2009). Such a SEM induces a unique joint distribution $P_{\mathbf{V}\cup\mathbf{U}}$ over all the variables, which satisfies Markov factorization property with respect to $\mathcal{G}$. That is,

$$P_{\mathbf{V}\cup\mathbf{U}}(\mathbf{V}, \mathbf{U}) = \prod_{X \in \mathbf{V}\cup\mathbf{U}} P_{\mathbf{V}\cup\mathbf{U}}(X | Pa_{\mathcal{G}}(X)).$$

The marginalized distribution $P_{\mathbf{V}} := \sum_{\mathbf{U}} P_{\mathbf{V}\cup\mathbf{U}}$ is the *observational distribution* of the underlying SEM. Furthermore, MAG $\mathcal{G}_{\mathbf{V}}$ (i.e., the latent projection of DAG $\mathcal{G}$ over $\mathbf{V}$ as introduced in Definition 9) is the causal MAG over the observed variables. If all the variables in the SEM are observable, i.e., when $\mathbf{U} = \varnothing$, $\mathcal{G}_{\mathbf{V}}$ coincides with $\mathcal{G}$. This assumption is commonly referred to as *causal sufficiency*. Therefore, the causal graph over the observed variables is a DAG when causal sufficiency holds.

Next, we show that under suitable assumptions, causal MAG $\mathcal{G}_{\mathbf{V}}$ captures the conditional independencies of $P_{\mathbf{V}}$.

2.3 Markov Property and Faithfulness

Let P be the joint distribution of a set of variables $\mathbf{W}$. For distinct variables $X, Y \in \mathbf{W}$ and $\mathbf{Z} \subseteq \mathbf{W} \setminus \{X, Y\}$, a *conditional independence (CI)* test in P on the triplet $(X, Y, \mathbf{Z})$ yields independence, denoted by $(X \perp\!\!\!\perp Y | \mathbf{Z})_P$, if $P(X|Y, \mathbf{Z}) = P(X|\mathbf{Z})$. Distribution P satisfies *Markov property* with respect to MG $\mathcal{G}$ if m-separations in $\mathcal{G}$ imply CIs in P. That is,

$$(X \perp\!\!\!\perp Y | \mathbf{Z})_{\mathcal{G}} \Longrightarrow (X \perp\!\!\!\perp Y | \mathbf{Z})_P \,.$$

Conversely, distribution P satisfies *faithfulness* with respect to MG $\mathcal{G}$ if CIs in P imply m-separations in $\mathcal{G}$. That is,

$$(X \perp\!\!\!\perp Y | \mathbf{Z})_P \Longrightarrow (X \perp\!\!\!\perp Y | \mathbf{Z})_{\mathcal{G}} \,.$$

Consider a SEM over $\mathbf{V} \sqcup \mathbf{U}$ with causal DAG $\mathcal{G}$ and joint distribution $P_{\mathbf{V}\cup\mathbf{U}}$. The joint distribution $P_{\mathbf{V}\cup\mathbf{U}}$ satisfies Markov property with respect to DAG $\mathcal{G}$ (Pearl, 2000). Although $P_{\mathbf{V}\cup\mathbf{U}}$ does not necessarily satisfy faithfulness with respect to $\mathcal{G}$, it is a common assumption in the literature. When faithfulness holds, we have

$$(X \perp\!\!\!\perp Y | \mathbf{Z})_{P_{\mathbf{V}\cup\mathbf{U}}} \iff (X \perp\!\!\!\perp Y | \mathbf{Z})_{\mathcal{G}} \,, \tag{2}$$

for any distinct variables X, Y in $\mathbf{V} \cup \mathbf{U}$ and $\mathbf{Z} \subseteq \mathbf{V} \cup \mathbf{U} \setminus \{X, Y\}$.

Recall that $\mathbf{U}$ is the set of unobserved variables. Causal discovery aims to find a graphical model over $\mathbf{V}$ that encodes the CI relations in $P_{\mathbf{V}}$. Note that the induced subgraph $\mathcal{G}[\mathbf{V}]$ does not necessarily satisfy Markov property and faithfulness with respect to $P_{\mathbf{V}}$, failing to encode the CIs of $P_{\mathbf{V}}$. On the other hand, Equations (1) and (2) imply that the latent

projection of $\mathcal{G}$ over $\mathbf{V}$, i.e., $\mathcal{G}_{\mathbf{V}}$, satisfies Markov property and faithfulness with respect to $P_{\mathbf{V}}$. That is, for distinct variables X, Y in $\mathbf{V}$ and $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$,

$$(X \perp\!\!\!\perp Y|\mathbf{Z})_{P_{\mathbf{V}}} \iff (X \perp\!\!\!\perp Y|\mathbf{Z})_{\mathcal{G}_{\mathbf{V}}}. \tag{3}$$

Accordingly, causal discovery aims to learn $\mathcal{G}_{\mathbf{V}}$. However, as we discuss next, using the observational distribution $P_{\mathbf{V}}$, MAG $\mathcal{G}_{\mathbf{V}}$ can only be learned up to an equivalency class.

2.4 Causal Discovery from Observational Distribution

By performing CI tests in the observational distribution $P_{\mathbf{V}}$, the m-separations of causal MAG $\mathcal{G}_{\mathbf{V}}$ can be recovered using Equation (3). However, two MAGs might have the same set of m-separations.

Definition 11 (MEC) *Two MAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are* Markov equivalent *if they impose the same set of m-separations, i.e., $(X \perp\!\!\!\perp Y|\mathbf{Z})_{\mathcal{G}_1} \iff (X \perp\!\!\!\perp Y|\mathbf{Z})_{\mathcal{G}_2}$. We denote by $[\mathcal{G}]$ the set of Markov equivalent MAGs of $\mathcal{G}$, known as the Markov equivalence class (MEC). Moreover, if $\mathcal{G}$ is a DAG, we denote by $[\mathcal{G}]^d$ the set of Markov equivalent DAGs of $\mathcal{G}$.*

CIs are statistically sufficient for causal discovery in non-parametric models[3]. That is, if multiple MAGs satisfy Markov property and faithfulness with respect to the observational distribution $P_{\mathbf{V}}$, we cannot identify which one is the causal MAG of the underlying SEM. Nevertheless, without the assumption of causal sufficiency, causal discovery from observational distribution aims to identify $[\mathcal{G}_{\mathbf{V}}]$. When causal sufficiency holds, i.e., $\mathbf{U} = \varnothing$ and $\mathcal{G}_{\mathbf{V}} = \mathcal{G}$, the goal is to identify $[\mathcal{G}]^d$.

2.5 Markov Boundary

The majority of the approaches that we present in this paper employ the concept of Markov boundary. This notion can be defined based on either a distribution or a graph.

Definition 12 ($\mathbf{Mb}_{\mathbf{W}}(X)$) *Suppose P is the joint distribution of a set of random variables $\mathbf{W}$ and let $X \in \mathbf{W}$. Markov boundary of X with respect to $\mathbf{W}$, denoted by $Mb_{\mathbf{W}}(X)$, is a minimal subset of $\mathbf{W} \setminus \{X\}$, such that*

$$(X \perp\!\!\!\perp \mathbf{W} \setminus (Mb_{\mathbf{W}}(X) \cup \{X\})|Mb_{\mathbf{W}}(X))_P.$$

Additionally, we define $Mb_{\mathbf{W}}$ as the set of Markov boundaries of all variables in $\mathbf{W}$ with respect to $\mathbf{W}$, i.e., $Mb_{\mathbf{W}} \coloneqq \{Mb_{\mathbf{W}}(X)|X \in \mathbf{W}\}$.

Definition 13 ($\mathbf{Mb}_{\mathcal{G}}(X)$) *In an MG $\mathcal{G}$, the Markov boundary of a variable X, denoted by $Mb_{\mathcal{G}}(X)$, consists of all the variables that have a collider path to X. We denote by $\alpha(\mathcal{G})$ the maximum Markov boundary size of the variables in $\mathcal{G}$.*

3. Side information about the underlying SEM can render the causal graph uniquely identifiable. For instance, when the functions are linear and the exogenous noises are non-Gaussian, CIs are no longer statistically sufficient, and the causal MAG is uniquely identifiable from observational distribution.

Remark 14 *If $\mathcal{G}$ is a DAG, then*

$$Mb_{\mathcal{G}}(X) = Pa_{\mathcal{G}}(X) \cup Ch_{\mathcal{G}}(X) \cup \Lambda_{\mathcal{G}}(X) = Ne_{G}(X) \cup \Lambda_{\mathcal{G}}(X).$$

Consider a MAG $\mathcal{G}$ and a distribution P, both over a set $\mathbf{W}$. Pellet and Elisseeff (2008a) and Yu et al. (2018) showed that if $\mathcal{G}$ satisfies Markov property and faithfulness with respect to P, then for each variable $X \in \mathbf{W}$, $\mathrm{Mb}_{\mathbf{W}}(X)$ is unique and is equal to $\mathrm{Mb}_{\mathcal{G}}(X)$.

3. Theoretical Foundation of Proposed Algorithms

In this section, we establish the theoretical basis for our proposed algorithms in causal discovery. Firstly, we present a generic recursive framework for causal discovery in Section 3.1. Then, we explore the concept and characteristics of removable variables in Section 3.2. Finally, we compare and contrast the newly proposed removable orders with traditional approaches in Section 3.3.

3.1 A Recursive Framework for Causal Discovery

Herein, we present a generic recursive framework for causal discovery that does not necessarily assume causal sufficiency. Consider a SEM over $\mathbf{V} \sqcup \mathbf{U}$ with causal DAG $\mathcal{G}$, where $\mathbf{V}$ and $\mathbf{U}$ denote the set of observed and unobserved variables, respectively. As discussed in Section 2.4, our goal is to learn $[\mathcal{G}_{\mathbf{V}}]$ (or $[\mathcal{G}]^d$ with the assumption of causal sufficiency) from $P_{\mathbf{V}}$ when Equation (3) holds. In the following sections, we mainly focus on recovering the skeleton of $\mathcal{G}_{\mathbf{V}}$. Later in Section 5.7, we show that a slight variation of the discussed methods can be employed to recover the MEC.

Definition 15 ($Ne_{\mathbf{W}}(X)$) *Suppose $P_{\mathbf{W}}$ is the joint distribution of variables in a set $\mathbf{W}$. For $X \in \mathbf{W}$, we denote by $Ne_{\mathbf{W}}(X)$ the set of variables $Y \in \mathbf{W} \setminus \{X\}$ such that for each $\mathbf{Z} \subseteq \mathbf{W} \setminus \{X, Y\}$, we have $(X \not\perp\!\!\!\perp Y | \mathbf{Z})_{P_{\mathbf{W}}}$.*

Since non-neighbors in any MAG are m-separable, for any $\mathbf{W} \subseteq \mathbf{V}$ we have

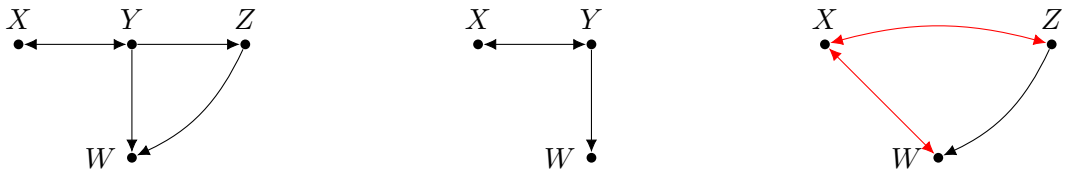
$$Ne_{\mathbf{W}}(X) = Ne_{\mathcal{G}_{\mathbf{W}}}(X). \tag{4}$$

Hence, to learn the skeleton of $\mathcal{G}_{\mathbf{V}}$, it suffices to learn $Ne_{\mathbf{V}}(X)$ for each $X \in \mathbf{V}$. In practice, a finite set of samples from $P_{\mathbf{V}}$ is available instead of knowing the exact probability distribution. Let Data($\mathbf{V}$) denote a collection of i.i.d. samples from $P_{\mathbf{V}}$, sufficiently large to recover the CI relations in $P_{\mathbf{V}}$. For any subset $\mathbf{W} \subseteq \mathbf{V}$, Data($\mathbf{W}$) represents the samples corresponding to the variables in $\mathbf{W}$.

A generic recursive framework for causal discovery is presented in Algorithm 1. The algorithm iteratively removes variables from $\mathbf{V}$ and learns the skeleton over the remaining variables. At the i-th iteration, a variable $X_i \in \mathbf{V}_i$ is selected, where $\mathbf{V}_i$ denotes the set of remaining variables. As we shall discuss shortly, this variable cannot be arbitrary and must be *removable*. After finding such a variable X_i, the algorithm learns the set of neighbors of X_i in $\mathcal{G}_{\mathbf{V}_i}$ using Data($\mathbf{V}_i$). Note that this is possible because of Equation (4). Then, the samples corresponding to X_i are discarded, and the causal discovery problem is solved recursively for the remaining variables $\mathbf{V}_{i+1} = \mathbf{V}_i \setminus \{X_i\}$.

Algorithm 1: Recursive framework for learning the skeleton of $\mathcal{G}_{\mathbf{V}}$

1: **Input:** Data($\mathbf{V}$)
2: $\mathbf{V}_1 \leftarrow \mathbf{V}$
3: **for** i from 1 to $n-1$ **do**
4: $\quad X_i \leftarrow$ Find a [*removable*] variable in $\mathcal{G}_{\mathbf{V}_i}$ using Data($\mathbf{V}_i$)
5: $\quad$ Learn $Ne_{\mathbf{V}_i}(X_i)$ using Data($\mathbf{V}_i$)
6: $\quad \mathbf{V}_{i+1} \leftarrow \mathbf{V}_i \setminus \{X_i\}$



(a) $\mathcal{G}_{\{X,Y,Z,W\}}$ (b) $\mathcal{G}_{\{X,Y,W\}} = \mathcal{G}_{\{X,Y,Z,W\}}[X,Y,W]$ (c) $\mathcal{G}_{\{X,Z,W\}}$

Figure 1: Graphs in Example 1.

For Algorithm 1 to correctly learn the skeleton of $\mathcal{G}_{\mathbf{V}}$, it is necessary that at each iteration, $Ne_{\mathbf{V}_i}(X_i) = Ne_{\mathcal{G}_{\mathbf{V}}[\mathbf{V}_i]}(X_i)$. Recall that $Ne_{\mathbf{V}_i}(X_i) = Ne_{\mathcal{G}_{\mathbf{V}_i}}(X_i)$ and $\mathcal{G}_{\mathbf{V}}[\mathbf{V}_i]$ denotes the induced subgraph of $\mathcal{G}_{\mathbf{V}}$ over $\mathbf{V}_i$. Since the latent projection can only add new edges to the projected graph, this condition is equivalent to

$$\mathcal{G}_{\mathbf{V}_i} = \mathcal{G}_{\mathbf{V}}[\mathbf{V}_i], \quad \forall 1 \leq i < n. \tag{5}$$

However, selecting an arbitrary variable X_i in line 4 may not uphold this property.

Example 1 *Consider MAG $\mathcal{G}_{\{X,Y,Z,W\}}$ shown in Figure 1a. In Figure 1b, MAG $\mathcal{G}_{\{X,Y,W\}}$ is the same as $\mathcal{G}_{\{X,Y,Z,W\}}[X,Y,W]$. However, in Figure 1c, MAG $\mathcal{G}_{\{X,Z,W\}}$ has two extra edges between X,Z and X,W, which are not present in $\mathcal{G}_{\{X,Y,Z,W\}}[X,Z,W]$. This demonstrates that when $\mathbf{V}_i = \{X,Y,Z,W\}$ in Algorithm 1, Z can be selected in line 4, whereas Y cannot.*

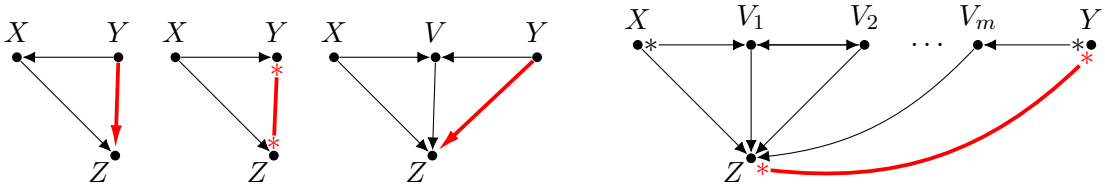
To employ Algorithm 1, we need to provide a method to find a removable variable and learn its neighbors at each iteration. In the next section, we will define the concept of removable variables and show that Equation (5) holds *if and only if* a removable variable is selected at each iteration of this recursive framework.

3.2 Removable Variables

In this section, we define removable variables, present a graphical characterization for them under different assumptions, and provide certain crucial properties.

Definition 16 (Removable variable) *In a MAG $\mathcal{G}$ over $\mathbf{W}$, a vertex $X \in \mathbf{W}$ is called removable if $\mathcal{G}$ and $\mathcal{G}[\mathbf{W} \setminus \{X\}]$ impose the same set of m-separations over $\mathbf{W} \setminus \{X\}$. That is, for any distinct vertices $Y,T \in \mathbf{W} \setminus \{X\}$ and $\mathbf{Z} \subseteq \mathbf{W} \setminus \{X,Y,T\}$,*

$$(Y \perp\!\!\!\perp T|\mathbf{Z})_{\mathcal{G}} \iff (Y \perp\!\!\!\perp T|\mathbf{Z})_{\mathcal{G}[\mathbf{W}\setminus\{X\}]}.$$



(a) For DAGs (b) For MAGs

Figure 2: Graphical criterion of removability. Asterisk (∗) is used as a wildcard, which indicates that the edge endpoint can be either an arrowhead or a tail. In the case of MAGs, the path $(X, V_1, ..., Y)$ is a collider path and $X, ..., V_m \in Pa_{\mathcal{G}}(Z)$. X is removable if and only if for all such paths, Y and Z are adjacent. In the case of a DAG, it suffices to check this condition for vertices Y, where Y is a parent, child, or co-parent of X.

Proposition 17 (Only removables can get removed) *Suppose $\mathcal{G}$ is a MAG over $\mathbf{W}$ and $X \in \mathbf{W}$. MAG $\mathcal{G}_{\mathbf{W}\setminus\{X\}}$ is equal to $\mathcal{G}[\mathbf{W} \setminus \{X\}]$ if and only if X is removable in $\mathcal{G}$.*

Proposition 17 implies that Equation (5) holds for Algorithm 1 if and only if X_i is removable in $\mathcal{G}_{\mathbf{V}_i}$ at each iteration. Next, we provide graphical characterizations of removable variables in both MAGs and DAGs, along with their key properties. We then define removable orders and discuss their properties.

3.2.1 Graphical Characterization of Removable Variables

We present graphical characterizations of removable variables in DAGs and MAGs.

Theorem 18 (Graphical characterization in DAGs) *A variable X is removable in a DAG $\mathcal{G}$ if and only if the following two conditions are satisfied for every $Z \in Ch_{\mathcal{G}}(X)$.*

Condition 1: $Ne_{\mathcal{G}}(X) \subset Ne_{\mathcal{G}}(Z) \cup \{Z\}$.

Condition 2: $Pa_{\mathcal{G}}(V) \subset Pa_{\mathcal{G}}(Z), \quad \forall V \in Ch_{\mathcal{G}}(X) \cap Pa_{\mathcal{G}}(Z)$.

Figure 2a depicts the two conditions of Theorem 18 for removable variables in DAGs.

Theorem 19 (Graphical characterization in MAGs) *Let $\mathcal{G}$ be a MAG over $\mathbf{W}$. Vertex X is removable in $\mathcal{G}$ if and only if for any collider path $u = (X, V_1, ..., V_m, Y)$ and $Z \in \mathbf{W} \setminus \{X, Y, V_1, ..., V_m\}$ such that $\{X, V_1, ..., V_m\} \subseteq Pa_{\mathcal{G}}(Z)$, Y and Z are neighbors.*

Figure 2b depicts the condition of Theorem 19 for removable variables in MAGs. Path $u = (X, V_1, ..., V_m, Y)$ is a collider path where $\{X, V_1, ..., V_m\} \subseteq Pa_{\mathcal{G}}(Z)$. Theorem 19 states that X is removable if and only if, for any such path, Y and Z are neighbors. In the case of a DAG, the collider paths are of size either 1 (parents and children of X) or 2 (co-parents of X). As shown in Figure 2a, the graphical criterion of Theorem 19 reduces to checking the adjacency of these three groups of vertices with each child of X (see Theorem 18).

3.2.2 Properties of Removable Variables

Herein, we discuss some key properties of removable variables.

Proposition 20 (Removables exist) *In any MAG, the variables with no children are removable, and the set of vertices with no children is non-empty. Therefore, any MAG has at least one removable variable.*

Proposition 20 implies that X_i in line 4 of Algorithm 1 is well-defined.

Proposition 21 (Removables have small Mb size) *In a MAG $\mathcal{G}$, if a vertex X is removable, then $|Mb_{\mathcal{G}}(X)| \leq \Delta^{+}_{in}(\mathcal{G})$. Furthermore, if $\mathcal{G}$ is a DAG, then $|Mb_{\mathcal{G}}(X)| \leq \Delta_{in}(\mathcal{G})$.*

Note that for an arbitrary variable X, $|\mathrm{Mb}_{\mathcal{G}}(X)|$ can be as large as n whereas $\Delta^{+}_{in}(\mathcal{G})$ (or $\Delta_{in}(\mathcal{G})$ for DAGs) is typically a small number, demonstrating that removable variables have relatively small Markov boundary sizes.

Proposition 22 (Removables are invariant in a MEC) *Two Markov equivalent MAGs have the same set of removable variables.*

We note that the set of vertices without children is not the same for all MAGs in a MEC. However, Proposition 22 implies that the set of removable variables is a superset of the vertices without children, which is invariant across all MAGs in a MEC.

3.3 Removable Orders

In Algorithm 1, our generic framework can be viewed as an ordering-based approach. In this approach, a variable is eliminated at each iteration based on a specific order. In this section, we first define c-orders, which are standard orders that existing ordering-based methods use to recover the graph. We then introduce r-order, which are orders that can be integrated into Algorithm 1. Finally, we show that r-orders are advantageous over c-orders for structure learning, because of the properties that are depicted in Figure 3.

Score-based methods are one of the main classes of algorithms for causal discovery. These algorithms use a score function, such as a regularized likelihood function or Bayesian information criterion (BIC), to evaluate graphs and determine the structure that maximizes the score. Under the causal sufficiency assumption, the search space of the majority of these methods is the space of DAGs, which contains $2^{\Omega(n^2)}$ members. The first ordering-based search strategy was introduced by Teyssier and Koller (2005). These methods search through the space of orders (Definition 2), which includes $2^{\mathcal{O}(n\log(n))}$ members. It is worth noting that the space of orders is much smaller than the space of DAGs. Ordering-based methods divide the learning task into two stages. In the first stage, they use the available data to find a causal order (c-order), which is defined as follows.

Definition 23 (c-order) *In a DAG $\mathcal{G} = \langle \mathbf{W}, \mathbf{E}_1, \varnothing \rangle$, an order $\pi = (X_1, \ldots, X_m) \in \Pi(\mathbf{W})$ is called causal (in short c-order) if $i > j$ for each $(X_i, X_j) \in \mathbf{E}_1$. Equivalently, π is c-order if X_i has no children in $\mathcal{G}[\{X_i, X_{i+1}, \ldots, X_n\}]$ for each $1 \leq i \leq n$. We denote by $\Pi^c(\mathcal{G})$ the set of c-orders of $\mathcal{G}$.*

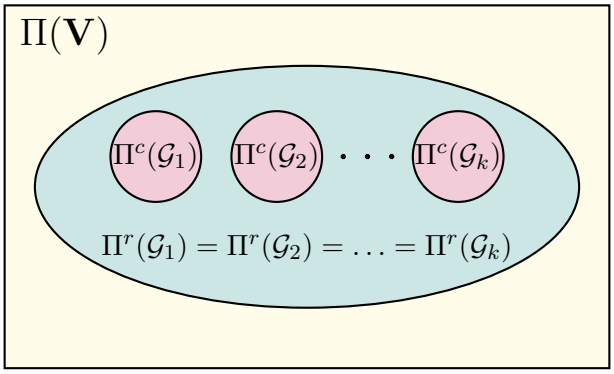


Figure 3: In this figure, $\{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$ denotes a set of Markov equivalent DAGs. $\Pi(\mathbf{V})$ denotes the set of orders over $\mathbf{V}$, which is the search space of ordering-based methods. $\Pi^c(\mathcal{G}_i)$ denotes the set of c-orders of $\mathcal{G}_i$, the target space of existing ordering-based methods in the literature. $\Pi^r(\mathcal{G}_i)$ denotes the set of r-orders of $\mathcal{G}_i$, which is the target space of ROL.

We note that c-orders are defined over DAGs, and accordingly, most of the ordering-based approaches for causal discovery require causal sufficiency. In the second stage, ordering-based methods use the learned order to identify the MEC of $\mathcal{G}$.

We introduce a novel type of order for MAGs, called removable order (in short, r-order), and argue that r-orders are advantageous over c-orders for structure learning.

Definition 24 (r-order) *In a MAG $\mathcal{G}$, an order $\pi = (X_1, \cdots, X_n)$ is called removable (r-order) if X_i is a removable variable in $\mathcal{G}[\{X_i, , X_{i+1}, \ldots, X_n\}]$ for each $1 \leq i \leq n$. We denote by $\Pi^r(\mathcal{G})$ the set of r-orders of $\mathcal{G}$.*

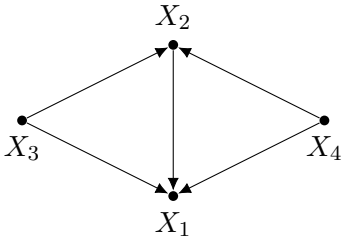
Note that r-orders are defined over MAGs, which enables us to design ordering-based methods that do not assume causal sufficiency.

Example 2 *Consider the two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ and their sets of c-orders depicted in Figure 4. In this case, $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent and together form a MEC. Furthermore, $\Pi^c(\mathcal{G}_1)$ and $\Pi^c(\mathcal{G}_2)$ are disjoint, and each contains 2 c-orders, and any order over the set of vertices is an r-order for both $\mathcal{G}_1$ and $\mathcal{G}_2$. Hence, each graph has 24 r-orders.*

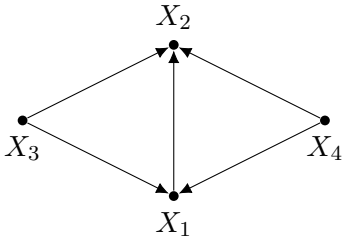
In a MEC, all MAGs share the same r-orders. In DAGs, r-orders include all c-orders as subsets. This is illustrated in Figure 3 and formalized in the subsequent propositions.

Proposition 25 (r-orders are invariant across MEC) *If $\mathcal{G}_1$ and $\mathcal{G}_2$ are two Markov equivalent MAGs, then $\Pi^r(\mathcal{G}_1) = \Pi^r(\mathcal{G}_2)$.*

Proposition 26 (r-orders include c-orders) *For any DAG $\mathcal{G}$, we have $\Pi^c(\mathcal{G}) \subseteq \Pi^r(\mathcal{G})$.*



(a) DAG $\mathcal{G}_1$
$\Pi^c(\mathcal{G}_1) = \{(X_1, X_2, X_3, X_4), (X_1, X_2, X_4, X_3)\}$



(b) DAG $\mathcal{G}_2$
$\Pi^c(\mathcal{G}_2) = \{(X_2, X_1, X_3, X_4), (X_2, X_1, X_4, X_3)\}$

Figure 4: Two Markov equivalent DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ that form a MEC together and their disjoint sets of c-orders. In this example, any order over $\mathbf{V} = \{X_1, X_2, X_3, X_4\}$ is an r-order, i.e., $\Pi^r(\mathcal{G}_1) = \Pi^r(\mathcal{G}_2) = \Pi(\mathbf{V})$. Note that $|\Pi^r(\mathcal{G}_1)| = |\Pi^r(\mathcal{G}_2)| = 24 > 2 = |\Pi^c(\mathcal{G}_1)| = |\Pi^c(\mathcal{G}_2)|$.

4. RCD Methods: MARVEL, L-MARVEL, RSL, ROL

In the previous section, we introduced a recursive framework for causal discovery. We defined removable variables and provided graphical characterizations of them in MAGs and DAGs, along with their important properties. Building on that framework, in this section, we present four recursive causal discovery methods: MARVEL, L-MARVEL, RSL, and ROL. MARVEL, L-MARVEL, and RSL provide various approaches for lines 4–5 of Algorithm 1 under different sets of assumptions. ROL, however, identifies all removable variables at once using r-orders.

4.1 Finding Removable Variables and Their Neighbors

Motivated by Proposition 21, we provide Algorithm 2, a framework for finding a removable variable.

Algorithm 2: Finding a removable variable.

1: **FindRemovable**(Data($\mathbf{W}$), $\text{Mb}_{\mathbf{W}}$)
2: $m \leftarrow |\mathbf{W}|$
3: $(W_1, \ldots, W_m) \leftarrow$ Sort $\mathbf{W}$ such that $|\text{Mb}_{\mathbf{W}}(W_1)| \leq |\text{Mb}_{\mathbf{W}}(W_2)| \leq \cdots \leq |\text{Mb}_{\mathbf{W}}(W_m)|$
4: **for** $i = 1$ to m **do**
5: **if IsRemovable**(W_i, Data($\text{Mb}_{\mathbf{W}}(W_i)$)) is True **then**
6: **return** W_i

Given a set $\mathbf{W}$, the algorithm takes the Markov boundaries as input, sorts the variables based on the size of their Markov boundaries, and applies the function *IsRemovable* to them. The function *IsRemovable* determines whether a variable W_i is removable in $\mathcal{G}_{\mathbf{W}}$. Algorithm 2 returns the first removable variable it identifies and halts. Therefore, Proposition 21 implies that *IsRemovable* will only be called for variables with Markov boundary sizes less than or equal to $\Delta^+_{in}(\mathcal{G})$. As we showed in Theorems 18 and 19, the removability of a

variable is a property of the causal graph over its Markov boundary. Therefore, we only need to utilize data from the variables in $\mathrm{Mb}_{\mathbf{W}}(W_i)$ for the *IsRemovable* function.

Next, we introduce three approaches for efficiently testing the removability of a variable under different sets of assumptions.

4.2 MARVEL: MArkov boundary-based Recursive Variable ELimination

The first recursive approach that introduced and utilized the notion of removability is MARVEL (Mokhtarian et al., 2021). MARVEL assumes causal sufficiency, i.e., the causal graph is a DAG.

Suppose $\mathcal{G}$ is a DAG with the set of vertices $\mathbf{V}$. To verify the removability of a variable $X \in \mathbf{V}$, MARVEL first learns the neighbors of X, i.e., $Ne_{\mathcal{G}}(X)$, and the set of v-structures in which X is a parent, i.e., $\mathrm{VS}_{\mathcal{G}}(X)$, using the following two lemmas.

Lemma 27 (Pellet and Elisseeff, 2008a) *Suppose $\mathcal{G}$ is a MAG over $\mathbf{V}$. For $X \in \mathbf{V}$ and $Y \in Mb_{\mathcal{G}}(X)$, Y is a neighbor of X if and only if*

$$(X \not\perp\!\!\!\perp Y|\mathbf{S})_{\mathcal{G}}\,, \quad \forall \mathbf{S} \subsetneq Mb_{\mathcal{G}}(X) \setminus \{Y\}. \tag{6}$$

We note that while MARVEL uses Lemma 27 when $\mathcal{G}$ is a DAG, Pellet and Elisseeff (2008a) showed that it also holds for MAGs.

Lemma 28 *Suppose $\mathcal{G}$ is a DAG, $Y \in \Lambda_{\mathcal{G}}(X)$, $(X \perp\!\!\!\perp Y|\mathbf{S}_{XY})_{\mathcal{G}}$, and $Z \in Ne_{\mathcal{G}}(X)$. Z is a common child of X and Y, i.e., $(X \to Z \leftarrow Y) \in VS_{\mathcal{G}}(X)$, if and only if*

$$Z \notin \mathbf{S}_{XY} \quad \textit{and} \quad (Y \not\perp\!\!\!\perp Z|\mathbf{S})_{\mathcal{G}}\,, \quad \forall \mathbf{S} \subseteq Mb_{\mathcal{G}}(X) \cup \{X\} \setminus \{Y, Z\}.$$

After learning $Ne_{\mathcal{G}}(X)$ and $\mathrm{VS}_{\mathcal{G}}(X)$ using Lemmas 27 and 28, MARVEL utilizes the following two lemmas to verify the two conditions of Theorem 18.

Lemma 29 *Variable X satisfies Condition 1 of Theorem 18 if and only if*

$$(Y \not\perp\!\!\!\perp Z|\mathbf{S} \cup \{X\})_{\mathcal{G}}\,, \quad \forall Y, Z \in Ne_{\mathcal{G}}(X), \mathbf{S} \subseteq Mb_{\mathcal{G}}(X) \setminus \{Y, Z\}.$$

Lemma 30 *Suppose variable X satisfies Condition 1 of Theorem 18. Then X satisfies Condition 2 of Theorem 18, and therefore, X is removable in $\mathcal{G}$, if and only if*

$$(Y \not\perp\!\!\!\perp Z|\mathbf{S} \cup \{X, V\})_{\mathcal{G}}\,,$$
$$\forall (X \to V \leftarrow Y) \in VS_{\mathcal{G}}(X), Z \in Ne_{\mathcal{G}}(X) \setminus \{V\}, \mathbf{S} \subseteq Mb_{\mathcal{G}}(X) \setminus \{V, Y, Z\}.$$

The following corollary outlines the computational complexity of applying these lemmas.

Corollary 31 *Given $Mb_{\mathcal{G}}(X)$, by applying Lemmas 27-30, we can identify $Ne_{\mathcal{G}}(X), \Lambda_{\mathcal{G}}(X)$, $VS_{\mathcal{G}}(X)$, and determine whether X is removable in $\mathcal{G}$ by performing at most*

$$\mathcal{O}\left(|Mb_{\mathcal{G}}(X)|^2 2^{|Mb_{\mathcal{G}}(X)|}\right)$$

unique CI tests.

4.3 L-MARVEL: Latent-MARVEL

L-MARVEL extends MARVEL to the case where causal sufficiency does not necessarily hold, i.e., the causal graph is a MAG (Akbari et al., 2021).[4] To verify the removability of a variable X in a MAG $\mathcal{G}$, L-MARVEL first learns $Ne_{\mathcal{G}}(X)$ using Lemma 27 as follows. If $Y \in \mathrm{Mb}_{\mathcal{G}}(X)$ is not a neighbor of X, then X and Y have a separating set in $\mathrm{Mb}_{\mathcal{G}}(X) \setminus \{Y\}$. Hence, identifying $Ne_{\mathcal{G}}(X)$ can be performed using a brute-force search in the Markov boundary, using at most

$$|\mathrm{Mb}_{\mathcal{G}}(X)|2^{|\mathrm{Mb}_{\mathcal{G}}(X)|-1}$$

CI tests. After learning $Ne_{\mathcal{G}}(X)$, L-MARVEL utilizes the following theorem to check the removability of X.

Theorem 32 *In a MAG $\mathcal{G}$ over $\mathbf{V}$, a variable $X \in \mathbf{V}$ is removable if and only if for every $Y \in Mb_{\mathcal{G}}(X)$ and $Z \in Ne_{\mathcal{G}}(X) \setminus \{Y\}$, at least one of the following holds.*

Condition 1: $\exists \mathbf{W} \subseteq Mb_{\mathcal{G}}(X) \setminus \{Y, Z\}: \quad (Y \perp\!\!\!\perp Z|\mathbf{W})_{\mathcal{G}}$.

Condition 2: $\forall \mathbf{W} \subseteq Mb_{\mathcal{G}}(X) \setminus \{Y, Z\}: \quad (Y \not\perp\!\!\!\perp Z|\mathbf{W} \cup \{X\})_{\mathcal{G}}$.

Remark 33 *If $Y \in Ne_{\mathcal{G}}(X)$, we can skip Condition 1 and only check Condition 2.*

Below, we present a proposition that we use in Section 5.2 to avoid performing duplicate CI tests in the implementation of L-MARVEL.

Proposition 34 *Suppose $\mathcal{G}$ is a MAG with the set of vertices $\mathbf{V}$, $X \in \mathbf{V}$, $Y \in Mb_{\mathcal{G}}(X)$, and $Z \in Ne_{\mathcal{G}}(X) \setminus \{Y\}$. If at least one of the two conditions of Theorem 32 holds for X, Y, Z, then the graphical characterization for MAGs in Theorem 19 also holds for X, Y, Z.*

4.4 RSL: Recursive Structure Learning

Another recursive algorithm for causal discovery is RSL, which aims to reduce the computational complexity of causal discovery under structural assumptions. RSL requires causal sufficiency and provides algorithms under two types of structural side information: (I) an upper bound on the clique number of the graph is known, or (II) the graph is diamond-free. The causal discovery algorithms provided under these assumptions are RSL_{ω} and RSL_{D}, respectively. Under the corresponding assumptions, both of these methods achieve polynomial-time complexity.

4.4.1 RSL_{ω}

RSL_{ω} assumes that an upper bound m on the clique number of the causal graph is known, i.e., $\omega(\mathcal{G}) \leq m$.

Remark 35 *For a random graph $\mathcal{G}$ generated from Erdos-Renyi model $G(n, p)$, $\omega(\mathcal{G}) \leq m$ with high probability when $pn^{2/m} \to 0$ as $n \to \infty$.*

4. L-MARVEL, as presented in Akbari et al. (2021), can also handle the presence of selection bias. In this paper however, for the sake of simplicity, we assume there is no selection bias, and we have access to i.i.d. samples from $P_{\mathbf{V}}$.

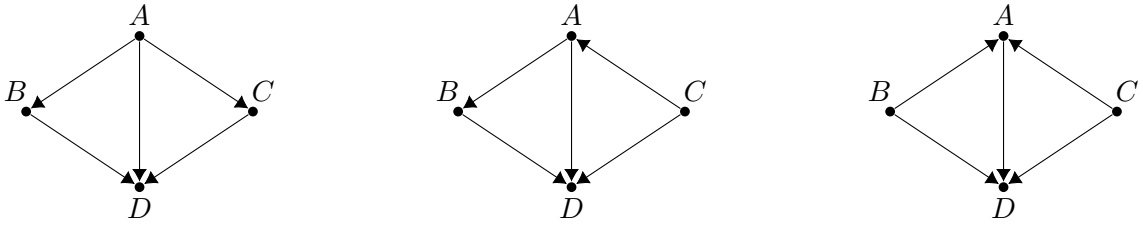


Figure 5: Diamond graphs.

RSL_ω provides the following result for verifying the removability of a variable under the assumption $\omega(\mathcal{G}) \leq m$.

Theorem 36 *Suppose $\mathcal{G}$ is a DAG such that $\omega(\mathcal{G}) \leq m$. Vertex X is removable in $\mathcal{G}$ if for any $\mathbf{S} \subseteq Mb_{\mathcal{G}}(X)$ with $|\mathbf{S}| \leq m-2$, the following conditions hold.*

Condition 1: $\big(Y \not\perp\!\!\!\perp Z | \big(Mb_{\mathcal{G}}(X) \cup \{X\}\big) \setminus \big(\{Y, Z\} \cup \mathbf{S}\big)\big)_{\mathcal{G}}\,, \quad \forall Y, Z \in Mb_{\mathcal{G}}(X) \setminus \mathbf{S}.$

Condition 2: $(X \not\perp\!\!\!\perp Y | Mb_{\mathcal{G}}(X) \setminus (\{Y\} \cup \mathbf{S}))_{\mathcal{G}}\,, \quad \forall Y \in Mb_{\mathcal{G}}(X) \setminus \mathbf{S}.$

Also, the set of vertices that satisfy these conditions is nonempty.

To identify the neighbors of a removable variable detected using Theorem 36, RSL_ω provides the following proposition that distinguishes co-parents from neighbors in the Markov boundary.

Proposition 37 *Suppose $\mathcal{G}$ is a DAG such that $\omega(\mathcal{G}) \leq m$. Let X be a vertex that satisfies the two conditions of Theorem 36 and $Y \in Mb_{\mathcal{G}}(X)$. Then, $Y \in \Lambda_{\mathcal{G}}(X)$ if and only if*

$$\exists \mathbf{S} \subseteq Mb_{\mathcal{G}}(X) \setminus \{Y\}: \quad |\mathbf{S}| = m-1\,, \quad (X \perp\!\!\!\perp Y | Mb_{\mathcal{G}}(X) \setminus (\{Y\} \cup \mathbf{S}))_{\mathcal{G}}\,.$$

Moreover, set $\mathbf{S}$ is unique and $\mathbf{S} = Ch_{\mathcal{G}}(X) \cap Ch_{\mathcal{G}}(Y)$.

4.4.2 RSL_D

RSL_D assumes that the causal DAG is diamond-free, i.e., it contains no diamond as an induced subgraph. Diamond is one of the three types of DAGs shown in Figure 5.

Remark 38 *A random graph $\mathcal{G}$ generated from Erdos-Renyi model $G(n,p)$ is diamond-free with high probability when $pn^{0.8} \to 0$.*

The following theorem provides an efficient method for checking the removability of a variable when the causal DAG is diamond-free.

Theorem 39 *In a diamond-free DAG $\mathcal{G}$, a vertex X is removable if and only if*

$$(Y \not\perp\!\!\!\perp Z | (Mb_{\mathcal{G}}(X) \cup \{X\}) \setminus \{Y, Z\})_{\mathcal{G}}\,, \quad \forall Y, Z \in Mb_{\mathcal{G}}(X).$$

Analogous to the case with the bounded clique number, the following proposition can be used to learn the neighbors of a removable variable in a diamond-free DAG.

Proposition 40 *In a diamond-free DAG $\mathcal{G}$, let X be a removable variable and $Y \in Mb_{\mathcal{G}}(X)$. In this case, $Y \in \Lambda_{\mathcal{G}}(X)$ if and only if*

$$\exists Z \in Mb_{\mathcal{G}}(X) \setminus \{Y\} : \quad (X \perp\!\!\!\perp Y | Mb_{\mathcal{G}}(X) \setminus \{Y, Z\})_{\mathcal{G}} . \tag{7}$$

Moreover, such a variable Z is unique and $\{Z\} = Ch_{\mathcal{G}}(X) \cap Ch_{\mathcal{G}}(Y)$.

4.5 Removable-Order Learning: ROL

ROL is an ordering-based method that leverages the notion of removability for causal discovery and does not require the assumption of causal sufficiency. As discussed in Section 3.3, ordering-based methods in the literature prior to this approach recover a graph through learning a causal order (c-order) of DAGs, which is a topological order of variables (Definition 23). ROL introduces and uses a novel order called removable order (r-order), which we defined for MAGs in Definition 24.

Note that in our general framework given in Algorithm 1, $(X_1, \ldots, X_n)$ forms an r-order. While the recursive methods that we discussed in the previous subsections seek to identify a removable variable in each iteration, ROL aims to learn the whole order at once. To this end, ROL first proposes a recursive algorithm that learns an undirected graph, whose pseudocode is given in Algorithm 3.

Algorithm 3: Learning $\mathcal{G}^{\pi}$.

Function LearnGPi (π, Data($\mathbf{V}$))
$\mathbf{V}_1 \leftarrow \mathbf{V}$, $\mathbf{E}^{\pi} \leftarrow \varnothing$
for $t = 1$ to $n - 1$ **do**
  $X_t \leftarrow \pi(t)$
  $Ne_{\mathbf{V}_t}(X_t) \leftarrow$ **FindNeighbors**$(X_t, \text{Data}(\mathbf{V}_t))$
  Add undirected edges between X_t and the variables in $Ne_{\mathbf{V}_t}(X_t)$ to $\mathbf{E}^{\pi}$.
  $\mathbf{V}_{t+1} \leftarrow \mathbf{V}_t \setminus \{X_t\}$
Return $\mathcal{G}^{\pi} = (\mathbf{V}, \mathbf{E}^{\pi})$

Algorithm 3 removes variables according to an arbitrary given order π and learns an undirected graph $\mathcal{G}^{\pi}$ recursively. Using this algorithm, ROL defines the cost of an order π to be the number of edges in $\mathcal{G}^{\pi}$, denoted by $|\mathbf{E}^{\pi}|$. It then defines an optimization problem that seeks to find an order with the minimum cost.

Theorem 41 (Consistency) *Any solution of the optimization problem*

$$\arg\min_{\pi} |\mathbf{E}^{\pi}|, \tag{8}$$

is a member of $\Pi^{r}(\mathcal{G})$. Conversely every member of $\Pi^{r}(\mathcal{G})$ is a solution of (8).

Theorem 41 shows that finding an r-order is equivalent to solving (8). To solve this optimization problem, ROL proposes three algorithms.

- ROL_{HC}, a Hill-climbing-based heuristic algorithm that is scalable to large graphs.

- ROL_{VI}, an exact reinforcement learning (RL)-based algorithm that has theoretical guarantees but is not scalable to large graphs.
- ROL_{PG}, an approximate RL-based algorithm that exploits stochastic policy gradient.

In Section 5.4, we will delve into the details of these methods. Herein, we discuss how ROL formulates the optimization problem in Theorem 41 as an RL problem.

ROL interprets the process of recovering $\mathcal{G}^\pi$ from a given order π as a Markov decision process (MDP), where the index t denotes time, the action space is the set of variables $\mathbf{V}$, and the state space is the set of all subsets of $\mathbf{V}$. More precisely, let s_t and a_t denote the state and the action of the MDP at iteration t, respectively. Then, s_t is the remaining variables at time t (i.e., $s_t = \mathbf{V}_t$) and action a_t is the variable that will be removed from $\mathbf{V}_t$ in that iteration (i.e., $a_t = X_t$). Consequently, the state transition due to action a_t is $s_{t+1} = \mathbf{V}_t \setminus \{a_t\}$. The immediate reward of selecting action a_t at state s_t will be the negative of the instantaneous cost, naturally defined as the number of discovered neighbors for a_t by **FindNeighbors** in line 5 of Algorithm 3. Formally, the reward of picking action a_t when in state s_t is thus given by

$$r\left(s_t, a_t\right) = \left|\textbf{FindNeighbors}\left(a_t, \text{Data}\left(s_t\right)\right)\right| = -\left|Ne_{s_t}(a_t)\right|.$$

In Sections 5.4.2 and 5.4.3, we discuss two of our RL-based approaches with the above formulation.

5. Implementation Details

In Section 4, we explored techniques for identifying removable variables that can be used in various recursive causal discovery algorithms. However, certain details were left out. In this section, we will discuss implementation details and provide pseudocode for these methods. Moreover, in Section 5.7, we discuss how our methods can be augmented to identify the MEC of the underlying causal graph.

5.1 MARVEL

Algorithm 4 provides a pseudocode for MARVEL, which is compatible with the generic frameworks of Algorithms 1 and 2. Algorithm 5 presents the main functions that MARVEL uses to learn neighbors and v-structures, as well as to verify the removability of a variable. As elaborated in Section 5.1.1, MARVEL incorporates three data structures - SkipCheck_vec, SkipCheck_cond1, and SkipCheck_cond2. These structures are designed to avoid performing redundant CI tests and improve overall computational efficiency.

Algorithm 4 initializes the required variables in lines 2–5. It then calls the **ComputeMb** function to initially compute the Markov boundaries. We will discuss this step in Section 5.5. In the for loop of lines 11–30, the removability of variable X_j is checked, and its neighbors among the remaining variables are learned. If the neighbors of X_j have not been learned in the previous iterations, the algorithm calls the **FindNeighbors** function in Algorithm 5 to learn its neighbors. This function uses Lemma 27 to learn the following.

$$Ne_{\mathbf{V}_i}(X_j), \quad \Lambda_{\mathbf{V}_i}(X_j), \quad \{\mathbf{S}_{X_jY} : Y \in \Lambda_{\mathbf{V}_i}(X_j), \left(X_j \perp\!\!\!\perp Y | \mathbf{S}_{X_jY}\right)_{P_{\mathbf{V}_i}}\}.$$

Algorithm 4: MARVEL

1: **Input:** Data($\mathbf{V}$)
2: Initialize undirected graph $\hat{\mathcal{G}} = (\mathbf{V}, \mathbf{E} = \varnothing)$
3: $\mathbf{V}_1 \leftarrow \mathbf{V}$
4: $\forall X \in \mathbf{V}$: SkipCheck_vec$(X) \leftarrow$ False
5: $\forall X, Y, Z \in \mathbf{V}$: SkipCheck_cond1$(X, Y, Z)$, SkipCheck_cond2$(X, Y, Z) \leftarrow$ False
6: $\text{Mb}_{\mathbf{V}_1} \leftarrow$ **ComputeMb**(Data($\mathbf{V}$))
7: **for** i from 1 to $n-1$ **do**
8: $\quad \tilde{\mathbf{V}}_i \leftarrow \{X \in \mathbf{V}_i | \text{SkipCheck\_vec}(X_j) = \text{False}\}$
9: $\quad r \leftarrow |\tilde{\mathbf{V}}_i|$
10: $\quad (X_1, \ldots, X_r) \leftarrow$ Sort $\tilde{\mathbf{V}}_i$ such that $|\text{Mb}_{\mathbf{V}_i}(X_1)| \leq |\text{Mb}_{\mathbf{V}_i}(X_2)| \leq \cdots \leq |\text{Mb}_{\mathbf{V}_i}(X_r)|$
11: $\quad$ **for** j from 1 to r **do**
12: $\quad\quad$ **if** First time applying the **FindNeighbors** function to X_j **then**
13: $\quad\quad\quad Ne_{\mathbf{V}_i}(X_j)$, $\Lambda_{\mathbf{V}_i}(X_j)$, SepSet$(X_j) \leftarrow$ **FindNeighbors**$(X_j, \mathbf{Mb}_{\mathbf{V}_i}(X_j))$
14: $\quad\quad\quad$ Add undirected edges between X_j and $Ne_{\mathbf{V}_i}(X_j)$ in $\hat{\mathcal{G}}$ if not present
15: $\quad\quad$ **else**
16: $\quad\quad\quad Ne_{\mathbf{V}_i}(X_j) \leftarrow Ne_{\hat{\mathcal{G}}[\mathbf{V}_i]}(X_j)$
17: $\quad\quad\quad \Lambda_{\mathbf{V}_i}(X_j) \leftarrow \text{Mb}_{\mathbf{V}_i}(X_j) \setminus Ne_{\mathbf{V}_i}(X_j)$
18: $\quad\quad$ **if Condition1**$(X_j, Ne_{\mathbf{V}_i}(X_j), \text{Mb}_{\mathbf{V}_i}(X_j))$ is True **then**
19: $\quad\quad\quad$ **if** First time applying the **FindVS** function to X_j **then**
20: $\quad\quad\quad\quad \text{VS}(X_j) \leftarrow$ **FindVS**$(X_j, Ne_{\mathbf{V}_i}(X_j), \Lambda_{\mathbf{V}_i}(X_j), \text{Mb}_{\mathbf{V}_i}(X_j), \text{SepSet}(X_j))$
21: $\quad\quad\quad$ **else**
22: $\quad\quad\quad\quad \text{VS}(X_j) \leftarrow \{(X_j \to Z \leftarrow Y) \in \text{VS}(X_j) | Z, Y \in \mathbf{V}_i\}$
23: $\quad\quad\quad$ **if Condition2**$((X_j, Ne_{\mathbf{V}_i}(X_j), \Lambda_{\mathbf{V}_i}(X_j), \text{Mb}_{\mathbf{V}_i}(X_j), \text{VS}(X_j)))$ is True **then**
24: $\quad\quad\quad\quad \mathbf{V}_{i+1} \leftarrow \mathbf{V}_i \setminus \{X_j\}$
25: $\quad\quad\quad\quad \text{Mb}_{\mathbf{V}_{i+1}} \leftarrow$ **UpdateMb**$(X_j, Ne_{\mathbf{V}_i}(X_j), \text{Mb}_{\mathbf{V}_i})$
26: $\quad\quad\quad\quad$ **Break** the for loop of line 11
27: $\quad\quad\quad$ **else**
28: $\quad\quad\quad\quad$ SkipCheck_vec $\leftarrow$ True
29: $\quad\quad$ **else**
30: $\quad\quad\quad$ SkipCheck_vec $\leftarrow$ True
31: **return** $\hat{\mathcal{G}}$

If the neighbors of X_j have been learned in previous iterations, the algorithm uses $\hat{\mathcal{G}}$ to recover this information. Then, upon calling function **Condition1**, we verify the first condition of Theorem 18, which is designed in accordance with Lemma 29. If this condition is satisfied, Lemma 28 is applied (function **FindVS**) to learn $\text{VS}_{\mathbf{V}_i}(X_j)$. Note that if $\text{VS}_{\mathbf{V}_i}(X)$ is learned using Lemma 28 in an iteration, we can save it and delete a v-structure from it when one of the three variables of the v-structure is removed. Finally, the second condition of Theorem 18 is verified using Lemma 30 (function **Condition2**). If this condition is also satisfied, the algorithm concludes that X_j is removable and proceeds to remove it from the variables. For the next iteration, we update the Markov boundaries by calling the **UpdateMb** function in Algorithm 11, which we discuss in Section 5.6.

Algorithm 5: MARVEL functions

Function FindNeighbors(X, $\mathrm{Mb}_{\mathbf{W}}(X)$) % Lemma 27
$Ne_{\mathbf{W}}(X)$, $\Lambda_{\mathbf{W}}(X)$, $\mathrm{SepSet}(X) \leftarrow \varnothing$
for $Y \in \mathrm{Mb}_{\mathbf{W}}(X)$ **do**
  if $\exists \mathbf{S}_{X,Y} \subsetneq \mathrm{Mb}_{\mathbf{W}}(X) \setminus \{Y\} : \quad (X \perp\!\!\!\perp Y | \mathbf{S}_{X,Y})_{\mathrm{Data}(\mathbf{W})}$ **then**
    Add Y to $\Lambda_{\mathbf{W}}(X)$
    Add $\mathbf{S}_{X,Y}$ to $\mathrm{SepSet}(X)$
  else
    Add Y to $Ne_{\mathbf{W}}(X)$
return $Ne_{\mathbf{W}}(X)$, $\Lambda_{\mathbf{W}}(X)$, $\mathrm{SepSet}(X)$

Function FindVS(X, $Ne_{\mathbf{W}}(X)$, $\Lambda_{\mathbf{W}}(X)$, $\mathrm{Mb}_{\mathbf{W}}(X)$, $\mathrm{SepSet}(X)$) % Lemma 28
$\mathrm{VS}(X) \leftarrow \varnothing$
for $Y \in \Lambda_{\mathbf{W}}(X)$ and $Z \in Ne_{\mathbf{W}}(X)$ **do**
  $\mathbf{S}_{X,Y} \leftarrow$ The separating set in SepSet corresponding to Y
  if $Z \notin \mathbf{S}_{XY}$ and $\forall \mathbf{S} \subseteq \mathrm{Mb}_{\mathbf{W}}(X) \cup \{X\} \setminus \{Y, Z\} : \quad (Y \not\perp\!\!\!\perp Z | \mathbf{S})_{\mathrm{Data}(\mathbf{W})}$ **then**
    Add $(X \rightarrow Z \leftarrow Y)$ to $\mathrm{VS}(X)$
return $\mathrm{VS}(X)$

Function Condition1(X, $Ne_{\mathbf{W}}(X)$, $\mathrm{Mb}_{\mathbf{W}}(X)$) % Lemma 29
for $Y, Z \in Ne_{\mathbf{W}}(X)$ such that SkipCheck_cond1(X, Y, Z) is False **do**
  for $\mathbf{S} \subseteq \mathrm{Mb}_{\mathbf{W}}(X) \setminus \{Y, Z\}$ **do**
    if $(Y \perp\!\!\!\perp Z | \mathbf{S} \cup \{X\})_{\mathrm{Data}(\mathbf{W})}$ **then**
      return False
  SkipCheck_cond1$(X, Y, Z) \leftarrow$ True
return True

Function Condition2(X, $Ne_{\mathbf{W}}(X)$, $\Lambda_{\mathbf{W}}(X)$, $\mathrm{Mb}_{\mathbf{W}}(X)$, $\mathrm{VS}(X)$) % Lemma 30
for $Y \in \Lambda_{\mathbf{W}}(X)$ and $Z \in Ne_{\mathbf{W}}(X)$ such that SkipCheck_cond2(X, Y, Z) is False **do**
  $\mathbf{\Gamma} \leftarrow \{V \neq Z : (X \rightarrow V \leftarrow Y) \in \mathrm{VS}(X)\}$
  for $\mathbf{S} \subseteq \mathrm{Mb}_{\mathbf{W}}(X) \setminus \{Y, Z\}$ s.t. $\mathbf{S} \cap \mathbf{\Gamma} \neq \emptyset$ **do**
    if $(Y \perp\!\!\!\perp Z | \mathbf{S} \cup \{X\})_{\mathrm{Data}(\mathbf{W})}$ **then**
      return False
  SkipCheck_cond2$(X, Y, Z) \leftarrow$ True
return True

5.1.1 Avoiding Duplicate CI Tests in MARVEL

Suppose that MARVEL verifies the removability of a variable X and determines that it is not removable. As a result, the algorithm will need to verify again the removability of X in some of the following iterations, potentially leading to redundant CI tests. To address

this, we propose a method to eliminate such redundancies by leveraging information from previous iterations.

Suppose that during iteration i, we invoke **Condition1** from Algorithm 5 for a variable X, where $\mathbf{W} = \mathbf{V}_i$. If two variables $Y, Z \in Ne_{\mathbf{V}_i}(X)$ do not have a separating set in $\text{Mb}_{\mathbf{V}_i}(X) \setminus \{Y, Z\}$, then they will not have a separating set in $\text{Mb}_{\mathbf{V}_{i'}}(X) \setminus \{Y, Z\}$ for any $i' > i$. Accordingly, to prevent redundant CI tests, SkipCheck_cond1 is employed in function **Condition1** to save this information and avoid performing duplicate CI tests. A similar approach is adopted in **Condition2** using SkipCheck_cond2.

To further enhance the implementation, SkipCheck_vec is integrated into Algorithm 4. This is based on the understanding that a variable's removability hinges on its Markov boundary. If a variable X is found non-removable in one iteration, it remains so as long as its Mb is unchanged, thereby obviating the need to recheck within the for loop of lines 11–30. Initially, every variable in $\mathbf{V}$ has SkipCheck_vec set to False. Should either function **Condition1** or **Condition2** return False, we switch SkipCheck_vec(X) to True. SkipCheck_vec(X) stays True unless there is a change in its Mb. As detailed in Section 5.6, updating SkipCheck_vec is crucial and is done in the **UpdateMb** function.

5.2 L-MARVEL

Similar to MARVEL, we provide a pseudocode for L-MARVEL in Algorithm 6.

In the for loop of lines 11–22, the algorithm learns the neighbors of X_j and verifies its removability. If the algorithm is learning the neighbors of X_j for the first time, it uses Lemma 27 (function **FindNeighbors**) to learn $Ne_{\mathbf{V}_i}(X_j)$. Otherwise, this information has already been stored in $\hat{\mathcal{G}}$, and the algorithm executes line 13 to recover the neighbors of X_j among the remaining variables. To verify the removability of X_j, the **IsRemovable** function is invoked. This function checks the two conditions of Theorem 32. If both conditions are not met, the function determines that X_j is not removable.

Algorithm 6 uses SkipCheck_vec and SkipCheck_mat to avoid performing duplicate CI tests. Similar to MARVEL, if a variable is found non-removable in one iteration, its SkipCheck_vec value is set to True, and it remains so as long as the Markov boundary of the variable remains unchanged. Additionally, Proposition 34 implies that if the condition of Theorem 32 is met for Y and Z during an iteration in the **IsRemovable** function, the graphical characterization of Theorem 19 holds for X, Y, Z. Therefore, to check the removability of X, it is not necessary to check the two conditions of Theorem 32. Accordingly, SkipCheck_mat is employed in the **IsRemovable** function to avoid performing redundant CI tests.

5.3 RSL

In this part, we discuss the implementation details of RSL_ω and RSL_D.

5.3.1 RSL_ω

Algorithm 7 presents the pseudocode for RSL_ω. This algorithm takes m as an input, which is an upper bound on the clique number of the true underlying graph. One of the differences between this algorithm and MARVEL and L-MARVEL is the sequence of operations. This algorithm first checks the removability of a variable using Theorem 36

Algorithm 6: L-MARVEL

1: **Input:** Data($\mathbf{V}$)
2: Initialize undirected graph $\hat{\mathcal{G}} = (\mathbf{V}, \mathbf{E} = \varnothing)$
3: $\forall X \in \mathbf{V}$: SkipCheck_vec(X) $\leftarrow$ False
4: $\forall X, Y, Z \in \mathbf{V}$: SkipCheck_mat($X, Y, Z$) $\leftarrow$ False
5: $\mathbf{V}_1 \leftarrow \mathbf{V}$
6: $\text{Mb}_{\mathbf{V}_1} \leftarrow$ **ComputeMb**(Data($\mathbf{V}$))
7: **for** i from 1 to $n-1$ **do**
8: $\quad \tilde{\mathbf{V}}_i \leftarrow \{X \in \mathbf{V}_i | \text{SkipCheck\_vec}(X_j) = \text{False}\}$
9: $\quad r \leftarrow |\tilde{\mathbf{V}}_i|$
10: $\quad (X_1, \ldots, X_r) \leftarrow$ Sort $\tilde{\mathbf{V}}_i$ such that $|\text{Mb}_{\mathbf{V}_i}(X_1)| \leq |\text{Mb}_{\mathbf{V}_i}(X_2)| \leq \cdots \leq |\text{Mb}_{\mathbf{V}_i}(X_r)|$
11: $\quad$ **for** j from 1 to r **do**
12: $\quad\quad$ **if** First time learning the neighbors of X_j **then**
13: $\quad\quad\quad Ne_{\mathbf{V}_i}(X_j) \leftarrow$ **FindNeighbors**(X_j, $\text{Mb}_{\mathbf{V}_i}(X_j)$)
14: $\quad\quad\quad$ Add undirected edges between X_j and $Ne_{\mathbf{V}_i}(X_j)$ in $\hat{\mathcal{G}}$ if not present
15: $\quad\quad$ **else**
16: $\quad\quad\quad Ne_{\mathbf{V}_i}(X_j) \leftarrow Ne_{\hat{\mathcal{G}}[\mathbf{V}_i]}(X_j)$
17: $\quad\quad$ **if IsRemovable**(X_j, $Ne_{\mathbf{V}_i}(X_j)$, $\text{Mb}_{\mathbf{V}_i}(X_j)$)) is True **then**
18: $\quad\quad\quad \mathbf{V}_{i+1} \leftarrow \mathbf{V}_i \setminus \{X_j\}$
19: $\quad\quad\quad \text{Mb}_{\mathbf{V}_{i+1}} \leftarrow$ **UpdateMb**(X_j, $Ne_{\mathbf{V}_i}(X_j)$, $\text{Mb}_{\mathbf{V}_i}$)
20: $\quad\quad\quad$ **Break** the for loop of line 11
21: $\quad\quad$ **else**
22: $\quad\quad\quad$ SkipCheck_vec(X_j) $\leftarrow$ True
23: **return** $\hat{\mathcal{G}}$

1: **Function FindNeighbors**(X, $\text{Mb}_{\mathbf{W}}(X)$) % Lemma 27
2: $Ne_{\mathbf{W}}(X) \leftarrow \text{Mb}_{\mathbf{W}}(X)$
3: **for** $Y \in \text{Mb}_{\mathbf{W}}(X)$ **do**
4: $\quad$ **if** $\exists \mathbf{S} \subsetneq \text{Mb}_{\mathbf{W}}(X) \setminus \{Y\}$: $\quad (X \perp\!\!\!\perp Y | \mathbf{S})_{\text{Data}(\mathbf{W})}$ **then**
5: $\quad\quad$ Remove Y from $Ne_{\mathbf{W}}(X)$
6: **return** $Ne_{\mathbf{W}}(X)$

1: **Function IsRemovable**(X, $Ne_{\mathbf{W}}(X)$, $\text{Mb}_{\mathbf{W}}(X)$) % Theorem 32
2: **for** $Y \in \text{Mb}_{\mathbf{W}}(X)$ and $Z \in Ne_{\mathbf{W}}(X)$ such that SkipCheck_mat(X, Y, Z) is False **do**
3: $\quad$ **if** $\forall \mathbf{S} \subseteq \text{Mb}_{\mathbf{W}}(X) \setminus \{Y, Z\}$: $\quad (Y \not\perp\!\!\!\perp Z | \mathbf{S})_{\text{Data}(\mathbf{W})}$ **then**
4: $\quad\quad$ **if** $\exists \mathbf{S} \subseteq \text{Mb}_{\mathbf{W}}(X) \setminus \{Y, Z\}$: $\quad (Y \perp\!\!\!\perp Z | \mathbf{S} \cup \{X\})_{\text{Data}(\mathbf{W})}$ **then**
5: $\quad\quad\quad$ **Return** False
6: $\quad$ SkipCheck_mat(X, Y, Z) $\leftarrow$ True
7: **Return** True

through the function **IsRemovable**. Subsequently, it finds the neighbors of the variable by employing Proposition 37 in the function **FindNeighbors**. Furthermore, Algorithm 9 only uses one data structure, SkipCheck_vec, to efficiently prevent redundant CI tests.

As mentioned above, Algorithm 7 takes m, an upper bound on the clique number of the causal graph. But what happens if it is not a valid upper bound, i.e., $m < \omega(\mathcal{G})$? In this scenario, two outcomes are possible: either Algorithm 7 is unable to identify any removable variables at an iteration and halts, or RSL_ω terminates and returns a graph.

Proposition 42 (RSL_ω is verifiable) *If Algorithm 7 terminates with an input $m > 0$, then the clique number of the learned skeleton is greater than or equal to the clique number of the true causal graph.*

Proposition 42 implies that, upon termination of the algorithm, if the clique number of the learned graph is at most m, then m is a valid upper bound on the clique number, ensuring the correctness of the output. Otherwise, it indicates that the true clique number is greater than m. Accordingly, we present Algorithm 8 that can learn the skeleton of $\mathcal{G}$ without any prior knowledge about $\omega(\mathcal{G})$.

5.3.2 RSL_D

We present Algorithm 9 for RSL_D. The main body of the algorithm is the same as Algorithm 7 for RSL_ω. However, the difference lies in the **IsRemovable** and **FindNeighbors** functions. The former uses Theorem 39 to check the removability of a variable, while the latter uses Proposition 40 to learn the neighbors of a removable variable. Note that RSL_D assumes that the underlying graph is diamond-free.

5.4 ROL

As we discussed in Section 4.5, ROL aims to solve the optimization problem described in Equation (8), which uses Algorithm 3 to define a cost function for a given permutation. Additionally, we presented a reinforcement learning (RL) formulation to solve this optimization problem. In this section, we present the implementation details of three approaches for learning an r-order through solving Equation (8).

5.4.1 ROL_{HC}

In Algorithm 10, we propose a hill-climbing approach, called ROL_{HC} for finding an r-order. In general, the output of Algorithm 10 is a suboptimal solution to (8) as it takes an initial order π and gradually modifies it to another order with less cost, but it is not guaranteed to find a minimizer of (8) by taking such greedy approach. Nevertheless, this algorithm is suitable for practice as it is scalable to large graphs, and also achieves superior accuracy compared to the state-of-the-art methods.

Inputs to Algorithm 10 are the observational data Data($\mathbf{V}$) and two parameters maxIter and maxSwap. maxIter denotes the maximum number of iterations before the algorithm terminates, and maxSwap is an upper bound on the index difference of two variables that can get swapped in an iteration (line 6). Initial order π in line 2 can be any arbitrary order, but selecting it cleverly (such as initialization using the output of other approaches) will improve the performance of the algorithm.

Algorithm 7: RSL$_\omega$

Input: Data($\mathbf{V}$), m
Initialize undirected graph $\hat{\mathcal{G}} = (\mathbf{V}, \mathbf{E} = \varnothing)$
$\forall X \in \mathbf{V}$: SkipCheck_vec(X) $\leftarrow$ False
$\mathbf{V}_1 \leftarrow \mathbf{V}$
$\text{Mb}_{\mathbf{V}_1} \leftarrow$ **ComputeMb**(Data($\mathbf{V}$))
for i from 1 to $n-1$ **do**
  $\tilde{\mathbf{V}}_i \leftarrow \{X \in \mathbf{V}_i | \text{SkipCheck\_vec}(X_j) = \text{False}\}$
  $r \leftarrow |\tilde{\mathbf{V}}_i|$
  $(X_1, \ldots, X_r) \leftarrow$ Sort $\tilde{\mathbf{V}}_i$ such that $|\text{Mb}_{\mathbf{V}_i}(X_1)| \leq |\text{Mb}_{\mathbf{V}_i}(X_2)| \leq \cdots \leq |\text{Mb}_{\mathbf{V}_i}(X_r)|$
  for j from 1 to r **do**
    if IsRemovable(X_j, $\text{Mb}_{\mathbf{V}_i}(X_j)$), m) is True **then**
      $Ne_{\mathbf{V}_i}(X_j) \leftarrow$ **FindNeighbors**(X_j, $\text{Mb}_{\mathbf{V}_i}(X_j)$, m)
      Add undirected edges between X_j and $Ne_{\mathbf{V}_i}(X_j)$ in $\hat{\mathcal{G}}$ if not present
      $\mathbf{V}_{i+1} \leftarrow \mathbf{V}_i \setminus \{X_j\}$
      $\text{Mb}_{\mathbf{V}_{i+1}} \leftarrow$ **UpdateMb**(X_j, $Ne_{\mathbf{V}_i}(X_j)$, $\text{Mb}_{\mathbf{V}_i}$)
      Break the for loop of line 10
    else
      SkipCheck_vec(X_j) $\leftarrow$ True
return $\hat{\mathcal{G}}$

Function IsRemovable(X, $\text{Mb}_{\mathbf{W}}(X)$, m) % Theorem 36
for $\mathbf{S} \subseteq \text{Mb}_{\mathbf{W}}(X)$ with $|\mathbf{S}| \leq m-2$ **do**
  if $\Big(\exists Y, Z \in \text{Mb}_{\mathbf{W}}(X) \setminus \mathbf{S}: \quad \big(Y \perp\!\!\!\perp Z | \big(\text{Mb}_{\mathbf{W}}(X) \cup \{X\}\big) \setminus \big(\{Y, Z\} \cup \mathbf{S}\big)\big)_{\text{Data}(\mathbf{W})}\Big)$
  or $\Big(\exists Y \in \text{Mb}_{\mathbf{W}}(X) \setminus \mathbf{S}: \quad (X \perp\!\!\!\perp Y | \text{Mb}_{\mathbf{W}}(X) \setminus (\{Y\} \cup \mathbf{S}))_{\text{Data}(\mathbf{W})}\Big)$ **then**
    Return False
Return True

Function FindNeighbors(X, $\text{Mb}_{\mathbf{W}}(X)$, m) % Proposition 37
$Ne_{\mathbf{W}}(X) \leftarrow \text{Mb}_{\mathbf{W}}(X)$
for $Y \in \text{Mb}_{\mathbf{W}}(X)$ **do**
  if $\exists \mathbf{S} \subseteq \text{Mb}_{\mathbf{W}}(X) \setminus \{Y\} : |\mathbf{S}| = m-1, (X \perp\!\!\!\perp Y | \text{Mb}_{\mathbf{W}}(X) \setminus (\{Y\} \cup \mathbf{S}))_{\text{Data}(\mathbf{W})}$ **then**
    Remove Y from $Ne_{\mathbf{W}}(X)$
return $Ne_{\mathbf{W}}(X)$

Algorithm 8: RSL$_\omega$ Without Side Information.

Input: Data($\mathbf{V}$)
for m from 1 to n **do**
  $\hat{\mathcal{G}} \leftarrow$ **RSL**$_\omega$(Data($\mathbf{V}$), m)
  if RSL$_\omega$ terminates and $\omega(\hat{\mathcal{G}}) \leq m$ **then**
    return $\hat{\mathcal{G}}$

Algorithm 9: RSL_D

1: **Input:** Data($\mathbf{V}$)
2: Initialize undirected graph $\hat{\mathcal{G}} = (\mathbf{V}, \mathbf{E} = \varnothing)$
3: $\forall X \in \mathbf{V}$: SkipCheck_vec(X) $\leftarrow$ False
4: $\mathbf{V}_1 \leftarrow \mathbf{V}$
5: $\text{Mb}_{\mathbf{V}_1} \leftarrow$ **ComputeMb**(Data($\mathbf{V}$))
6: **for** i from 1 to $n-1$ **do**
7: $\quad \tilde{\mathbf{V}}_i \leftarrow \{X \in \mathbf{V}_i | \text{SkipCheck\_vec}(X_j) = \text{False}\}$
8: $\quad r \leftarrow |\tilde{\mathbf{V}}_i|$
9: $\quad (X_1, \ldots, X_r) \leftarrow$ Sort $\tilde{\mathbf{V}}_i$ such that $|\text{Mb}_{\mathbf{V}_i}(X_1)| \leq |\text{Mb}_{\mathbf{V}_i}(X_2)| \leq \cdots \leq |\text{Mb}_{\mathbf{V}_i}(X_r)|$
10: $\quad$ **for** j from 1 to r **do**
11: $\quad\quad$ **if IsRemovable**(X_j, $\text{Mb}_{\mathbf{V}_i}(X_j)$) is True **then**
12: $\quad\quad\quad Ne_{\mathbf{V}_i}(X_j) \leftarrow$ **FindNeighbors**(X_j, $\text{Mb}_{\mathbf{V}_i}(X_j)$)
13: $\quad\quad\quad$ Add undirected edges between X_j and $Ne_{\mathbf{V}_i}(X_j)$ in $\hat{\mathcal{G}}$ if not present
14: $\quad\quad\quad \mathbf{V}_{i+1} \leftarrow \mathbf{V}_i \setminus \{X_j\}$
15: $\quad\quad\quad \text{Mb}_{\mathbf{V}_{i+1}} \leftarrow$ **UpdateMb**(X_j, $Ne_{\mathbf{V}_i}(X_j)$, $\text{Mb}_{\mathbf{V}_i}$)
16: $\quad\quad\quad$ **Break** the for loop of line 10
17: $\quad\quad$ **else**
18: $\quad\quad\quad$ SkipCheck_vec(X_j) $\leftarrow$ True
19: **return** $\hat{\mathcal{G}}$

1: **Function IsRemovable**(X, $\text{Mb}_{\mathbf{W}}(X)$) % Theorem 39
2: **for** $Y, Z \in \text{Mb}_{\mathbf{W}}(X)$ **do**
3: $\quad$ **if** $(Y \perp\!\!\!\perp Z | (\text{Mb}_{\mathbf{W}}(X) \cup \{X\}) \setminus \{Y, Z\})_{\text{Data}(\mathbf{W})}$ **then**
4: $\quad\quad$ **Return** False
5: **Return** True

1: **Function FindNeighbors**(X, $\text{Mb}_{\mathbf{W}}(X)$) % Proposition 40
2: $Ne_{\mathbf{W}}(X) \leftarrow \text{Mb}_{\mathbf{W}}(X)$
3: **for** $Y \in \text{Mb}_{\mathbf{W}}(X)$ **do**
4: $\quad$ **if** $\exists Z \in \text{Mb}_{\mathbf{W}}(X) \setminus \{Y\}$: $\quad (X \perp\!\!\!\perp Y | \text{Mb}_{\mathbf{W}}(X) \setminus \{Y, Z\})_{\text{Data}(\mathbf{W})}$ **then**
5: $\quad\quad$ Remove Y from $Ne_{\mathbf{W}}(X)$
6: **return** $Ne_{\mathbf{W}}(X)$

The subroutine *ComputeCost* takes as input an order π and two numbers $1 \leq a < b \leq n$ as input and returns a vector $C_{ab} \in \mathbb{R}^n$. For $a \leq t \leq b$, the t-th entry of C_{ab} is equal to $|Ne_{\mathbf{V}_t}(X_t)|$ which is the number of neighbors of X_t in the remaining graph. Hence, to learn the total cost of π, we can call this function with $a = 1$ and $b = n$ and then compute the sum of the entries of the output C_{1n}.

Accordingly, the main algorithm initially computes the cost vector of π in line 3. The remainder of the algorithm (lines 4–13) updates π iteratively, maxIter number of times. It updates the current order $\pi = (X_1, \cdots, X_n)$ as follows: first, it constructs a set of orders $\Pi^{\text{new}} \subseteq \Pi(\mathbf{V})$ from π by swapping any two variables X_a and X_b in π as long as $1 \leq b - a \leq$

maxSwap. Next, for each $\pi_{\text{new}} \in \Pi^{\text{new}}$, it computes the cost of π_{new} and if it has a lower cost compared to the current order, the algorithm replaces π by that order and repeats the process. Note that in line 8, the algorithm calls function *ComputeCost* with a and b to compute the cost of the new policy. The reason is that for $i < a$ and $i > b$, the i-th entry of the cost of π and π_{new} are the same. This is because the set of remaining variables is the same. Hence, to compare the cost of π with the cost of π_{new}, it suffices to compare them for entries between a and b. Accordingly, Algorithm 10 checks the condition in line 9, and if the cost of the new policy is better, then the algorithm updates π and its corresponding cost. Note that it suffices to update the entries between a to b of C_{1n}.

5.4.2 ROL_{VI}

Following the RL formulation of ROL introduced in Section 4.5, we present ROL_{VI}. This algorithm uses value iteration to tackle the problem.

Given a deterministic policy π_θ that is parameterized by θ, we can adapt Algorithm 3 to the RL setting as follows: the algorithm takes as input a policy π_θ instead of a permutation π. Furthermore, it uses the given policy to select X_t in line 4, given by $X_t = \pi_\theta(\mathbf{V}_t)$. Finally, given a policy π_θ, and the initial state $s_1 = \mathbf{V}$, the cumulative reward of a trajectory

Algorithm 10: ROL_{HC}

1: **Input:** Data($\mathbf{V}$), maxSwap, maxIter
2: Initialize $\pi \in \Pi(\mathbf{V})$
3: $C_{1n} \leftarrow$ **ComputeCost**$(\pi, 1, n, \text{Data}(\mathbf{V}))$
4: **for** 1 to maxIter **do**
5: Denote π by $(X_1, \cdots, X_n)$
6: **for** $1 \le a < b \le n$ such that $b - a <$ maxSwap **do**
7: $\pi_{\text{new}} \leftarrow$ Swap X_a and X_b in π
8: $C_{ab}^{\text{new}} \leftarrow$ **ComputeCost**$(\pi_{\text{new}}, a, b, \text{Data}(\mathbf{V}))$
9: **if** $\sum_{i=a}^{b} C_{ab}^{\text{new}}(i) < \sum_{i=a}^{b} C_{1n}(i)$ **then**
10: $\pi \leftarrow \pi_{\text{new}}$
11: **for** j from a to b **do**
12: $C_{1n}(j) \leftarrow C_{ab}^{\text{new}}(j)$
13: **Break** the for loop of line 6
14: **Return** π

1: **Function ComputeCost** (π, a, b, Data($\mathbf{V}$))
2: $\mathbf{V}_a \leftarrow \{X_a, X_{a+1}, \ldots, X_n\}$
3: $C_{ab} \leftarrow (0, 0, \cdots, 0) \in \mathbb{R}^n$
4: **for** $t = a$ to b **do**
5: $X_t \leftarrow \pi(t)$
6: $Ne_{\mathbf{V}_t}(X_t) \leftarrow$ **FindNeighbors**$(X_t, \text{Data}(\mathbf{V}_t))$
7: $C_{ab}(t) \leftarrow |Ne_{\mathbf{V}_t}(X_t)|$
8: $\mathbf{V}_{t+1} \leftarrow \mathbf{V}_t \setminus \{X_t\}$
9: **Return** C_{ab}

$\tau = (s_1, a_1, s_2, a_2, \cdots, s_{n-1}, a_{n-1})$, which denotes the sequence of states and actions selected by π_θ, is given by

$$R(\tau_\theta) = \sum_{t=1}^{n-1} r(s_t, a_t) = -\sum_{t=1}^{n-1} \left| Ne_{\mathcal{G}_{s_t}}(a_t) \right|.$$

Hence, if we denote the output of this modified algorithm by $\mathcal{G}^\theta = \left(\mathbf{V}, \mathbf{E}^\theta\right)$, then $R(\tau_\theta) = -\left|\mathbf{E}^\theta\right|$. With this RL formulation, any optimal policy-finding RL algorithm can be used to find a minimum-cost policy π_θ and thus solve the optimization problem given in Theorem 41. Accordingly, ROL_{VI} applies value iteration algorithm.

5.4.3 ROL_{PG}

Although any algorithm suited for RL is capable of finding an optimal deterministic policy for us, the complexity does not scale well as the graph size. Therefore, we can use a stochastic policy that increases the exploration during the training of an RL algorithm. ROL_{PG} exploit stochastic policies parameterized by neural networks to further improve scalability. However, this could come at the price of approximating the optimal solution instead of finding the exact one. In the stochastic setting, an action a_t is selected according to a distribution over the remaining variables, i.e., $a_t \sim P_\theta(\cdot|s_t = \mathbf{V}_t)$, where θ denotes the parameters of the policy (e.g., the weights used in training of a neural network). In this case, the objective of the algorithm is to minimize the expected total number of edges learned by policy $P_\theta(\cdot|s_t = \mathbf{V}_t)$, i.e.,

$$\arg\max_\theta \mathbb{E}_{\tau_\theta \sim P_\theta}\big[-|\mathbf{E}^\theta|\big], \tag{9}$$

where the expectation is taken w.r.t. randomness of the stochastic policy. Many algorithms have been developed in the literature for finding stochastic policies and solving (9). Some examples include Vanilla Policy Gradient (VPG) (Williams, 1992), REINFORCE (Sutton et al., 1999),and Deep Q-Networks (DQN) (Mnih et al., 2013). Accordingly, ROL_{VI} applies VPG.

5.5 Initially Computing Markov Boundaries

Several algorithms have been proposed in the literature for discovering Markov boundaries (Margaritis and Thrun, 1999; Guyon et al., 2002; Tsamardinos and Aliferis, 2003; Tsamardinos et al., 2003; Yaramakala and Margaritis, 2005; Fu and Desmarais, 2010). One simple approach is to use total conditioning (TC) (Pellet and Elisseeff, 2008b). TC states that X and Y are in each other's Markov boundary if and only if

$$(X \not\perp\!\!\!\perp Y | \mathbf{V} \setminus \{X, Y\})_{P_{\mathbf{V}}}. \tag{10}$$

Using total conditioning, $\binom{n}{2}$ CI tests suffice to identify the Markov boundaries of all vertices. However, each test requires conditioning on a large set of variables. This issue has been addressed in multiple algorithms, including Grow-Shrink (GS) (Margaritis and Thrun, 1999), IAMB (Tsamardinos et al., 2003), and its various modifications. These algorithms

Algorithm 11: Updates Markov boundaries.

1: **UpdateMb**(X, $Ne_{\mathbf{W}}(X)$, $\mathrm{Mb}_{\mathbf{W}}$)
2: $\mathrm{Mb}_{\mathbf{W}\setminus\{X\}} \leftarrow \{\mathrm{Mb}_{\mathbf{W}}(Y)\colon Y \in \mathbf{W}\setminus\{X\}\}$
3: **for** $Y \in \mathrm{Mb}_{\mathbf{W}}(X)$ **do**
4: Remove X from $\mathrm{Mb}_{\mathbf{W}\setminus\{X\}}(Y)$
5: **for** $Y, Z \in Ne_{\mathbf{W}}(X)$ such that $|\mathrm{Mb}_{\mathbf{W}}(Y)| \leq |\mathrm{Mb}_{\mathbf{W}}(Z)|$ **do**
6: **if** $\big(Y \perp\!\!\!\perp Z|\mathrm{Mb}_{\mathbf{W}\setminus\{X\}}(Y)\setminus\{Y,Z\}\big)_{\mathrm{Data}(\mathbf{W})}$ **then**
7: Remove Z from $\mathrm{Mb}_{\mathbf{W}\setminus\{X\}}(Y)$
8: Remove Y from $\mathrm{Mb}_{\mathbf{W}\setminus\{X\}}(Z)$
9: SkipCheck_vec(Y) $\leftarrow$ False
10: SkipCheck_vec(Z) $\leftarrow$ False
11: **return** $\mathrm{Mb}_{\mathbf{W}\setminus\{X\}}$

propose methods that perform more CI tests[5] but with smaller conditioning sets. Choosing the right algorithm for computing the Markov boundaries depends on the available data.

5.6 Updating Markov Boundaries

When a variable is removed in the recursive framework of Algorithm 1, we do not need to recompute the Markov boundaries of all the vertices. Instead, we can update the Markov boundaries of the remaining vertices.

Let $\mathbf{W}$ be the set of variables in an iteration with the set of Markov boundaries $\mathrm{Mb}_{\mathbf{W}}$. Suppose we want to remove a variable X from $\mathbf{W}$ at the end of the current iteration, where $Ne_{\mathbf{W}}(X)$ is the set of neighbors of X. In this case, we only need to compute $\mathrm{Mb}_{\mathbf{W}\setminus\{X\}}$, which is the set of Markov boundaries of the remaining variables.

We can use Algorithm 11 to compute $\mathrm{Mb}_{\mathbf{W}\setminus\{X\}}$. Removing vertex X from MAG $\mathcal{G}_{\mathbf{W}}$ has two effects.

1. X is removed from all Markov boundaries, and
2. for $Y, Z \in \mathbf{W}\setminus\{X\}$, if all of the collider paths between Y and Z in $\mathcal{G}_{\mathbf{W}}$ pass through X, then Y and Z must be excluded from each others Markov boundary.

In the latter case, $Y, Z \in \mathrm{Mb}_{\mathbf{W}}(X)$ and the update is performed using a single CI test,

$$(Y \not\perp\!\!\!\perp Z|\mathrm{Mb}_{\mathbf{W}}(Z)\setminus\{X,Y,Z\})_{P_{\mathbf{W}}}, \quad \text{or equivalently,} \quad (Y \not\perp\!\!\!\perp Z|\mathrm{Mb}_{\mathbf{W}}(Y)\setminus\{X,Y,Z\})_{P_{\mathbf{W}}}.$$

We chose the CI test with the smaller conditioning set. If the CI test shows that Y and Z are conditionally independent, we remove them from each other's Markov boundary.

Recall that we used SkipCheck_vec in the proposed algorithms to avoid unnecessary computations. As discussed in Section 5.1.1, when the Markov boundary of a variable has changed, we need to set its SkipCheck_vec value to be False. Accordingly, when the If condition of line 6 holds, Algorithm 11 removes Y and Z from each other's Markov boundary. Therefore, we set their SkipCheck_vec values to False in lines 9 and 10.

5. These algorithms perform at most $\mathcal{O}(n^2)$ CI tests.

5.7 Identifying the MEC

In the previous sections, we primarily addressed the task of learning the skeleton of the causal graph. In this section, we discuss how our methods can be augmented to identify the MEC of the underlying causal mechanisms ($[\mathcal{G}_{\mathbf{V}}]$). As a general rule, having access to the true skeleton and a separating set for each pair of non-neighbor vertices suffice to identify the MEC (Zhang, 2008b). However, we will provide modifications tailored to a few of our algorithms, which will improve computational complexity.

Algorithms MARVEL and RSL require causal sufficiency. This implies that their objective is to recover the Markov equivalence class of a DAG, $[\mathcal{G}]^d$. Verma and Pearl (1991) showed that two DAGs are Markov equivalent if and only if they have the same skeleton and v-structures. Accordingly, to identify $[\mathcal{G}]^d$, it suffices to learn the skeleton and v-structures of DAG $\mathcal{G}$. As such, in Sections 5.7.1 and 5.7.2 we describe how to recover the v-structures. With this information at hand, Meek rules (Meek, 1995) can be applied to achieve a *maximally oriented* DAG (CPDAG), also known as *essential graph*. For characterization and graph-theoretical properties of such graphs, refer to Andersson et al. (1997).

On the other hand, Algorithms L-MARVEL and ROL serve to recover the MEC of a MAG. Having the same skeleton and unshielded colliders is necessary but not sufficient for two MAGs to be Markov equivalent. The following proposition by Spirtes and Richardson (1996) characterizes necessary and sufficient conditions for two MAGs to be Markov equivalent.

Proposition 43 (Spirtes and Richardson, 1996) *Two MAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent if and only if (i) they have the same skeleton, (ii) they have the same unshielded colliders, and (iii) if a path $\mathcal{P}$ is a discriminating path (Definition 7) for a vertex X in both MAGs, then X is a collider on $\mathcal{P}$ in $\mathcal{G}_1$ if and only if it is a collider on $\mathcal{P}$ in $\mathcal{G}_2$.*

Building upon this proposition, it suffices to learn the skeleton, unshielded colliders, and shielded colliders for which a discriminating path exists. Subsequently, complete orientation rules can be applied to achieve a maximally oriented (aka maximally informative) *partial ancestral graph* (PAG). We refer to Zhang (2008b) for complete orientation rules and further discussion.

5.7.1 Recover V-structures in MARVEL

Our goal is to recover the v-structures. Note that a v-structure comprises a pair of co-parents and a common child of them. As such, we first describe how to identify the pairs of variables that are co-parents.

Since each variable is identified as removable in exactly one iteration of MARVEL, we can identify the co-parents of a variable at the iteration where it gets removed. However, removing a variable does not preserve co-parent relationships. Indeed, if Y, Z are co-parents in a DAG $\mathcal{G}$, where X is a removable variable, then either Y, Z are still co-parents of each other in $\mathcal{G}[\mathbf{V}\backslash\{X\}]$, or $Y \in \mathrm{Mb}_{\mathcal{G}}(Z)\backslash\mathrm{Mb}_{\mathcal{G}[\mathbf{V}\backslash\{X\}]}(Z)$. The latter case happens when X is the only common child of Y and Z. Based on this observation, in order to identify all co-parent pairs, we first modify Algorithm 11, the procedure for updating Markov boundaries, so that whenever a pair of variables (Y, Z) are removed from each other's Markov boundary in lines 7 and 8, this pair is marked as co-parents of each other in the final graph. Furthermore,

the separating set for this pair, namely $\mathrm{Mb}_{\mathbf{W}\setminus\{X\}}(Y)$, is stored as $\mathbf{S}_{YZ}$. The rest of the co-parents and their corresponding separating sets are discovered in line 13 of Algorithm 4, which consists of an application of Lemma 27. During this step, the Markov boundary of X is partitioned into $\mathrm{Mb}_{\mathbf{V}_i}(X) = Ne_{\mathbf{V}_i}(X) \sqcup \Lambda_{\mathbf{V}_i}(X)$ through finding a separating set $\mathbf{S}_{XY}$ for every variable $Y \in \Lambda_{\mathbf{V}_i}(X)$. These separating sets are stored for every pair of co-parents (X, Y). At the end of the skeleton discovery phase, the set of v-structures can be identified based on the following lemma.

Lemma 44 (Verma and Pearl, 1991) *Let X, Y, Z be three arbitrary vertices of DAG $\mathcal{G}$. These vertices form a v-structure in $\mathcal{G}$, i.e., $X \to Z \leftarrow Y$ if and only if all of the following hold:*

$$Y \in \Lambda_{\mathcal{G}}(X), \quad Z \in Ne_{\mathcal{G}}(X) \cap Ne_{\mathcal{G}}(Y), \quad Z \notin \mathbf{S}_{XY}.$$

Accordingly, Lemma 44 can be integrated into MARVEL to identify the v-structures.

5.7.2 Recover V-structures in RSL

In analogy to MARVEL, it suffices to identify the v-structures. Also, the v-structures that are not preserved due to vertex removals can be identified and oriented through modifying Algorithm 11, just as described in the case of MARVEL. Herein, we present the procedure for identifying the v-structures $\mathrm{VS}_{\mathcal{G}}(X)$ for a removable variable X. We describe this procedure for RSL_D and RSL_ω separately.

RSL_D. In the case of diamond-free graphs, if X is a removable variable and $Y \in \Lambda_{\mathcal{G}}(X)$, then X and Y have a *unique* common child. Indeed, Proposition 40 reveals not only the co-parents of a removable variable X but also the unique common child of X and each of its co-parents. Accordingly, to identify the v-structures $\mathrm{VS}_{\mathcal{G}}(X)$ during RSL_D, it suffices to modify the **FindNeighbors** subroutine as follows. Each time a variable $Z \in \mathrm{Mb}_{\mathcal{G}}(X)\setminus\{Y\}$ satisfies Equation (7) in line 4 of **FindNeighbors**, the edges are oriented as $X \to Z$ and $Y \to Z$, since Z is the unique common child of X and Y.

RSL_ω. Analogous to RSL_D, we can exploit the procedure for finding the neighbors to further identify the v-structures. In particular, Proposition 37 identifies a unique set $\mathbf{S}$ as the common children of X and Y, for any Y that is a co-parent of X. Once such a set $\mathbf{S}$ is found in line 4 of **FindNeighbors** in RSL_ω (see line 12 of Algorithm 7) for a removable variable X_j and $Y \in \mathrm{Mb}_{\mathcal{G}}(X_j)$, it suffices to orient every edge $U - V$ such that $U \in \{X_j, Y\}$ and $V \in \mathbf{S}$ as $U \to V$.

6. Complexity and Completeness Analysis

In this section, we discuss the complexity and completeness of various causal discovery methods, with a particular emphasis on our proposed recursive approaches. This analysis is crucial for understanding the efficiency and reliability of these methods in practical applications. In addition, we delve into the theoretical limits of these algorithms by providing lower bounds for the complexity of constraint-based algorithms in learning DAGs and MAGs.

6.1 Lower Bound

We introduce two fundamental theorems: Theorem 45, which establishes a lower bound for the complexity of constraint-based algorithms in learning DAGs, and Theorem 46, which does the same for MAGs. These theorems are instrumental in quantifying the theoretical limits of constraint-based methods.

Theorem 45 (Lower bound for DAGs) *For any positive integers n and $1 \leq c \leq n$, there exists a DAG $\mathcal{G}$ with n vertices and $\Delta_{in}(\mathcal{G}) = c$ such that the number of d-separations of the form $(X \perp\!\!\!\perp Y|\mathbf{Z})_{\mathcal{G}}$ required by any constraint-based algorithm to learn the skeleton of $\mathcal{G}$ is lower bounded by*

$$\Omega(n^2 + n\Delta_{in}(\mathcal{G})2^{\Delta_{in}(\mathcal{G})}). \tag{11}$$

Theorem 45 determines the hardness of causal discovery under causal sufficiency parameterized based on $\Delta_{in}(\mathcal{G})$ rather than other graph parameters such as $\Delta(\mathcal{G})$ or $\alpha(\mathcal{G})$.

Theorem 46 (Lower bound for MAGs) *For any positive integers n and $1 \leq c \leq n$, there exists a MAG $\mathcal{G}$ with n vertices and $\Delta^{+}_{in}(\mathcal{G}) = c$ such that the number of m-separations of the form $(X \perp\!\!\!\perp Y|\mathbf{Z})_{\mathcal{G}}$ required by any constraint-based algorithm to learn the skeleton of $\mathcal{G}$ is lower bounded by*

$$\Omega(n^2 + n\Delta^{+}_{in}(\mathcal{G})2^{\Delta^{+}_{in}(\mathcal{G})}). \tag{12}$$

Theorem 46 extends the complexity analysis to the setting of MAGs, addressing scenarios in causal discovery where the assumption of causal sufficiency does not hold. It similarly parameterizes the hardness of the problem based on $\Delta^{+}_{in}(\mathcal{G})$, which is an extension of Δ_{in} for MAGs (Definition 3).

While our analysis focuses on the computational complexity of constraint-based causal discovery algorithms, we also acknowledge the complementary line of research on sample complexity and methods addressing limited data scenarios in causal discovery. Notably, Ghoshal and Honorio (2017); Gao et al. (2022); Jamshidi et al. (2024) have established lower bounds on the number of samples required to recover causal graphs under various assumptions and settings. Additionally, Cui et al. (2022) propose methods to improve constraint-based causal discovery under insufficient data by enhancing the robustness of statistical tests.

6.2 Completeness Analysis and Achievable Bounds of RCD Methods

We present completeness guarantees and upper bounds on the number of performed CI tests by MARVEL, L-MARVEL, RSL, and ROL, as implemented in Section 5.

Recall that our algorithms take as input Data($\mathbf{V}$), a set of i.i.d. samples from the observational distribution $P_{\mathbf{V}}$. In the propositions that follow, we provide asymptotic guarantees of correctness for our methods under the assumption that Data($\mathbf{V}$) is sufficiently large to accurately recover the CI relations present in $P_{\mathbf{V}}$. Additionally, as outlined in Section 2, we assume that the true underlying graph encodes the CI relations of the observational distribution, as stated in Equation (3). It should be noted that while our methods are also compatible with weaker notions of faithfulness, we focus on this primary assumption for simplicity in our presentation.

Proposition 47 (Completeness and complexity of MARVEL) *Under the assumption of causal sufficiency, MARVEL, as implemented in Algorithm 4, correctly returns the skeleton of DAG $\mathcal{G}$ by performing at most*

$$\binom{n}{2} + n\binom{\Delta_{in}(\mathcal{G})}{2} + \frac{n}{2}\Delta_{in}(\mathcal{G})(1 + 0.45\Delta_{in}(\mathcal{G}))2^{\Delta_{in}(\mathcal{G})} = \mathcal{O}\big(n^2 + n\Delta_{in}(\mathcal{G})^2 2^{\Delta_{in}(\mathcal{G})}\big) \quad (13)$$

unique CI tests, and at most $\binom{n}{2}2^{\Delta_{in}(\mathcal{G})-1}$ duplicate CI tests in the worst case[6]*.*

Corollary 48 *If $\Delta_{in}(\mathcal{G}) \leq c \log n$, MARVEL uses at most $\mathcal{O}(n^2 + n^{c+1}\log^2 n)$ unique CI tests in the worst case, which is polynomial in the number of variables.*

As n gets larger, if DAG $\mathcal{G}$ has a constant value of $\Delta_{in}(\mathcal{G})$, or more generally $\Delta_{in}(\mathcal{G}) \leq (1-\epsilon)\log n$, where $\epsilon > 0$, then both the achievable upper bound of MARVEL in (13) and the lower bound in (11) are quadratic in n. For larger values of $\Delta_{in}(\mathcal{G})$, the second terms in these equations become dominant. In this case, the upper bound of MARVEL differs from the lower bound only by a factor of $\Delta_{in}(\mathcal{G})$. This demonstrates that under causal sufficiency and without any additional information, MARVEL has a worst-case performance that nearly matches the lower bound.

Proposition 49 (Completeness and complexity of L-MARVEL) *L-MARVEL, as implemented in Algorithm 6, correctly returns the skeleton of MAG $\mathcal{G}_{\mathbf{V}}$ by performing at most*

$$\mathcal{O}\big(n^2 + n\Delta^+_{in}(\mathcal{G}_{\mathbf{V}})^2 2^{\Delta^+_{in}(\mathcal{G}_{\mathbf{V}})}\big)$$

number of CI tests.

Similar to our previous argument for MARVEL, Proposition 49 shows that the complexity of L-MARVEL aligns closely with the lower bound for MAGs in Theorem 46, demonstrating its near-optimal performance in causal models with latent variables.

Proposition 50 (Completeness and complexity of RSL$_\omega$) *Under causal sufficiency, if $\omega(\mathcal{G}) \leq m$, then RSL$_\omega$, as implemented in Algorithm 7, correctly returns the skeleton of DAG $\mathcal{G}$ by performing at most*

$$\mathcal{O}\big(n^2 + n\Delta_{in}(\mathcal{G})^{m+1}\big)$$

number of CI tests.

Proposition 50 marks a pivotal development in causal discovery for DAGs with bounded *TreeWidth*. Recent studies including Korhonen and Parviainen (2013); Nie et al. (2014); Ramaswamy and Szeider (2021), have emphasized the importance of algorithms tailored for scenarios where an upper bound on the TreeWidth of the causal graph is given as side information. A bound on TreeWidth is more restrictive than a bounded clique number, as indicated by the following inequality.

$$\omega(\mathcal{G}) \leq \text{TreeWidth}(\mathcal{G}) + 1$$

6. Duplicate CI tests can be completely eliminated through efficient use of memory. This aspect is omitted here for simplicity and readability.

As such, our RSL_ω algorithm is also applicable to causal discovery in DAGs with bounded TreeWidth. Notably, while existing exact discovery algorithms for these networks demonstrate exponential complexity, RSL_ω maintains a polynomial complexity. This indicates that when the TreeWidth is constant, causal discovery is no longer NP-hard and can be solved in polynomial time.

Proposition 51 (Completeness and complexity of RSL_D) *Under causal sufficiency and if $\mathcal{G}$ is a diamond-free DAG, then RSL_D, as implemented in Algorithm 9, correctly returns the skeleton of DAG $\mathcal{G}$ by performing at most*

$$\mathcal{O}\big(n^2 + n\Delta_{in}(\mathcal{G})^3\big) \tag{14}$$

number of CI tests.

Remark 52 *Even if DAG $\mathcal{G}$ has diamonds, RSL_D correctly recovers all the existing edges with possibly extra edges, i.e., RSL_D has no false negative.*

RSL_D is the fastest among our proposed recursive methods. This can also be seen by the upper bound in Equation (14), where the number of CI tests stays tractable for graphs with more than 1000 variables.

Proposition 53 (Complexity of ROL_{HC}) *ROL_{HC}, as implemented in Algorithm 10, performs at most*

$$\mathcal{O}\big(\text{maxIter} \times n^3\big) \tag{15}$$

number of CI tests, excluding the initialization step in line 2.

Note that the upper bound in Equation (15) may vary depending on the initialization step in line 2 and the choice of the *FindNeighbors* function. Also, here we are assuming that maxSwap is a constant.

Proposition 54 (Completeness and complexity of ROL_{VI}) *According to the introduced RL setting in Section 5.4.2, ROL_{VI} finds the optimal policy by performing at most $\mathcal{O}(n^2 2^n)$ number of CI tests.*

Note that the bound in Proposition 54 is much lower than $\mathcal{O}(n!)$ for iterating over all orders.

6.3 Comparison

In this part, we present a comparative analysis of various causal discovery methods, including our proposed algorithms. The summary of this comparison is presented in Table 3, which categorizes each algorithm based on its assumptions, completeness guarantees, and computational complexity in terms of the number of CI tests required. The last two rows present the lower bounds for complexity under causal sufficiency and in the absence of it, as established in Section 6.1.

A critical observation from this analysis is that the upper bounds on the complexity of our algorithms (MARVEL, L-MARVEL, RSL, and ROL) are significantly more efficient compared to the others. This efficiency in DAGs is largely due to the following inequality.

$$\Delta_{in}(\mathcal{G}) \leq \Delta(\mathcal{G}) \leq \alpha(\mathcal{G})$$

Algorithm	Assumptions		Completeness	#CI tests
	Causal sufficiency	Other		
MARVEL	YES	-	YES	$\mathcal{O}(n^2 + n\Delta_{in}^2 2^{\Delta_{in}})$
L-MARVEL	NO	-	YES	$\mathcal{O}(n^2 + n(\Delta_{in}^+)^2 2^{\Delta_{in}^+})$
RSL_ω	YES	$\omega(\mathcal{G}) \leq m$	YES	$\mathcal{O}(n^2 + n\Delta_{in}^{m+1})$
RSL_D	YES	Diamond-free	YES	$\mathcal{O}(n^2 + n\Delta_{in}^3)$
ROL_{HC}	NO	-	NO	$\mathcal{O}(\text{maxIter} \times n^3)$
ROL_{VI}	NO	-	YES	$\mathcal{O}(n^2 2^n)$
ROL_{PG}	NO	-	NO	N/A
PC	YES	-	YES	$\mathcal{O}(n^\Delta)$
GS	YES	-	YES	$\mathcal{O}(n^2 + n\alpha^2 2^\alpha)$
MMPC	YES	-	YES	$\mathcal{O}(n^2 2^\alpha)$
CS	YES	-	YES	$\mathcal{O}(n^2 2^\alpha)$
FCI	NO	-	YES	N/A
RFCI	NO	-	NO	N/A
FCI+	NO	-	NO	$\mathcal{O}(n^{2(\Delta+2)})$
MBCS*	NO	-	YES	$\mathcal{O}(n^2 2^\alpha)$
Lower Bound	YES	-	YES	$\Omega(n^2 + n\Delta_{in} 2^{\Delta_{in}})$
Lower Bound	NO	-	YES	$\mathcal{O}(n^2 + n\Delta_{in}^+ 2^{\Delta_{in}^+})$

Table 3: Summary of the assumptions, guarantees, and the complexity of various causal discovery methods from observational data.

Additionally, in a DAG with a constant in-degree, Δ and α can grow linearly with the number of variables.

Corollary 55 *Under the assumption of causal sufficiency, RSL_D is the fastest among our proposed methods. In scenarios lacking causal sufficiency, L-MARVEL is fastest for sparse graphs, while ROL_{HC} outperforms in denser graphs.*

7. Related Work

Most of the causal discovery methods can be broadly categorized into constraint-based and score-based approaches. While constraint-based methods recover the structure that is consistent with conditional independence constraints, score-based methods opt for the graph that maximizes a specific score function. In this section, we present an overview of relevant works in the field, organizing them into these two main categories. Under the score-based category, we give special consideration to the so-called *ordering-based* methods, which rely on recovering a specific order among the variables to guide the causal discovery task. While we strive to provide a representative review of key works in causal discovery, we note that the vast body of literature includes many more studies and approaches. For a more comprehensive compilation and deeper discussion, we refer to surveys on causal discovery such as Kitson et al. (2023), Hasan et al. (2023), Vowels et al. (2022), Glymour et al. (2019), and Mooij et al. (2016).

7.1 Constraint-Based

PC algorithm (Spirtes et al., 2000), widely used for causal discovery, stands as a foundational approach for this task using observational data, laying the groundwork for most of the subsequent developments in constraint-based approaches. Notably, acknowledging the high computational cost of PC, Le et al. (2016) provided a parallel computing framework for it. Colombo et al. (2014) drew attention to the fact that under potentially erroneous conditional independence tests, the results obtained by the PC algorithm may depend on the order in which these tests were conducted. Following this observation, they introduced PC-stable, which offers an order-independent approach to causal discovery. This enhances the PC's robustness with respect to uncertainties over the order of variables. Further, RPC (Harris and Drton, 2013) was introduced as a relaxation of the PC algorithm to handle instances where strict adherence to conditional independence tests may not be possible.

PC and its derivatives work under causal sufficiency. In extending the scope of constraint-based methods, Spirtes et al. (1995) introduced the fast causal inference algorithm (FCI), which accommodates latent variables and selection bias. Although foundational, FCI faced challenges with incomplete edge orientation rules. Zhang (2008b) later augmented further orientation rules to ensure completeness of its output. FCI also suffers from an intractable computational complexity in moderate to high dimensional causal discovery tasks. Focusing on learning high-dimensional causal graphs, Colombo et al. (2012) introduced RFCI, which performs only a subset of the conditional independence tests that FCI requires. Faster computations came at the cost of not being *complete*, in the sense that the recovered graph may contain extra edges in general.

Closest to our approach are the works by Margaritis and Thrun (1999), Pellet and Elisseeff (2008a), and Pellet and Elisseeff (2008b), which put forward the idea of using Markov boundary information to guide causal discovery. The grow-shrink (GS) algorithm (Margaritis and Thrun, 1999) was originally devised as a method to infer Markov boundaries. The authors augmented GS with further steps to make it capable of recovering the causal structure. Pellet and Elisseeff (2008b) proposed the use of total conditioning (TC) to infer the Markov boundary information and recover the causal structure afterward. Pellet and Elisseeff (2008a) also generalized the same ideas to handle causal discovery in the presence of latent variables and selection bias.

Along a separate path, CPC (Ramsey et al., 2006) introduced the concept of adjacency-faithfulness, building a conservative framework for constraint-based causal discovery. KCL (Sun et al., 2007) presented a kernel-based causal discovery algorithm, extending the scope of applicability of causal discovery algorithms to non-linearly related structural models through the use of kernel methods. On the same note, the kernel-based conditional independence test (KCI-test) introduced by Zhang (2008a) further enriches the toolkit for assessing conditional independence through kernel methods. ION (Danks et al., 2008) focused on integrating observational data with narrative information to refine causal relationships.

7.2 Score-Based

Score-based methods provide an alternative approach to constraint-based methods, emphasizing the optimization of a score function to identify the most likely causal graph. The foundation for this line of research was laid by early work in Bayesian statistics, as well as

the development of graphical models (Pearl, 1988). A notable contribution was made by Heckerman et al. (1995), which put forward the idea of integrating the prior beliefs and statistical data through a Bayesian approach to causal discovery. The authors reviewed certain heuristic algorithms to search for the graphical structure maximizing their scoring function. Greedy equivalence search (GES) algorithm, introduced by Chickering (2002), represents another significant advancement in the field. GES employs a step-wise greedy search strategy to iteratively refine the graph structure, aiming for maximizing the Bayesian information criterion (Raftery, 1995; Geiger and Heckerman, 1994).

GES has been influential in guiding the subsequent advancements in score-based causal discovery. An extension was introduced by Bühlmann et al. (2014), which diverges from GES by decoupling the order search among variables and edge selection in the DAG from each other. While the variable order search is carried out through a non-regularized maximum likelihood estimation, sparse regression techniques are used for edge selection. The method developed by Bühlmann et al. (2014) is valid for additive models. Another extension is the fast greedy equivalence search (FGES) (Ramsey et al., 2017). FGES builds upon GES by introducing two modifications to increase the speed of the search in order to adapt it for high-dimensional causal models.

Beyond greedy search approaches, there is also literature based on coordinate descent for optimizing the score function. Fu and Zhou (2013) utilizes an ℓ_1-regularized likelihood approach and a block-wise coordinate descent to estimate the causal structure. Gu et al. (2019) model the conditional density of a variable given its parents by multi-logit regression, employing a group norm penalty to obtain a sparse graph. Aragam et al. (2015) reduce causal discovery to a series of neighborhood regressions under suitable assumptions.

A broad range of recent research on score-based causal discovery has focused on methods based on continuous optimization. The most influential work in this direction was DAGs with NO TEARS (Zheng et al., 2018), reformulating the combinatorial problem of causal search into a continuous optimization problem. Goudet et al. (2018) introduced CGNN, which uses neural networks to learn the functional mappings between variables and incorporates a hill-climbing search algorithm for the optimization. In order to address the computational costs of CGNN, Kalainathan et al. (2022) presented SAM. Recently, several methods have been introduced to further tackle the time-consuming nature of these methods. GraN-DAG (Lachapelle et al., 2019) improved upon NO TEARS by leveraging neural networks to model the nonlinear relationships while maintaining computational efficiency through gradient-based techniques. SparseRC (Misiakos et al., 2024) also addressed the scalability issue by adopting a sparse regularization framework. DAG-NoCurl (Yu et al., 2021) presented another promising approach, reducing computational costs without explicitly enforcing acyclicity constraints. Bhattacharya et al. (2021) extended the approach to graphs with hidden variables. Other notable extensions include but are not limited to DAGMA (Bello et al., 2022), TOPO (Deng et al., 2023), No FEARS (Wei et al., 2020), DAG-GNN (Yu et al., 2019), Graph AutoEncoder (Ng et al., 2019), NO BEARS (Lee et al., 2019), and DYNOTEARS (Pamfil et al., 2020).

Bayesian Approaches. These methods represent another prominent class within score-based approaches, emphasizing the integration of prior knowledge with observed data to learn causal structures. The foundational work in this area was the work by Heckerman et al. (1995), which proposed methods that merge expert knowledge and statistical data for

learning Bayesian networks. Building on this work, Friedman and Koller (2003) introduced a principled Bayesian approach to structure discovery in Bayesian networks. More recent works have advanced Bayesian approaches in different ways. Viinikka et al. (2020) and introduced a scalable Bayesian framework that leverages variational inference, making it computationally efficient and applicable to large-scale data sets. Zhang et al. (2024) employed a similar approach for active learning. Another significant contribution is from Lorch et al. (2021), which framed Bayesian structure learning as a flexible differentiable optimization problem. Deleu et al. (2022) proposed using generative flow networks to approximate the posterior distributions. Others have focused on making the posterior computations more tractable (Annadani et al., 2021; Hoang et al., 2024), as well as extending Bayesian approaches to latent confounder models (Ma et al., 2024), and active learning (Toth et al., 2022).

7.3 Ordering-Based

Despite the inherent difficulty of causal discovery (Chickering, 1996), finding the graphical structure that maximizes a scoring function becomes tractable when an ordering is postulated on the variables (Buntine, 1991; Cooper and Herskovits, 1992). Based on this observation, Teyssier and Koller (2005) proposed a search over the space of variable orderings, rather than the previously adopted search over the space of DAGs. After recovering the ordering among variables, Teyssier and Koller (2005) used an exhaustive search through all possible parent sets for each vertex. Improving on the latter, Schmidt et al. (2007) showed that this search can be well-approximated through ℓ_1 regularization. Another extension to the work by Teyssier and Koller (2005) was introduced by Scanagatta et al. (2015), who proposed anytime algorithms to circumvent the high costs of searching in the space of potential parent sets. Raskutti and Uhler (2018) introduced the sparsest permutation (SP) algorithm, which relaxes the common faithfulness assumption to a weaker assumption called *u-frugality*. On the same note, Lam et al. (2022) developed a class of permutation-based algorithms, namely GRaSP, which operate under weaker assumptions than faithfulness. Solus et al. (2021) were the first to provide consistency guarantees for a greedy permutation-based search algorithm, namely GSP. Bernstein et al. (2020) extended the scope of permutation-based methods to causal structures with latent variables. Ordering-based methods offer an alternative approach to causal discovery by postulating an ordering on the variables to reduce the search space. Recent approaches such as sortregress (Reisach et al., 2021) and HOST (Duong and Nguyen, 2023) have been proposed to tackle complex data structures, particularly for heterogeneous and heteroskedastic data.

7.4 Miscellaneous

While the focus of this paper is on methods for learning a causal graph based on observational data, it is essential to acknowledge other directions in causal discovery research. For example, Yu et al. (2023) introduces a novel approach for causal discovery in zero-inflated data, leveraging directed graphical models to enhance gene regulatory network analysis. Another example is Zhao et al. (2024), which introduces a neighborhood selection method for learning the structure of Gaussian functional graphical models for high-dimensional functional data, applicable to EEG and fMRI data. Furthermore, there has been a grow-

ing interest in causal discovery for cyclic graphs (Richardson, 1996a,b; Mooij et al., 2011; Sethuraman et al., 2023), as these models have implications for causal relationships that involve feedback loops and dynamic dependencies. It is noteworthy that cyclic structures pose additional challenges compared to acyclic graphs.

There has also been a growing interest in causal discovery for challenging data types. For instance, Huang et al. (2020) and Zhou et al. (2022) studied causal discovery from heterogeneous and non-stationary data. Günther et al. (2022) considered the problem of independence testing with heteroskedastic data. Several recent works investigated causal discovery under heteroskedastic noise models Duong and Nguyen (2023); Yin et al. (2024); Kikuchi (2022); Cai et al. (2020).

Another avenue of research has concentrated on causal discovery methods that leverage interventional data as well as observational data (Kocaoglu et al., 2019; Brouillard et al., 2020; Li et al., 2023). Provided access to interventional data, one can reduce the size of the equivalence class, resulting in a finer specification of the causal model. Some works consider causal discovery in an active manner, where experiments are designed explicitly to learn causal graphs (Hyttinen et al., 2013; Hauser and Bühlmann, 2014; Hu et al., 2014; Ghassami et al., 2017; Addanki et al., 2020; Mokhtarian et al., 2023b). This active approach involves strategically choosing interventions to gain the most informative data for learning the graphical structure.

Some other notable lines of research include causal discovery for temporal data (Entner and Hoyer, 2010; Assaad et al., 2022; Chu et al., 2014; Gong et al., 2023), and causal representation learning (Schölkopf et al., 2021; Wang and Jordan, 2021; Ahuja et al., 2023).

8. RCD: A Python Package for Recursive Causal Discovery

We have implemented the algorithms presented along with other necessary and auxiliary utility functions in our Python package, called RCD. Our implementation is available on GitHub with the following link:

`github.com/ban-epfl/rcd`

Additionally, you can find a detailed documentation of our package on the following website:

`rcdpackage.com`

Some of the key aspects of our package are highlighted in the following.

- **Simple installation:** RCD is available on `PyPI` for installation. Use the command

  ```
  pip install rcd
  ```

  to add it to your environment.

- **Lightweight dependencies:** RCD uses only four packages - `NetworkX`, `NumPy`, `Pandas`, and `SciPy`, all commonly used in causal discovery.

- **Well-documented:** We have written thorough documentation for each class and function in the Google Python documentation style, available in the website.

- **Readable code:** We used consistent naming schemes for functions and variables and added descriptive in-code comments for increased readability.

- **Efficient implementation:** Optimized for performance, we used optimal data structures and minimized loops and redundant CI tests, ensuring a lean and fast codebase.

8.1 Source Code Organization

The source code of the RCD package available on our GitHub repository is divided into four directories.

- **rcd**: It contains the implementation of the methods.

- **tests**: It contains unit tests in the framework of `pytest` that ensure the correctness of the implementation of each method.

- **examples**: It contains working demonstrations of each method.

- **docs**: It contains the configuration for `MkDocs`, which is responsible for generating our documentations site.

8.2 Method Implementation

Our proposed methods in the RCD package are implemented as Python modules:

`marvel`, `l_marvel`, `rsl_d`, `rsl_w`, `rol_hc`

RCD is designed with modularity at its core. Each module requires a CI testing function upon initialization and optionally accepts a Markov boundary matrix finding function. This design enables users to incorporate their own CI testing and Markov boundary-finding algorithms. If a Markov boundary-finding function is not given, our methods use a naive approach to find the initial Markov boundary matrix.

Each module has a primary public function that is intended to be used by the user, which is the `learn_and_get_skeleton` function. This function receives a Pandas Dataframe as input, which contains data samples for each variable, with each column representing a variable. The function then returns the learned skeleton as an undirected NetworkX graph.

Below is a small snippet showing the module corresponding to the RSL_D method, named `rsl_d`, being used to learn the skeleton corresponding to a data set using the Fisher-Z test.

```
from rcd import rsl_d
from rcd.utilities.ci_tests import fisher_z

# run RSL-D on data with the Fisher Z test
ci_test = lambda x, y, z, data: fisher_z(x, y, z, data,
    significance_level=2 / n ** 2)
learned_skeleton = rsl_d.learn_and_get_skeleton(ci_test, data_df)
```

	Algorithm	Original Paper	Source Code
RCD alg.	RSL_D	Mokhtarian et al. (2022)	rcd package
	RSL_ω	Mokhtarian et al. (2022)	rcd package
	L-MARVEL	Akbari et al. (2021)	rcd package
	MARVEL	Mokhtarian et al. (2021)	rcd package
	ROL-HC	Mokhtarian et al. (2023a)	rcd package
Other CD alg.	PC	Spirtes et al. (2000)	causal-learn package
	FCI	Spirtes et al. (2000)	causal-learn package
	CD-NOD	Huang et al. (2020)	causal-learn package
	GRaSP	Lam et al. (2022)	causal-learn package
	GES	Chickering (2002)	causal-learn package
	fGES	Ramsey et al. (2016)	py-tetrad package
	NoCURL	Yu et al. (2021)	Paper's repository
	GOLEM	Ng et al. (2020)	gCastle package
	SparseRC	Misiakos et al. (2024)	Paper's repository

Table 4: The causal discovery algorithms used in our experiments.

9. Experiments

In this section, we present a series of simulations to compare the RCD algorithms with other widely used causal discovery algorithms for learning the skeleton of a causal graph. We study the effect of varying the number of variables, graph density, and sample size in both linear and non-linear settings. We also include experiments on real-world networks.

9.1 Experimental Setup

We provide an overview of the experimental setup, including the algorithms compared, data sets used, experimental environment, evaluation metrics, and methods for Markov boundary estimation.

9.1.1 Algorithms

We compare our proposed algorithms in the `rcd` package—namely, RSL_D, RSL_ω, L-MARVEL, MARVEL, and ROL_{HC}—with the following nine causal discovery algorithms: PC, FCI, CD-NOD, GRaSP, GES, fGES, SparseRC, GOLEM, and NoCURL. More information about these methods is provided in Table 4.

9.1.2 Data sets

For the ground-truth DAGs, we consider both synthetic random graphs generated from Erdős-Rényi (ER) models and real-world networks available at `https://www.bnlearn.com/`

`bnrepository`. For the synthetic data sets, we generate data under two settings: linear and non-linear.

Linear setting. We generate linear SEMs with Gaussian noises based on the generated DAGs. Details of the data generation process are as follows:

- For each graph, we assign edge weights sampled uniformly from the intervals $[-1.5, -1]$ and $[1, 1.5]$, allowing for both positive and negative dependencies.
- Gaussian noise with a standard deviation randomly chosen from $[0.7, 1.2]$ is added to each variable to introduce variability.
- The value of each variable is computed as a linear combination of its parent variables, weighted by the assigned edge weights, plus the corresponding noise term.

Non-linear setting. We initially generate samples as described in the linear setting. Then, we apply a non-linear transformation to the samples for each variable. Specifically, each variable is transformed by applying the cumulative distribution function (CDF) of its marginal distribution. This transformation guarantees that the marginal distribution of each variable is uniform on $[0, 1]$.

9.1.3 Experiment Environment

We ran all algorithms using Python 3.10, except for fGES, which was run using JPype to connect to Tetrad running on the Amazon Corretto 22 JDK. All simulations were conducted on a PC equipped with two Intel Xeon E5-2680 v3 CPUs, 256 GB of RAM, and running Ubuntu 20.04.4 LTS. We set a time limit of 100 seconds for each algorithm, after which the execution was terminated.

9.1.4 Metrics

In all experiments, we measure the process-time taken by each algorithm to learn and produce the skeleton from the given data, the average conditioning set size, the number of CI tests, and the F1 score when compared with the true skeleton. Note that the CI test-related metrics are only measured and reported for algorithms that utilize CI tests—namely, all RCD algorithms, PC, FCI, and CD-NOD.

9.1.5 Markov Boundary Estimation

As discussed in Section 5.5, several algorithms exist for discovering Markov boundaries. In our experiments, we use the Total Conditioning (TC) method (Pellet and Elisseeff, 2008b) to compute the Markov boundary of each variable. Since we are dealing with linear Gaussian models, we can efficiently evaluate the required conditional independencies using the precision matrix (the inverse of the covariance matrix). Specifically, we compute the partial correlations between variables from the precision matrix and apply Fisher's z-transform to perform the conditional independence tests. This approach allows us to compute all the Markov boundaries efficiently. For the significance level in the conditional independence tests, we set $\alpha = 2/n^2$ (Pellet and Elisseeff, 2008b).

9.2 Results

We now present the results of our experiments. For the linear SEMs, we consider four scenarios: varying the number of variables, varying the graph density, varying the sample size, and using real-world graphs. For the non-linear SEMs, we evaluate two scenarios: varying the number of variables and varying the sample size.

9.2.1 Linear SEMs: Varying the Number of Variables

We evaluate the performance of each algorithm as a function of the number of variables n. We investigate two regimes: sparse and dense.

- **Sparse Regime**: We generate ER graphs with edge probability $p = 1/n$, varying n from 10 to 500.
- **Dense Regime**: We generate ER graphs with edge probability $p = \log n/n$, varying n from 10 to 150.

For each value of n, we generate 20 DAGs, and for each DAG, we generate 10 data sets, each containing $50n$ samples. We present the results for the sparse regime in Figure 6, and for the dense regime in Figure 7.

In the sparse regime (Figure 6), our RCD algorithms are the fastest by far. The time plots (note the logarithmic y-axis) demonstrate that our methods are orders of magnitude faster than the other causal discovery algorithms. This efficiency is corroborated by the number of CI tests performed, which is significantly lower for our methods, confirming that the number of CI tests is a good proxy for the time complexity of constraint-based methods. In terms of accuracy, our methods achieve near-perfect F1 scores. While fGES and GRaSP also achieve high accuracy, they are much slower than our methods in sparse graphs.

In the dense regime (Figure 7), RSL_D and fGES are the fastest algorithms. However, fGES suffers from lower accuracy, with F1 scores dropping below 0.7, whereas our methods maintain F1 scores around 0.9. GRaSP achieves the best accuracy in dense graphs but is only scalable up to 40 variables in this setting. In contrast, RSL_D can easily be applied to graphs with up to 150 variables and potentially much larger graphs, demonstrating better scalability while maintaining high accuracy.

9.2.2 Linear SEMs: Varying the Graph Density

We run simulations on ER graphs with $n = 50$ variables and a sample size of 2500. The edge probability p varies from 0.02 to 0.2. For each value of p, we generate 10 DAGs, and for each DAG, we generate 10 data sets.

We present the results in Figure 8. These results confirm our previous observations. For smaller values of the edge probability (i.e., sparser graphs), our RCD algorithms are faster than the other methods while achieving similar accuracy. As the graphs become denser, the runtime of our methods increases, becoming comparable to some of the other algorithms. In the densest cases, fGES becomes faster than our methods by an order of magnitude. However, this speed comes at the cost of accuracy—our methods maintain much higher F1 scores compared to fGES in dense graphs. This demonstrates that while fGES is faster in

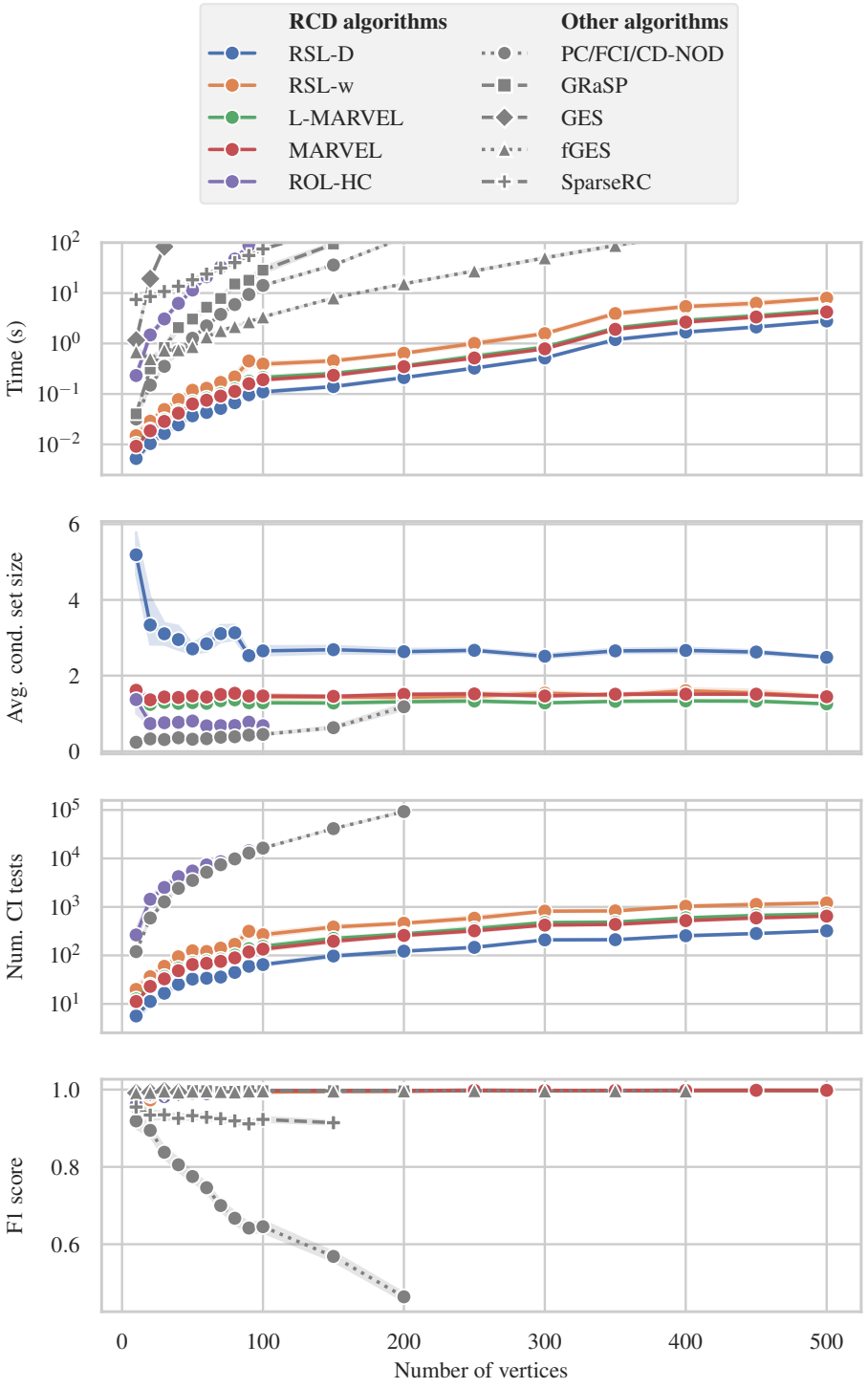


Figure 6: Time taken, average conditioning set size, number of CI tests, and F1 score of RCD algorithms and other CD algorithms on linear SEMs against the number of variables on the sparse ER data set.

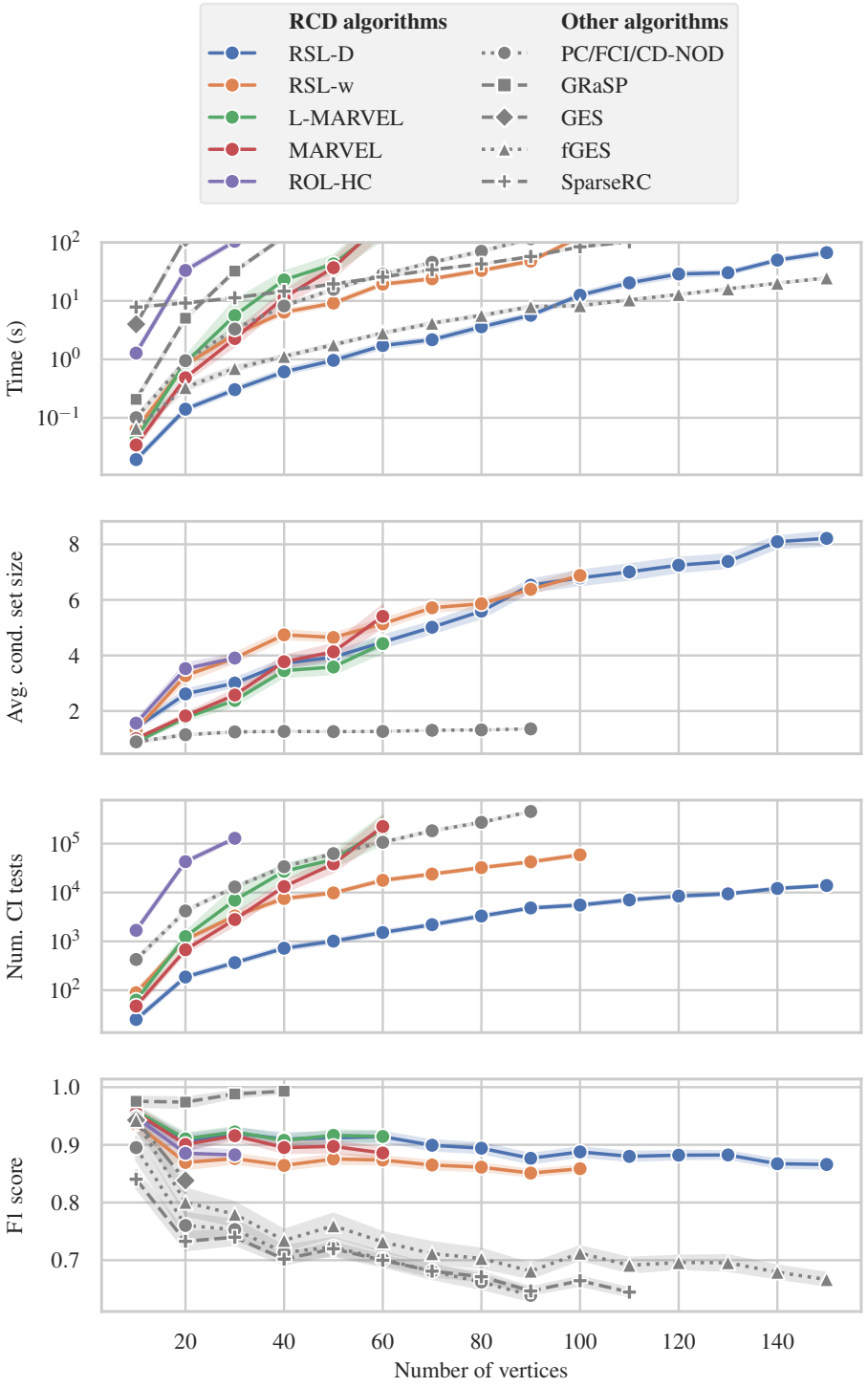


Figure 7: Time taken, average conditioning set size, number of CI tests, and F1 score of RCD algorithms and other CD algorithms on linear SEMs against the number of variables on the dense ER data set.

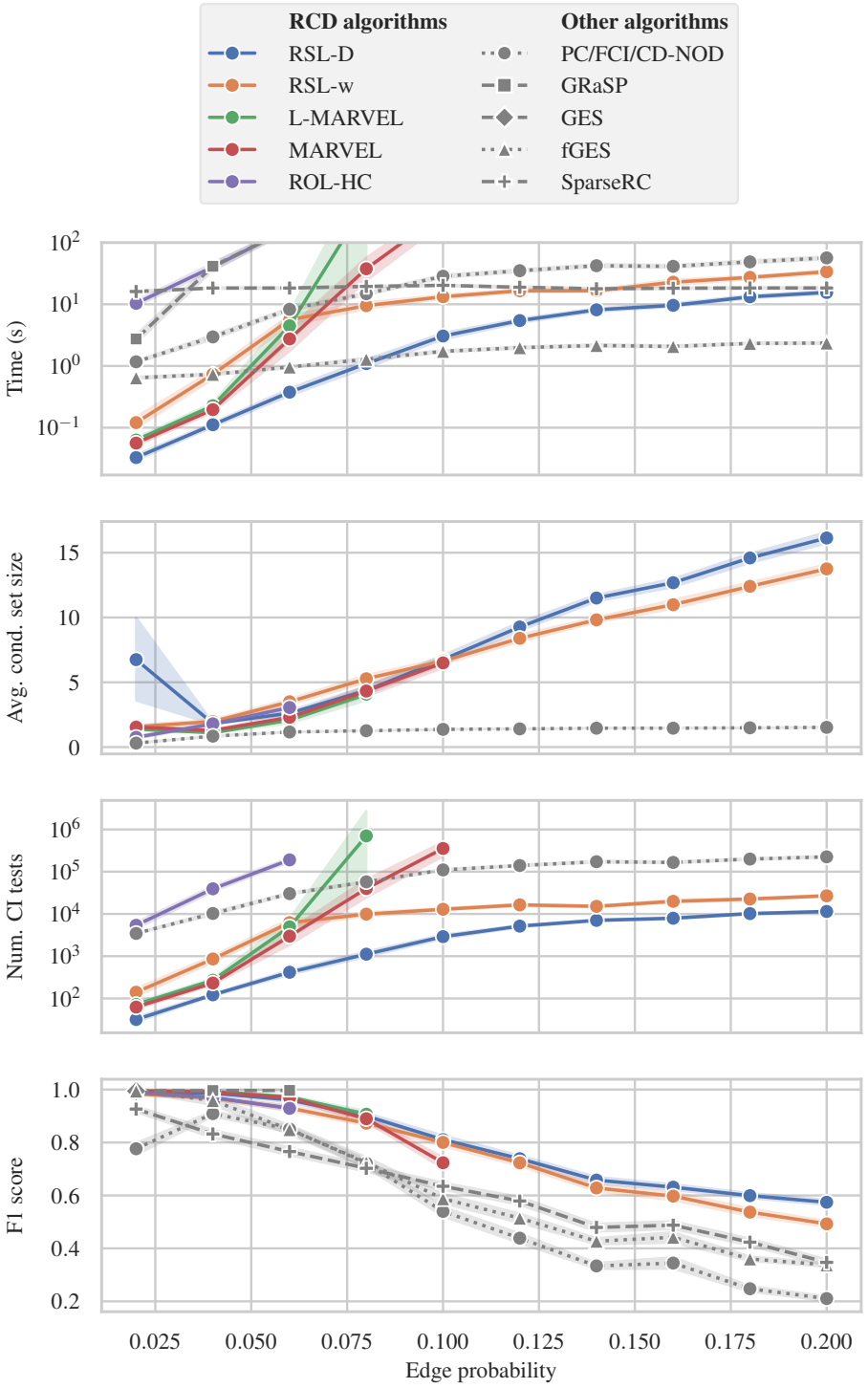


Figure 8: Time taken, average conditioning set size, number of CI tests, and F1 score of RCD algorithms and other CD algorithms on linear SEMs against the ER edge probability.

very dense graphs, our methods offer a better trade-off between speed and accuracy across different graph densities.

9.2.3 Linear SEMs: Varying the Sample Size

We run simulations on ER graphs with $n = 30$ variables and an edge probability of $p = 0.1$. We generate data sets with sample sizes ranging from 250 to 2000. For each sample size, we generate 10 DAGs and 10 data sets for each DAG.

We present the results in Figure 9. The figure shows that all algorithms are relatively insensitive to the number of samples. For sample sizes greater than 500 and up to 2000 samples, the performance of all algorithms remains stable in terms of both accuracy and runtime.

9.2.4 Linear SEMs: Real-World Graphs

We conduct experiments on 16 real-world graphs from the Bayesian Network Repository with the number of variables ranging from 8 to 724.[7] For each graph, we generate a data set following the data generation procedure described earlier. We present the results for graphs with fewer than 50 variables in Figure 10 and for those with more than 50 variables in Figure 11.

In almost all cases, our methods are the fastest, with RSL_D being the fastest in every instance. Regarding accuracy, our methods generally outperform others, with the difference being more significant in larger graphs (Figure 11). Notably, RSL_D not only has the best runtime but also achieves the highest accuracy across these 16 real-world graphs. This highlights that RSL_D is the best-performing method in this setting.

It is worth mentioning that some of these networks contain diamond structures, whereas RSL_D has theoretical guarantees when the graph is diamond-free. Despite this, RSL_D performs exceptionally well, demonstrating robustness to violations of its structural assumptions. This robustness is discussed further in Subsection 6.2.

9.2.5 Non-linear SEMs: Varying the Number of Variables

We evaluate the performance of each algorithm as a function of the number of variables n in non-linear SEMs. In this setting, we consider the dense regime by generating ER graphs with edge probability $p = \log n/n$, varying n from 10 to 100. For each value of n, we generate 20 DAGs, and for each DAG, we generate 10 data sets, each containing $50n$ samples. The non-linear SEMs are obtained by applying the CDF-based transformation to the generated data, ensuring that each variable follows a uniform distribution on $[0, 1]$.

Figure 12 presents the results in terms of time taken, average conditioning set size, number of CI tests, and F1 score. Our proposed methods outperform the other algorithms in terms of both time complexity and accuracy. Notably, GRaSP no longer performs the best in terms of accuracy but is now equally as good as our approaches, while much slower.

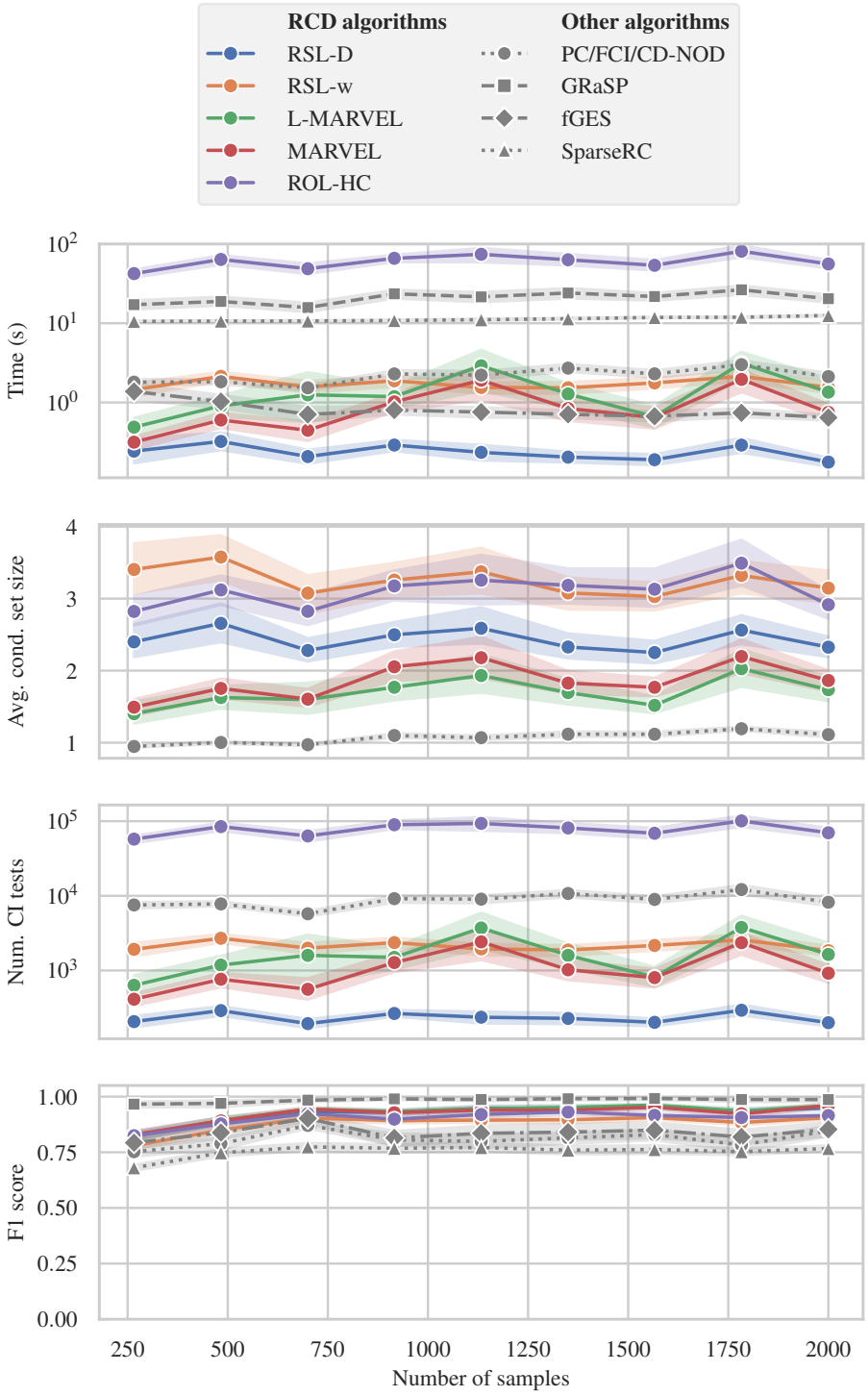


Figure 9: Time taken, average conditioning set size, number of CI tests, and F1 score of RCD algorithms and other CD algorithms on linear SEMs against the number of samples.

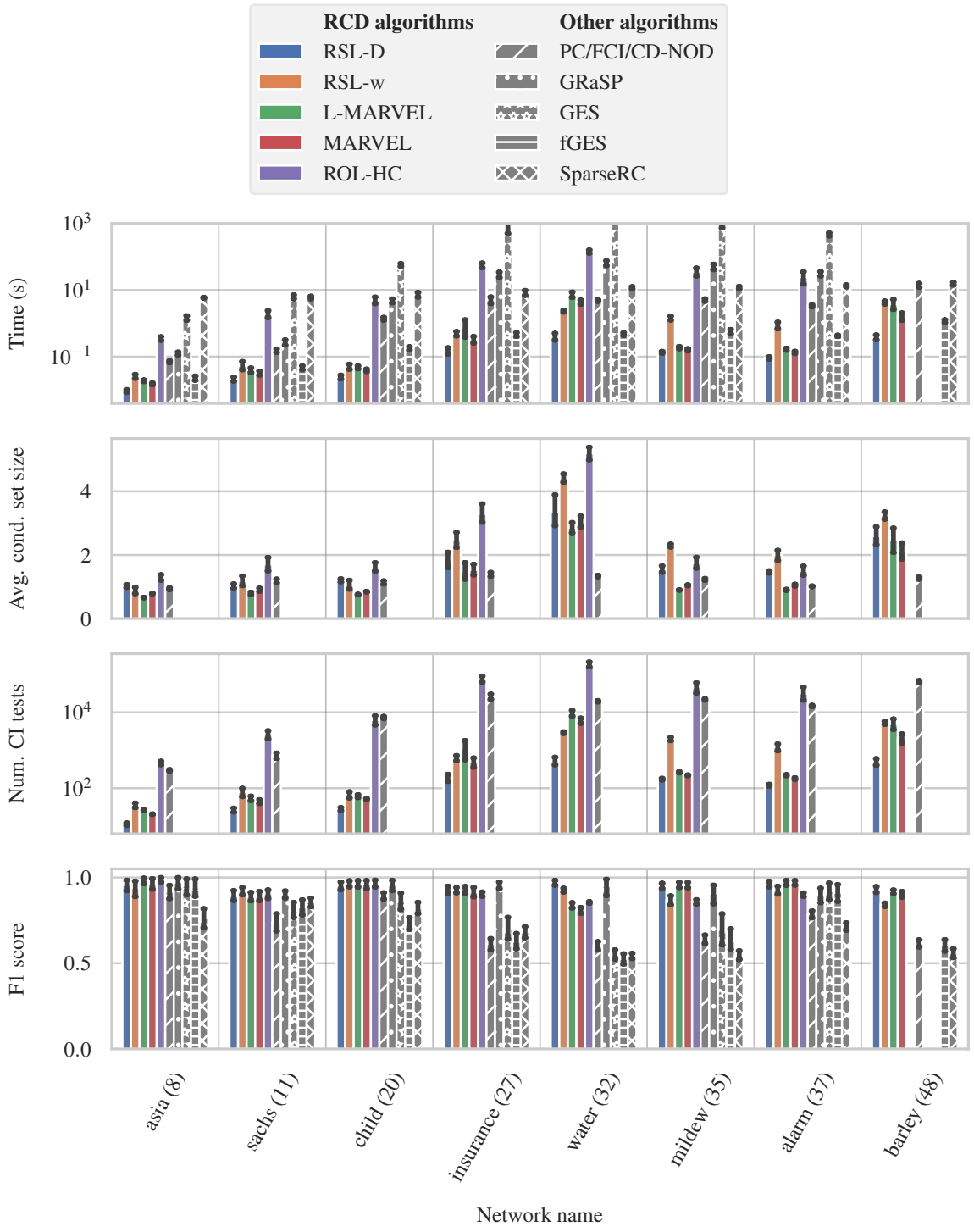


Figure 10: Time taken, average conditioning set size, number of CI tests, and F1 score of RCD algorithms and other CD algorithms on small BN repository graphs.

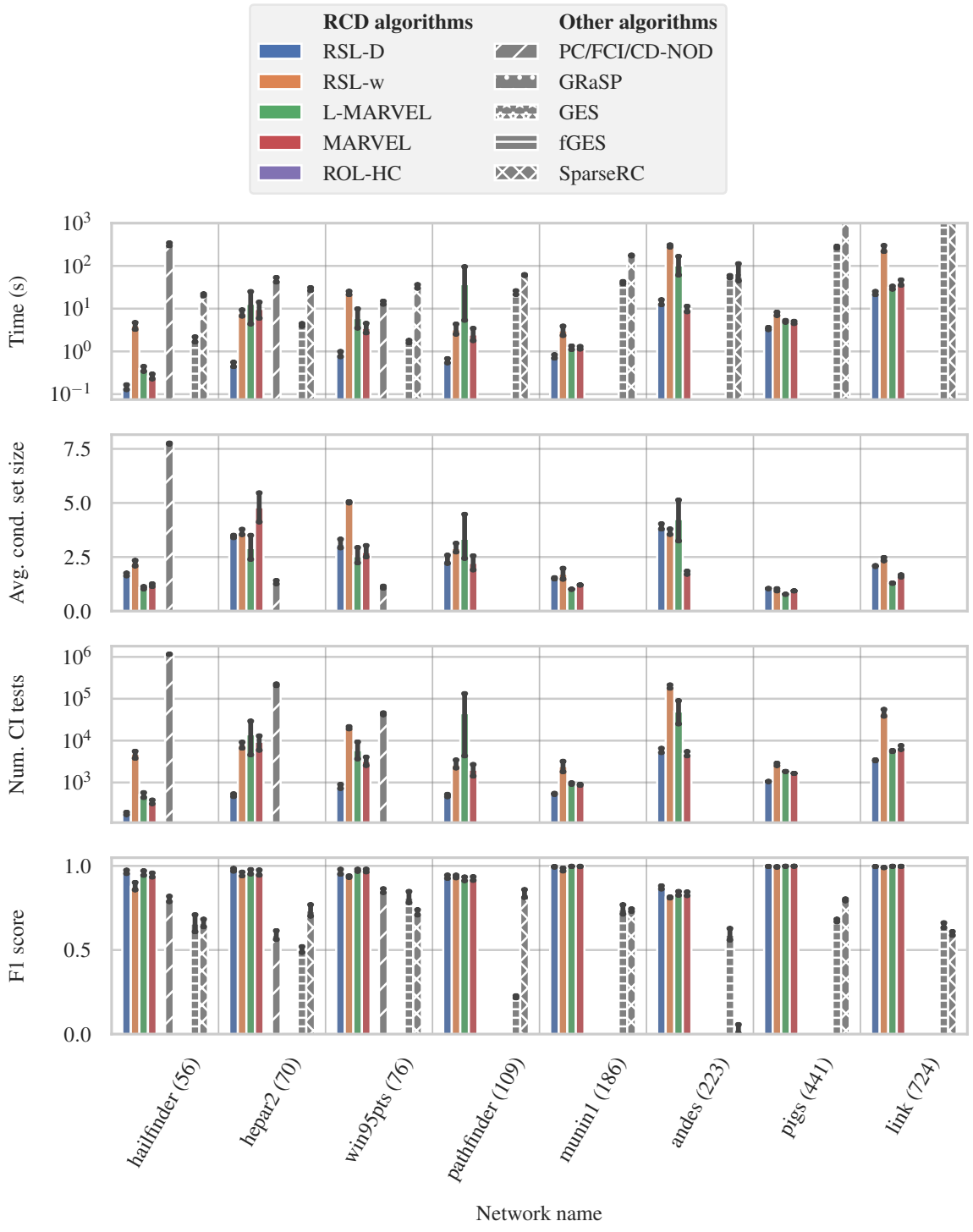


Figure 11: Time taken, average conditioning set size, number of CI tests, and F1 score of RCD algorithms and other CD algorithms on large BN repository graphs.

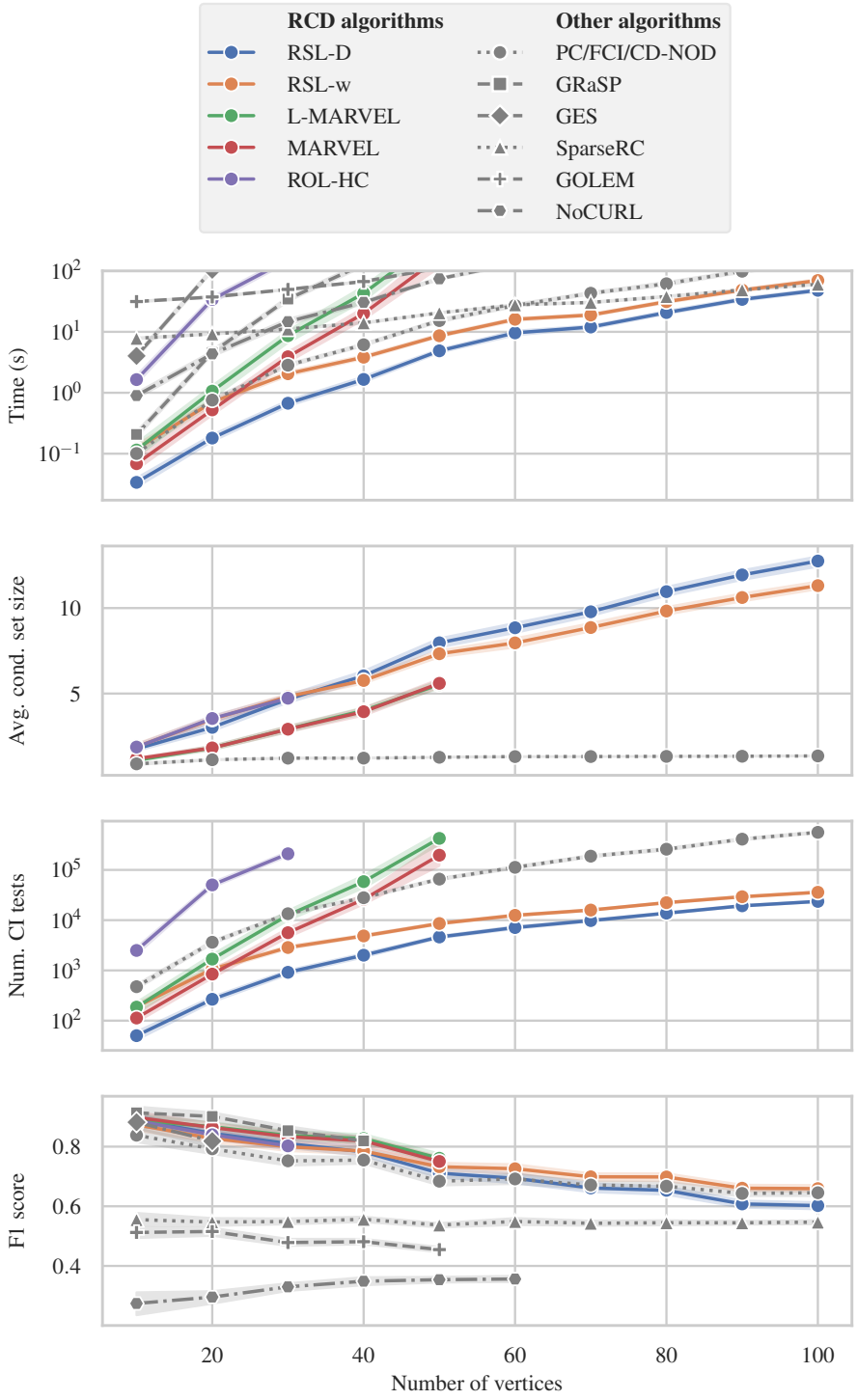


Figure 12: Time taken, average conditioning set size, number of CI tests, and F1 score of RCD algorithms and other CD algorithms on non-linear SEMs against the number of variables.

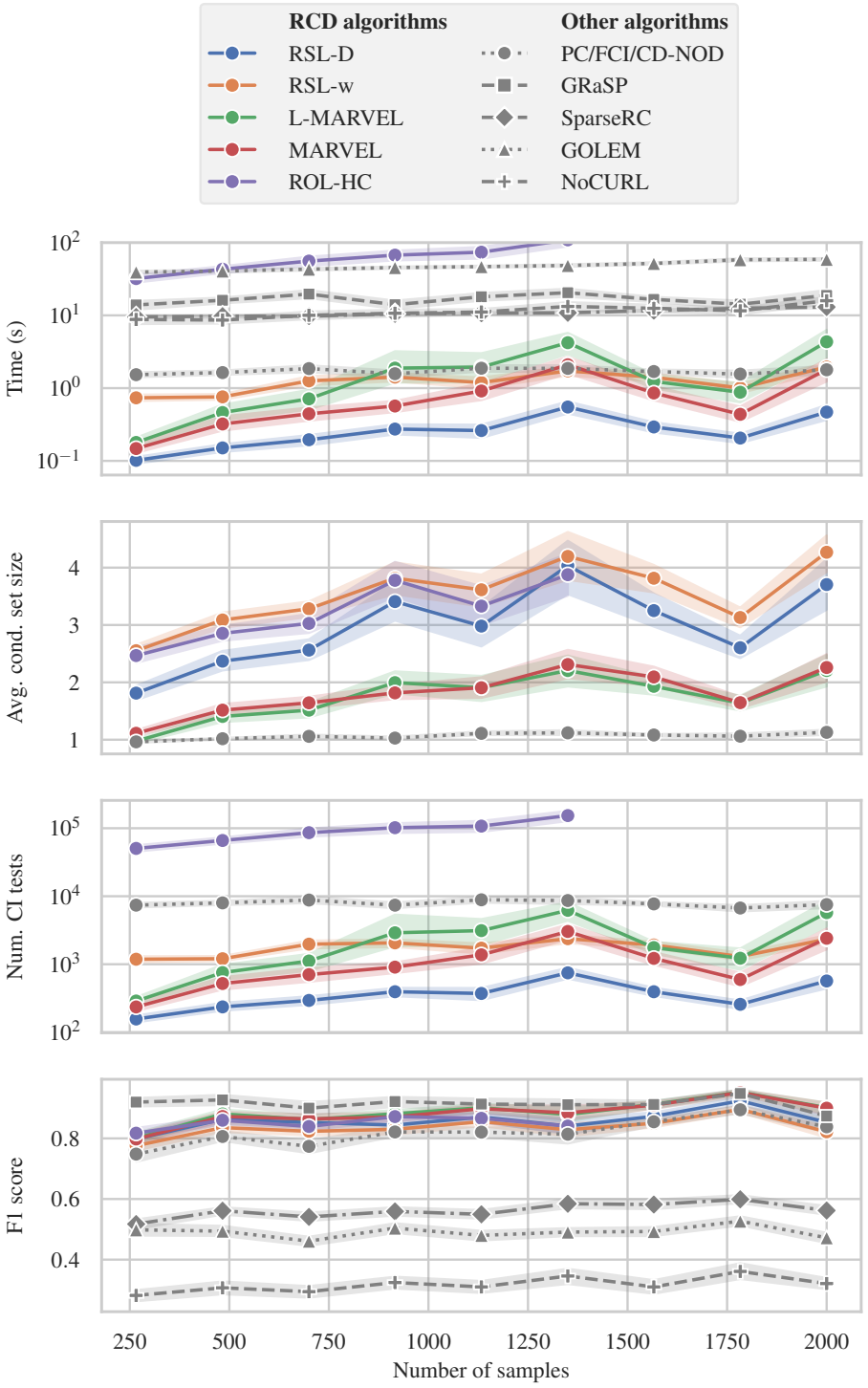


Figure 13: Time taken, average conditioning set size, number of CI tests, and F1 score of RCD algorithms and other CD algorithms on non-linear SEMs against the number of samples.

9.2.6 Non-linear SEMs: Varying the Sample Size

We run simulations on ER graphs with a fixed number of variables (set to $n = 30$) and edge probability $p = 0.1$, generating non-linear SEMs as described above. Data sets are generated with sample sizes ranging from 250 to 2000. For each sample size, we generate 10 DAGs and 10 data sets.

The performance metrics are reported in Figure 13. Similar to the linear case, all algorithms are relatively insensitive to the number of samples, and our approaches continue to be the fastest.

10. Conclusion, Limitations, and Future Work

In this work, we developed a comprehensive framework for recursive causal discovery, derived from our previous publications ($\mathfrak{R}_1$-$\mathfrak{R}_4$), refined with additional details and enhancements. The main results discussed in this paper are summarized in Table 5.

Our methodology revolves around identifying removable variables, learning their neighbors, discarding them, and then recursively learning the graph of the remaining variables. Through this iterative process, we have significantly reduced the number of performed CI tests, enhancing the computational efficiency and the accuracy of our methods. We further provided lower bounds on the complexity of constraint-based methods in the worst case and showed that our proposed methods almost match the lower bounds. Finally, we introduced `RCD`, an open-source Python package with documentation at rcdpackage.com and hosted on github.com/ban-epfl/rcd.

Limitations

While our framework offers significant advancements in recursive causal discovery, it shares a common limitation with all constraint-based methods: it relies on CI tests, which may not consistently provide reliable results, especially in non-parametric models where such tests can be less robust. Additionally, our approaches leverage Markov boundaries in their recursive framework, allowing the application of any existing algorithm for computing Markov boundaries. Although we proposed methods that iteratively update the Markov boundaries at a low cost, the initial step may pose a computational bottleneck in large-scale applications.

Future Work

In the following, we discuss potential future work.

- As mentioned earlier, although our current approaches can learn graphs up to the order of 10^3 variables with conventional computational power, the initial computation of Markov boundaries poses a computational challenge for larger graphs. An influential direction for future work is the development of recursive causal discovery methods that do not depend on Markov boundary computations.

7. bnlearn.com/bnrepository

Result	Description	Source
Proposition 17	Only removables can get removed	$\mathfrak{R}_1$, $\mathfrak{R}_2$
Theorem 18	Graphical characterization of removables in DAGs	$\mathfrak{R}_1$
Theorem 19	Graphical characterization of removables in MAGs	$\mathfrak{R}_2$
Proposition 20	Removables exist	$\mathfrak{R}_1$, $\mathfrak{R}_2$
Proposition 21	Removables have small Mb size	$\mathfrak{R}_1$, $\mathfrak{R}_2$
Proposition 22	Removables are invariant in a MEC	
Proposition 25	r-orders are invariant across MEC	$\mathfrak{R}_4$
Proposition 26	r-orders include c-orders	$\mathfrak{R}_4$
Lemma 28	Finding v-structures	$\mathfrak{R}_1$
Lemma 29	Testing condition 1 of removability in DAGs	$\mathfrak{R}_1$
Lemma 30	Testing condition 2 of removability in DAGs	$\mathfrak{R}_1$
Theorem 32	Testing removability in MAGs	$\mathfrak{R}_2$
Proposition 34	Avoiding duplicate CI tests in L-MARVEL	
Theorem 36	Removability test in RSL_ω	$\mathfrak{R}_3$
Proposition 37	Finding neighbors in RSL_ω	$\mathfrak{R}_3$
Theorem 39	Removability test in RSL_D	$\mathfrak{R}_3$
Proposition 40	Finding neighbors in RSL_D	$\mathfrak{R}_3$
Theorem 41	Consistency result of ROL's objective	$\mathfrak{R}_4$
Proposition 42	RSL_ω is verifiable	$\mathfrak{R}_3$
Theorem 45	Lower bound for DAGs	$\mathfrak{R}_1$
Theorem 46	Lower bound for MAGs	$\mathfrak{R}_2$
Proposition 47	Completeness and complexity of MARVEL	$\mathfrak{R}_1$
Proposition 49	Completeness and complexity of L-MARVEL	$\mathfrak{R}_2$
Proposition 50	Completeness and complexity of RSL_ω	$\mathfrak{R}_3$
Proposition 51	Completeness and complexity of RSL_D	$\mathfrak{R}_3$
Proposition 53	Complexity of ROL_{HC}	
Proposition 54	Completeness and complexity of ROL_{VI}	$\mathfrak{R}_4$

Table 5: Table of results.

- In exploring real-world applications of our package, one domain of particular interest is network biology, specifically for learning Gene Regulatory Networks (GRN). However, given the large number of genes in the human genome (approximately 20,000), applying our current methods directly might be challenging due to the dimensionality issues. While our methods can be used to analyze GRNs of other organisms with fewer genes or adapted for local analyses within the human genome, future modifications aimed at tailoring our recursive causal discovery methods for high-dimensional biology data, such as GRNs, could unlock new opportunities for applying our package to complex biological systems.

- Another direction for future work involves the parallelization of our proposed recursive causal discovery methods. By leveraging parallel computing techniques, such as performing removability checks in parallel, we can significantly reduce the computational time, making our algorithms more efficient and scalable.

- Another research question is to analyze the sample complexity of our proposed recursive causal discovery methods. While our methods have empirically shown to require fewer samples compared to other approaches in $\mathfrak{R}_1$-$\mathfrak{R}_4$, a theoretical analysis of sample complexity remains an open question.

References

Raghavendra Addanki, Shiva Kasiviswanathan, Andrew McGregor, and Cameron Musco. Efficient intervention design for causal discovery with latents. In *International Conference on Machine Learning*, pages 63–73. PMLR, 2020.

Ishan Agrawal, Zhijing Jin, Ehsan Mokhtarian, Siyuan Guo, Yuen Chen, Mrinmaya Sachan, and Bernhard Schölkopf. Causalcite: A causal formulation of paper citations. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 8395–8410, 2024.

Kartik Ahuja, Divyat Mahajan, Yixin Wang, and Yoshua Bengio. Interventional causal representation learning. In *International conference on machine learning*, pages 372–407. PMLR, 2023.

Sina Akbari, Ehsan Mokhtarian, AmirEmad Ghassami, and Negar Kiyavash. Recursive causal structure learning in the presence of latent variables and selection bias. *Advances in Neural Information Processing Systems*, 34:10119–10130, 2021.

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

Joshua D Angrist, Guido W Imbens, and Donald B Rubin. Identification of causal effects using instrumental variables. *Journal of the American statistical Association*, 91(434): 444–455, 1996.

Yashas Annadani, Jonas Rothfuss, Alexandre Lacoste, Nino Scherrer, Anirudh Goyal, Yoshua Bengio, and Stefan Bauer. Variational causal networks: Approximate bayesian inference over causal structures. *arXiv preprint arXiv:2106.07635*, 2021.

Bryon Aragam, Arash A Amini, and Qing Zhou. Learning directed acyclic graphs with penalized neighbourhood regression. *arXiv preprint arXiv:1511.08963*, 2015.

Charles K Assaad, Emilie Devijver, and Eric Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022.

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.

Daniel Bernstein, Basil Saeed, Chandler Squires, and Caroline Uhler. Ordering-based causal structure learning in the presence of latent variables. In *International Conference on Artificial Intelligence and Statistics*, pages 4098–4108. PMLR, 2020.

Rohit Bhattacharya, Tushar Nagarajan, Daniel Malinsky, and Ilya Shpitser. Differentiable causal discovery under unmeasured confounding. In *International Conference on Artificial Intelligence and Statistics*, pages 2314–2322. PMLR, 2021.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 33:21865–21877, 2020.

Peter Bühlmann, Jonas Peters, and Jan Ernest. CAM: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42(6):2526–2556, 2014.

Wray Buntine. Theory refinement on bayesian networks. In *Uncertainty proceedings 1991*, pages 52–60. Elsevier, 1991.

Ruichu Cai, Jincheng Ye, Jie Qiao, Huiyuan Fu, and Zhifeng Hao. Fom: Fourth-order moment based causal direction identification on the heteroscedastic data. *Neural Networks*, 124:193–201, 2020.

David Maxwell Chickering. Learning bayesian networks is np-complete. *Learning from data: Artificial intelligence and statistics V*, pages 121–130, 1996.

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

Max Chickering, David Heckerman, and Chris Meek. Large-sample learning of bayesian networks is np-hard. *Journal of Machine Learning Research*, 5, 2004.

Victor W Chu, Raymond K Wong, Wei Liu, and Fang Chen. Causal structure discovery for spatio-temporal data. In *International Conference on Database Systems for Advanced Applications*, pages 236–250. Springer, 2014.

Diego Colombo, Marloes H Maathuis, Markus Kalisch, and Thomas S Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321, 2012.

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

Gregory F Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9:309–347, 1992.

Zijun Cui, Naiyu Yin, Yuru Wang, and Qiang Ji. Empirical bayesian approaches for robust constraint-based causal discovery under insufficient data. *arXiv preprint arXiv:2206.08448*, 2022.

David Danks, Clark Glymour, and Robert Tillman. Integrating locally learned causal structures with overlapping variables. *Advances in Neural Information Processing Systems*, 21, 2008.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pages 518–528. PMLR, 2022.

Chang Deng, Kevin Bello, Bryon Aragam, and Pradeep Kumar Ravikumar. Optimizing notears objectives via topological swaps. In *International Conference on Machine Learning*, pages 7563–7595. PMLR, 2023.

Bao Duong and Thin Nguyen. Heteroscedastic causal structure learning. In *ECAI 2023*, pages 598–605. IOS Press, 2023.

Doris Entner and Patrik O Hoyer. On causal discovery from time series data using fci. *Probabilistic graphical models*, pages 121–128, 2010.

Nir Friedman and Daphne Koller. Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine learning*, 50:95–125, 2003.

Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. In *Proceedings of the fourth annual international conference on Computational molecular biology*, pages 127–135, 2000.

Fei Fu and Qing Zhou. Learning sparse causal Gaussian networks with experimental intervention: regularization and coordinate descent. *Journal of the American Statistical Association*, 108(501):288–300, 2013.

Shunkai Fu and Michel C Desmarais. Markov blanket based feature selection: A review of past decade. In *Proceedings of the world congress on engineering*, volume 1, pages 321–328. Newswood Ltd, 2010.

Ming Gao, Wai Ming Tai, and Bryon Aragam. Optimal estimation of gaussian dag models. In *International Conference on Artificial Intelligence and Statistics*, pages 8738–8757. PMLR, 2022.

Dan Geiger and David Heckerman. Learning Gaussian networks. In *Uncertainty Proceedings 1994*, pages 235–243. Elsevier, 1994.

AmirEmad Ghassami, Saber Salehkaleybar, and Negar Kiyavash. Optimal experiment design for causal discovery from fixed number of experiments. *arXiv preprint arXiv:1702.08567*, 2017.

Asish Ghoshal and Jean Honorio. Information-theoretic limits of bayesian network structure learning. In *Artificial Intelligence and Statistics*, pages 767–775. PMLR, 2017.

Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019.

Chang Gong, Di Yao, Chuzhe Zhang, Wenbin Li, and Jingping Bi. Causal discovery from temporal data: An overview and new perspectives. *arXiv preprint arXiv:2303.10112*, 2023.

Olivier Goudet, Diviyan Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michele Sebag. Learning functional causal models with generative neural networks. *Explainable and interpretable models in computer vision and machine learning*, pages 39–80, 2018.

Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

Jiaying Gu, Fei Fu, and Qing Zhou. Penalized estimation of directed acyclic graphs from discrete data. *Statistics and Computing*, 29:161–176, 2019.

Wiebke Günther, Urmi Ninad, Jonas Wahl, and Jakob Runge. Conditional independence testing with heteroskedastic data and applications to causal discovery. *Advances in Neural Information Processing Systems*, 35:16191–16202, 2022.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

Naftali Harris and Mathias Drton. Pc algorithm for nonparanormal graphical models. *Journal of Machine Learning Research*, 14(11), 2013.

Uzma Hasan, Emam Hossain, and Md Osman Gani. A survey on causal discovery methods for temporal and non-temporal data. *arXiv preprint arXiv:2303.15027*, 2023.

Alain Hauser and Peter Bühlmann. Two optimal strategies for active learning of causal models from interventional data. *International Journal of Approximate Reasoning*, 55 (4):926–939, 2014.

David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20:197–243, 1995.

James J Heckman. Econometric causality. *International Statistical Review*, 76(1):1–27, 2008.

Nu Hoang, Bao Duong, and Thin Nguyen. Scalable variational causal discovery unconstrained by acyclicity. In *ECAI 2024*, pages 2685–2692. IOS Press, 2024.

Huining Hu, Zhentao Li, and Adrian R Vetta. Randomized experimental design for causal graph discovery. *Advances in neural information processing systems*, 27, 2014.

Biwei Huang, Kun Zhang, Jiji Zhang, Joseph Ramsey, Ruben Sanchez-Romero, Clark Glymour, and Bernhard Schölkopf. Causal discovery from heterogeneous/nonstationary data. *Journal of Machine Learning Research*, 21(89):1–53, 2020.

Antti Hyttinen, Frederick Eberhardt, and Patrik O Hoyer. Experiment selection for causal discovery. *Journal of Machine Learning Research*, 14:3041–3071, 2013.

Fateme Jamshidi, Luca Ganassali, and Negar Kiyavash. On the sample complexity of conditional independence testing with von mises estimator with application to causal discovery. In *Forty-first International Conference on Machine Learning*, 2024.

Diviyan Kalainathan, Olivier Goudet, Isabelle Guyon, David Lopez-Paz, and Michèle Sebag. Structural agnostic modeling: Adversarial learning of causal graphs. *The Journal of Machine Learning Research*, 23(1):9831–9892, 2022.

Genta Kikuchi. Differentiable causal discovery under heteroscedastic noise. In *International Conference on Neural Information Processing*, pages 284–295. Springer, 2022.

Neville Kenneth Kitson, Anthony C Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. A survey of bayesian network structure learning. *Artificial Intelligence Review*, 56(8):8721–8814, 2023.

Murat Kocaoglu, Amin Jaber, Karthikeyan Shanmugam, and Elias Bareinboim. Characterization and learning of causal graphs with latent variables from soft interventions. *Advances in Neural Information Processing Systems*, 32, 2019.

Janne Korhonen and Pekka Parviainen. Exact learning of bounded tree-width bayesian networks. In *Artificial Intelligence and Statistics*, pages 370–378. PMLR, 2013.

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

Wai-Yin Lam, Bryan Andrews, and Joseph Ramsey. Greedy relaxations of the sparsest permutation algorithm. In *Uncertainty in Artificial Intelligence*, pages 1052–1062. PMLR, 2022.

Thuc Duy Le, Tao Hoang, Jiuyong Li, Lin Liu, Huawen Liu, and Shu Hu. A fast pc algorithm for high dimensional causal discovery with multi-core pcs. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(5):1483–1495, 2016.

Hao-Chih Lee, Matteo Danieletto, Riccardo Miotto, Sarah T Cherng, and Joel T Dudley. Scaling structural learning with no-bears to infer causal transcriptome networks. In *Pacific Symposium on Biocomputing 2020*, pages 391–402. World Scientific, 2019.

Adam Li, Amin Jaber, and Elias Bareinboim. Causal discovery from observational and interventional data across multiple environments. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Lars Lorch, Jonas Rothfuss, Bernhard Schölkopf, and Andreas Krause. Dibs: Differentiable bayesian structure learning. *Advances in Neural Information Processing Systems*, 34: 24111–24123, 2021.

Pingchuan Ma, Rui Ding, Qiang Fu, Jiaru Zhang, Shuai Wang, Shi Han, and Dongmei Zhang. Scalable differentiable causal discovery in the presence of latent confounders with skeleton posterior. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2141–2152, 2024.

Dimitris Margaritis and Sebastian Thrun. Bayesian network induction via local neighborhoods. *Advances in Neural Information Processing Systems*, 12:505–511, 1999.

Christopher Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 403–410, 1995.

Panagiotis Misiakos, Chris Wendler, and Markus Püschel. Learning dags from data with few root causes. *Advances in Neural Information Processing Systems*, 36, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ehsan Mokhtarian, Sina Akbari, AmirEmad Ghassami, and Negar Kiyavash. A recursive Markov boundary-based approach to causal structure learning. In *The KDD'21 Workshop on Causal Discovery*, pages 26–54. PMLR, 2021.

Ehsan Mokhtarian, Sina Akbari, Fateme Jamshidi, Jalal Etesami, and Negar Kiyavash. Learning bayesian networks in the presence of structural side information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7814–7822, 2022.

Ehsan Mokhtarian, Mohmmadsadegh Khorasani, Jalal Etesami, and Negar Kiyavash. Novel ordering-based approaches for causal structure learning in the presence of unobserved variables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12260–12268, 2023a.

Ehsan Mokhtarian, Saber Salehkaleybar, AmirEmad Ghassami, and Negar Kiyavash. A unified experiment design approach for cyclic and acyclic causal models. *Journal of Machine Learning Research*, 24(354):1–31, 2023b.

Ehsan Mokhtarian, Negar Kiyavash, and Federico Canè. Sector and style factor rotations in equity markets: Detection and towards causation. *Available at SSRN 4966272*, 2024.

Joris M Mooij, Dominik Janzing, Tom Heskes, and Bernhard Schölkopf. On causal discovery with cyclic additive noise models. *Advances in neural information processing systems*, 24, 2011.

Joris M Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing cause from effect using observational data: methods and benchmarks. *The Journal of Machine Learning Research*, 17(1):1103–1204, 2016.

Stephen L Morgan and Christopher Winship. *Counterfactuals and Causal Inference: Methods and Principles for Social Research*. Cambridge University Press, 2015.

Ignavier Ng, Shengyu Zhu, Zhitang Chen, and Zhuangyan Fang. A graph autoencoder approach to causal structure learning. *arXiv preprint arXiv:1911.07420*, 2019.

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.

Siqi Nie, Denis D Mauá, Cassio P De Campos, and Qiang Ji. Advances in learning bayesian networks of bounded treewidth. *Advances in neural information processing systems*, 27: 2285–2293, 2014.

Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605. PMLR, 2020.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

Judea Pearl. Causality: Models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 19(2):3, 2000.

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

Jean-Philippe Pellet and André Elisseeff. Finding latent causes in causal networks: an efficient approach based on Markov blankets. *Neural Information Processing Systems Foundation*, 2008a.

Jean-Philippe Pellet and André Elisseeff. Using Markov blankets for causal structure learning. *Journal of Machine Learning Research*, 9(Jul):1295–1342, 2008b.

Adrian E Raftery. Bayesian model selection in social research. *Sociological methodology*, pages 111–163, 1995.

Vaidyanathan Peruvemba Ramaswamy and Stefan Szeider. Turbocharging treewidth-bounded bayesian network structure learning. In *Proceeding of AAAI-21, the Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

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

Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, 3:121 – 129, 2016. URL `https://api.semanticscholar.org/CorpusID:3655067`.

Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International journal of data science and analytics*, 3:121–129, 2017.

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

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.

Thomas Richardson, Peter Spirtes, et al. Ancestral graph Markov models. *The Annals of Statistics*, 30(4):962–1030, 2002.

Thomas S Richardson. *Discovering cyclic causal structure*. Carnegie Mellon [Department of Philosophy], 1996a.

Thomas S Richardson. A discovery algorithm for directed cyclic graphs. In *Conference on Uncertainty in Artificial Intelligence*, pages 454–461, 1996b.

Mauro Scanagatta, Cassio P de Campos, Giorgio Corani, and Marco Zaffalon. Learning bayesian networks with thousands of variables. *Advances in neural information processing systems*, 28, 2015.

Eric E Schadt, Stephanie A Monks, Thomas A Drake, Aldons J Lusis, Nelson Che, Victor Colinayo, ..., and Stephen H Friend. Genetics of gene expression surveyed in maize, mouse and man. *Nature*, 422(6929):297–302, 2003.

Mark Schmidt, Alexandru Niculescu-Mizil, Kevin Murphy, et al. Learning graphical model structure using l1-regularization paths. In *AAAI*, volume 7, pages 1278–1283, 2007.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

Muralikrishnna G Sethuraman, Romain Lopez, Rahul Mohan, Faramarz Fekri, Tommaso Biancalani, and Jan-Christian Hütter. Nodags-flow: Nonlinear cyclic causal structure learning. In *International Conference on Artificial Intelligence and Statistics*, pages 6371–6387. PMLR, 2023.

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

Peter Spirtes and Thomas Richardson. A polynomial time algorithm for determining dag equivalence in the presence of latent variables and selection bias. In *Proceedings of the 6th International Workshop on Artificial Intelligence and Statistics*, pages 489–500. Citeseer, 1996.

Peter Spirtes, Christopher Meek, and Thomas Richardson. Causal inference in the presence of latent variables and selection bias. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 499–506, 1995.

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

Xiaohai Sun, Dominik Janzing, Bernhard Schölkopf, and Kenji Fukumizu. A kernel-based causal learning algorithm. In *Proceedings of the 24th international conference on Machine learning*, pages 855–862, 2007.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Marc Teyssier and Daphne Koller. Ordering-based search: a simple and effective algorithm for learning bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 584–590, 2005.

Christian Toth, Lars Lorch, Christian Knoll, Andreas Krause, Franz Pernkopf, Robert Peharz, and Julius Von Kügelgen. Active bayesian causal inference. *Advances in Neural Information Processing Systems*, 35:16261–16275, 2022.

Ioannis Tsamardinos and Constantin F Aliferis. Towards principled feature selection: Relevancy, filters and wrappers. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. Algorithms for large scale Markov blanket discovery. In *FLAIRS conference*, volume 2, pages 376–380, 2003.

Thomas Verma and Judea Pearl. *Equivalence and synthesis of causal models*. UCLA, Computer Science Department, 1991.

Jussi Viinikka, Antti Hyttinen, Johan Pensar, and Mikko Koivisto. Towards scalable bayesian learning of causal dags. *Advances in Neural Information Processing Systems*, 33:6584–6594, 2020.

Matthew J Vowels, Necati Cihan Camgoz, and Richard Bowden. D'ya like dags? a survey on structure learning and causal discovery. *ACM Computing Surveys*, 55(4):1–36, 2022.

Yixin Wang and Michael I Jordan. Desiderata for representation learning: A causal perspective. *arXiv preprint arXiv:2109.03795*, 2021.

Dennis Wei, Tian Gao, and Yue Yu. Dags with no fears: A closer look at continuous optimization for learning bayesian networks. *Advances in Neural Information Processing Systems*, 33:3895–3906, 2020.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Anpeng Wu, Kun Kuang, Minqin Zhu, Yingrong Wang, Yujia Zheng, Kairong Han, Baohong Li, Guangyi Chen, Fei Wu, and Kun Zhang. Causality for large language models. *arXiv preprint arXiv:2410.15319*, 2024.

Sandeep Yaramakala and Dimitris Margaritis. Speculative Markov blanket discovery for optimal feature selection. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

Naiyu Yin, Tian Gao, Yue Yu, and Qiang Ji. Effective causal discovery under identifiable heteroscedastic noise model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16486–16494, 2024.

Kui Yu, Lin Liu, Jiuyong Li, and Huanhuan Chen. Mining Markov blankets without causal sufficiency. *IEEE transactions on neural networks and learning systems*, 29(12):6333–6347, 2018.

Shiqing Yu, Mathias Drton, and Ali Shojaie. Directed graphical models and causal discovery for zero-inflated data. In *Conference on Causal Learning and Reasoning*, pages 27–67. PMLR, 2023.

Yue Yu, Jie Chen, Tian Gao, and Mo Yu. Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR, 2019.

Yue Yu, Tian Gao, Naiyu Yin, and Qiang Ji. Dags with no curl: An efficient dag structure learning approach. In *International Conference on Machine Learning*, pages 12156–12166. Pmlr, 2021.

Jiji Zhang. Causal reasoning with ancestral graphs. *Journal of Machine Learning Research*, 9:1437–1474, 2008a.

Jiji Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873–1896, 2008b.

Zeyu Zhang, Chaozhuo Li, Xu Chen, and Xing Xie. Bayesian active causal discovery with multi-fidelity experiments. *Advances in Neural Information Processing Systems*, 36, 2024.

Boxin Zhao, Percy S Zhai, Y Samuel Wang, and Mladen Kolar. High-dimensional functional graphical model structure learning via neighborhood selection approach. *Electronic Journal of Statistics*, 18(1):1042–1129, 2024.

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.

Fangting Zhou, Kejun He, and Yang Ni. Causal discovery with heterogeneous observational data. In *Uncertainty in Artificial Intelligence*, pages 2383–2393. PMLR, 2022.