Learning Bayesian Networks in the Presence of Structural Side Information

Ehsan Mokhtarian,[1] Sina Akbari,[1] Fateme Jamshidi,[2] Jalal Etesami,[1] Negar Kiyavash [1,2]

[1] Department of Computer and Communication Science, EPFL, Lausanne, Switzerland
[2] College of Management of Technology, EPFL, Lausanne, Switzerland
{ehsan.mokhtarian, sina.akbari, fateme.jamshidi, seyed.etesami, negar.kiyavash}@epfl.ch

Abstract

We study the problem of learning a Bayesian network (BN) of a set of variables when structural side information about the system is available. It is well known that learning the structure of a general BN is both computationally and statistically challenging. However, often in many applications, side information about the underlying structure can potentially reduce the learning complexity. In this paper, we develop a recursive constraint-based algorithm that efficiently incorporates such knowledge (i.e., side information) into the learning process. In particular, we study two types of structural side information about the underlying BN: (I) an upper bound on its clique number is known, or (II) it is diamond-free. We provide theoretical guarantees for the learning algorithms, including the worst-case number of tests required in each scenario. As a consequence of our work, we show that bounded treewidth BNs can be learned with polynomial complexity. Furthermore, we evaluate the performance and the scalability of our algorithms in both synthetic and real-world structures and show that they outperform the state-of-the-art structure learning algorithms.

Introduction

Bayesian networks (BNs) are probabilistic graphical models that represent conditional dependencies in a set of random variables via directed acyclic graphs (DAGs). Due to their succinct representations and power to improve the prediction and to remove systematic biases in inference (Pearl 2009; Spirtes et al. 2000), BNs have been widely applied in various areas including medicine (Flores et al. 2011), bioinformatics (Friedman et al. 2000), ecology (Pollino et al. 2007), etc. Learning a BN from data is in general NP-hard (Chickering, Heckerman, and Meek 2004). However, any type of side information about the network can potentially reduce the complexity of the learning task.

BN structure learning algorithms are of three flavors: constraint-based, e.g., parent-child (PC) algorithm (Spirtes et al. 2000), score-based, e.g., (Chickering 2002; Solus, Wang, and Uhler 2017; Zheng et al. 2018; Zhu, Ng, and Chen 2020), and hybrid, e.g., MMHC algorithm (Tsamardinos, Brown, and Aliferis 2006).

Although constraint-based methods do not require any assumptions about the underlying generative model, they often require conditional independence (CI) tests with large conditioning sets or a large number of CI tests which grows exponentially as the number of variables increases[1]. Often in practice, we have side information about the network that can improve learning accuracy or reduce complexity. We show in this work that such side information can reduce the learning complexity to polynomial in terms of the number of CI tests. Our main contributions are as follows.

- We propose a constraint-based Recursive Structure Learning (RSL) algorithm to recover BNs. In addition, we study 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. In each case, we provide a learning algorithm. RSL follows a divide-and-conquer approach: it breaks the learning problem into several sub-problems that are similar to the original problem but smaller in size by eliminating *removable* variables (see Definition 1). Thus, in each recursion, both the size of the conditioning sets and the number of CI tests decrease.
- Learning BNs with bounded treewidth has recently attracted attention. Works such as (Korhonen and Parviainen 2013; Nie et al. 2014; Ramaswamy and Szeider 2021) aim to develop learning algorithms for BNs when an upper bound on the treewidth of the graph is given as side information. Assuming bounded treewidth is more restrictive than bounded clique number assumption, i.e., having a bound on the treewidth implies an upper bound on the clique number of the network. Hence, our proposed algorithm with structural side information of type (I) can also learn bounded treewidth BNs. However, our algorithm has polynomial complexity, while the state-of-the-art exact learning algorithms have exponential complexity.
- We show that when the clique number of the underlying BN is upper bounded by m, i.e., $\omega(\mathcal{G}) \leq m$ (See Table 1 for the graphical notations), our algorithm requires $\mathcal{O}(n^2 + n\Delta_{in}^{m+1})$ CI tests (Theorem 1). Furthermore, when the graph is diamond-free, our algorithm requires $\mathcal{O}(n^2 + n\Delta_{in}^{3})$ CI tests (Theorem 2). These bounds significantly improve over the state of the art.

[1] See (Scutari 2014) for an overview on implementations of constraint-based algorithms.

n	Number of variables
$\Delta(\mathcal{G})$	Maximum degree of DAG $\mathcal{G}$
$\Delta_{in}(\mathcal{G})$	Maximum in-degree of DAG $\mathcal{G}$
$\omega(\mathcal{G})$	Clique number of graph $\mathcal{G}$
$N_{\mathcal{G}}(X)$	Neighbors of X in DAG $\mathcal{G}$
$Ch_{\mathcal{G}}(X)$	Children of X in DAG $\mathcal{G}$
$Pa_{\mathcal{G}}(X)$	Parents of X in DAG $\mathcal{G}$
$CP_{\mathcal{G}}(X)$	Co-parents of X in DAG $\mathcal{G}$
$Mb_{\mathbf{V}}(X)$	Markov boundary of X among set $\mathbf{V}$
$\alpha(\mathcal{G})$	Maximum Mb size of $\mathcal{G}$

Table 1: Graphical notations that we use in this paper.

Related work: Herein, we review the relevant work on BN learning methods as well as those with side information.

The PC algorithm (Spirtes et al. 2000) is a classical example of constraint-based methods that requires $\mathcal{O}(n^{\Delta})$ number of CI tests. CS (Pellet and Elisseeff 2008) and MARVEL (Mokhtarian et al. 2021) are two examples that focus on BN structure learning with small number of CI tests by using the Markov boundaries (Mbs). This results in $\mathcal{O}(n^2 2^{\alpha})$ and $\mathcal{O}(n^2 + n\Delta_{\text{in}}^2 2^{\Delta_{\text{in}}})$ number of CI tests for CS and MARVEL, respectively. On the other hand, methods such as GS (Margaritis and Thrun 1999), MMPC (Tsamardinos, Aliferis, and Statnikov 2003a), and HPC (de Morais and Aussem 2010) focus on reducing the size of the conditioning sets in their CI tests. However, the aforementioned methods are not equipped to take advantage of side information. Table 2 compares the complexity of various constraint-based algorithms in terms of their CI tests. RSL$_{\omega}$ and RSL$_{D}$ are our proposed algorithms when an upper bound on the clique number is given and when the BN is diamond-free, respectively. Note that in general, $\Delta_{\text{in}} \leq \Delta \leq \alpha$, and in a DAG with a constant in-degree, Δ and α can grow linearly with the number of variables.

Side information about the underlying generative model has been harnessed for structure learning in limited fashion, e.g., (Sesen et al. 2013; Flores et al. 2011; Oyen, Anderson, and Anderson-Cook 2016; McLachlan et al. 2020). As an example, (Takeishi and Kawahara 2020) propose an approach to incorporate side knowledge about feature relations into the learning process. (Shimizu 2019) and (Sondhi and Shojaie 2019) study the structure learning problem when the data is from a linear structural equation model and propose LiNGAM and reduced PC algorithms, respectively. (Claassen, M. Mooij, and Heskes 2013; Zheng et al. 2020) consider learning sparse BNs. In particular, (Claassen, M. Mooij, and Heskes 2013) show that in sparse setting, BN recovery is no longer NP-hard, even in the presence of unobserved variables. That is for sparse graphs with maximum node degree of Δ, a sound and complete BN can be obtained by performing $\mathcal{O}(n^{2(\Delta+2)})$ CI tests.

Algorithm	#CI tests
PC	$\mathcal{O}(n^{\Delta})$
GS	$\mathcal{O}(n^2 + n\alpha^2 2^{\alpha})$
MMPC, CS	$\mathcal{O}(n^2 2^{\alpha})$
MARVEL	$\mathcal{O}(n^2 + n\Delta_{in}^2 2^{\Delta_{in}})$
RSL$_D$	$\mathcal{O}(n^2 + n\Delta_{in}^3)$
RSL$_{\omega}$	$\mathcal{O}(n^2 + n\Delta_{in}^{m+1})$

Table 2: Required number of CI tests in the worst case by various algorithms.

Side information has been incorporated into score-based methods in limited fashions too, e.g., (Chen et al. 2016; Li and van Beek 2018; Bartlett and Cussens 2017). The side information in the aforementioned works is in the form of ancestral constraints which are about the absence or presence of a directed path between two vertices in the underlying BN. (Bartlett and Cussens 2017) cast this problem as an integer linear program. The proposed method by (Chen et al. 2016) recovers the network with guaranteed optimality but it does not scale beyond 20 random variables. The method by (Li and van Beek 2018) scales up to 50 variables but it does not provide any optimality guarantee.

Another related problem is optimizing $\sum_{v \in \mathbf{V}} f_v(Pa(v))$ over a set of DAGs with vertices $\mathbf{V}$ and parent sets $\{Pa(v)\}_{v \in \mathbf{V}}$. In this problem $\{f_v(\cdot)\}_{v \in \mathbf{V}}$ is a set of predefined local score functions. This problem is NP-hard (Chickering, Heckerman, and Meek 2004). Note that the BN structure learning can be formulated as a special case of this problem by selecting appropriate local score functions. (Korhonen and Parviainen 2013) introduce an exact algorithm for solving this problem with complexity $3^n n^{t+\mathcal{O}(1)}$ under a constraint that the optimal BN has treewidth at most t. (Elidan and Gould 2008) propose a heuristic algorithm that finds a suboptimal DAG with bounded treewidth which runs in time polynomial in n and t. Knowing a bound on the treewidth is yet another type of structural side information that is more restrictive[2] than our structural assumptions. Therefore, RSL$_{\omega}$ can learn bounded treewidth BNs with polynomial complexity, i.e., $\mathcal{O}(n^2 + n\Delta_{in}^{t+2})$, where t is a bound on the treewidth and $\Delta_{in} < n$.

(Korhonen and Parviainen 2015) is another score-based method that study the BN structure learning when an upper bound k on the vertex cover number of the underlying BN is available. Their algorithm has complexity $4^k n^{2k+\mathcal{O}(1)}$. Since the vertex cover number of a graph is greater than its clique number minus one, then RSL$_{\omega}$ can also recover a bounded vertex cover numbers BN with complexity $\mathcal{O}(n^2 + n\Delta_{in}^{k+2})$. (Grüttemeier and Komusiewicz 2020) consider the structural constraint that the moralized graph can be transformed into a graph with maximum degree one by at most r vertex deletions. They show that under this constraint, an optimal network can be learned in $n^{\mathcal{O}(r^2)}$ time.

Preliminaries

Throughout the paper, we use capital letters for random variables and bold letters for sets. Also, the graphical notations are presented in Table 1.

A graph is defined as a pair $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V}$ is a finite set of vertices and $\mathbf{E}$ is the set of edges. If $\mathbf{E}$ is a set of unordered pairs of vertices, the graph is called *undirected*

[2] In General, Treewidth$+1 \geq \omega$, (Bodlaender and Möhring 1993).

and if it is a set of ordered pairs, it is called *directed*. An undirected graph is called *complete* if $\mathbf{E}$ contains all edges. A *directed acyclic graph* (DAG) is a directed graph with no directed cycle. In an edge $(X, Y) \in \mathbf{E}$ (or $\{X, Y\} \in \mathbf{E}$, in case of an undirected graph), the vertices X and Y are the endpoints of that edge and they are called *neighbors*. Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a (directed or undirected) graph and $\overline{\mathbf{V}} \subseteq \mathbf{V}$, then the *induced subgraph* $\mathcal{G}[\overline{\mathbf{V}}]$ is the graph whose vertex set is $\overline{\mathbf{V}}$ and whose edge set consists of all of the edges in $\mathbf{E}$ that have both endpoints in $\overline{\mathbf{V}}$. The *skeleton* of a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is its undirected version. The *clique number* of an undirected graph $\mathcal{G}$ is the number of vertices in the largest induced subgraph of $\mathcal{G}$ that is complete.

Let $\mathbf{X}, \mathbf{Y}$, and $\mathbf{S}$ be three disjoint subsets of $\mathbf{V}$. We use $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{S}$ to indicate $\mathbf{S}$ d-separates[3] $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}$. In this case, the set $\mathbf{S}$ is called a *separating* set for $\mathbf{X}$ and $\mathbf{Y}$. Suppose $P_{\mathbf{V}}$ is the joint probability distribution of $\mathbf{V}$. We use $\mathbf{X} \perp\!\!\!\perp_{P_{\mathbf{V}}} \mathbf{Y}|\mathbf{S}$ to denote the Conditional Independence (CI) of $\mathbf{X}$ and $\mathbf{Y}$ given $\mathbf{S}$. Also, a CI test refers to detecting whether $X \perp\!\!\!\perp_{P_{\mathbf{V}}} Y|\mathbf{S}$. A DAG $\mathcal{G}$ is said to be an *independency map (I-map)* of $P_{\mathbf{V}}$ if for every three disjoint subsets of vertices $\mathbf{X}, \mathbf{Y}$, and $\mathbf{S}$ we have $X \perp_{\mathcal{G}} Y|\mathbf{S} \Rightarrow X \perp\!\!\!\perp_{P_{\mathbf{V}}} Y|\mathbf{S}$. A DAG $\mathcal{G}$ is a *minimal* I-map of $P_{\mathbf{V}}$ if it is an I-map of $P_{\mathbf{V}}$ and the resulting DAG after removing any edge is no longer an I-map of $P_{\mathbf{V}}$. A DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is called a *Bayesian network* (BN) of $P_{\mathbf{V}}$, if and only if $\mathcal{G}$ is a minimal I-map of $P_{\mathbf{V}}$. The joint probability distribution $P_{\mathbf{V}}$ with a BN $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ satisfies the Markov factorization property, that is $P_{\mathbf{V}} = \prod_{X \in \mathbf{V}} P_{\mathbf{V}}(X|Pa_{\mathcal{G}}(X))$ (Pearl 1988).

A joint distribution $P_{\mathbf{V}}$ may have several BNs. The *Markov equivalence class* (MEC) of $P_{\mathbf{V}}$, denoted by $\langle P_{\mathbf{V}} \rangle$, is the set of all its BNs. It has been shown that two DAGs belong to a MEC if and only if they share the same skeleton and the same set of v-structures[4] (Pearl 2009). A MEC $\langle P_{\mathbf{V}} \rangle$ can be uniquely represented by a partially directed graph[5] called *essential graph*. A DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is called a *dependency map (D-map)* of $P_{\mathbf{V}}$ if for every three disjoint subsets of vertices $\mathbf{X}, \mathbf{Y}$, and $\mathbf{S}$, $\mathbf{X} \perp\!\!\!\perp_{P_{\mathbf{V}}} \mathbf{Y}|\mathbf{S}$ implies $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{S}$. This property is also known as *faithfulness* in the causality literature (Pearl 2009). Furthermore, $\mathcal{G}$ is called a *perfect map* if it is both an I-map and a D-map of $P_{\mathbf{V}}$, i.e., $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{S} \iff \mathbf{X} \perp\!\!\!\perp_{P_{\mathbf{V}}} \mathbf{Y}|\mathbf{S}$. Note that if $\mathcal{G}$ is perfect map of $P_{\mathbf{V}}$, then it belongs to $\langle P_{\mathbf{V}} \rangle$, i.e., a perfect map is a BN.

Problem description: The BN structure learning problem involves identifying $\langle P_{\mathbf{V}} \rangle$ from $P_{\mathbf{V}}$ on the population-level or from a set of samples of $P_{\mathbf{V}}$. As mentioned earlier, the constraint-based methods perform this task using a series of CI tests. In this paper, we consider the BN structure learning problem using a constraint-based method, when we are given structural side information about the underlying DAG.

Algorithm 1: Recursive Structure Learning (RSL).

```
1: Input: $\mathbf{V}$, $P_{\mathbf{V}}$, SideInfo
2: $Mb_{\mathbf{V}} \leftarrow$ ComputeMb($\mathbf{V}$, $P_{\mathbf{V}}$)
3: $(\mathcal{H}, \boldsymbol{\mathcal{S}}_{\mathbf{V}}) \leftarrow$ RSL($\mathbf{V}$, $P_{\mathbf{V}}$, $Mb_{\mathbf{V}}$, SideInfo)

1: RSL($\overline{\mathbf{V}}$, $P_{\overline{\mathbf{V}}}$, $Mb_{\overline{\mathbf{V}}}$, SideInfo)
2: if $|\overline{\mathbf{V}}| = 1$ then
3:     return $((\overline{\mathbf{V}}, \varnothing), \varnothing)$
4: else
5:     $X \leftarrow$ FindRemovable($\overline{\mathbf{V}}$, $P_{\overline{\mathbf{V}}}$, $Mb_{\overline{\mathbf{V}}}$, SideInfo)
6:     $(N_{\mathcal{G}[\overline{\mathbf{V}}]}(X), \boldsymbol{\mathcal{S}}_X) \leftarrow$
           FindNeighbors($X$, $\overline{\mathbf{V}}$, $P_{\overline{\mathbf{V}}}$, $Mb_{\overline{\mathbf{V}}}(X)$, SideInfo)
7:     $Mb_{\overline{\mathbf{V}} \setminus \{X\}} \leftarrow$ UpdateMb($X$, $P_{\overline{\mathbf{V}}}$, $N_{\mathcal{G}[\overline{\mathbf{V}}]}(X)$, $Mb_{\overline{\mathbf{V}}}$)
8:     $(\mathcal{H}[\overline{\mathbf{V}} \setminus \{X\}], \boldsymbol{\mathcal{S}}_{\overline{\mathbf{V}} \setminus \{X\}}) \leftarrow$
           RSL($\overline{\mathbf{V}} \setminus \{X\}$, $P_{\overline{\mathbf{V}} \setminus \{X\}}$, $Mb_{\overline{\mathbf{V}} \setminus \{X\}}$, SideInfo)
9:     Construct $\mathcal{H}[\overline{\mathbf{V}}]$ by $\mathcal{H}[\overline{\mathbf{V}} \setminus \{X\}]$ and undirected
       edges between $X$ and $N_{\mathcal{G}[\overline{\mathbf{V}}]}(X)$.
10:    $\boldsymbol{\mathcal{S}}_{\overline{\mathbf{V}}} \leftarrow \boldsymbol{\mathcal{S}}_{\overline{\mathbf{V}} \setminus \{X\}} \cup \boldsymbol{\mathcal{S}}_X$
11:    return $(\mathcal{H}[\overline{\mathbf{V}}], \boldsymbol{\mathcal{S}}_{\overline{\mathbf{V}}})$
```

Learning Bayesian Networks Recursively

Suppose $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a perfect map of $P_{\mathbf{V}}$ and let $\mathcal{H}$ denote its skeleton. Recall that learning $\langle P_{\mathbf{V}} \rangle$ requires recovering $\mathcal{H}$ and the set of v-structures of $\mathcal{G}$. It has been shown that finding a separating set for each pair of non-neighbor vertices in $\mathcal{G}$ suffices to recover its set of v-structures (Spirtes et al. 2000). Thus, we propose an algorithm called Recursive Structure Learning (**RSL**) that recursively finds $\mathcal{H}$ along with a set of separating sets $\boldsymbol{\mathcal{S}}_{\mathbf{V}}$ for non-neighbor vertices in $\mathbf{V}$. The pseudocode of **RSL** is presented in Algorithm 1.

RSL's inputs comprise a subset $\overline{\mathbf{V}} \subseteq \mathbf{V}$ with its joint distribution $P_{\overline{\mathbf{V}}}$[6] such that $\mathcal{G}[\overline{\mathbf{V}}]$ is a perfect map of $P_{\overline{\mathbf{V}}}$, and their Markov boundaries $Mb_{\overline{\mathbf{V}}}$ (see Definition 2), along with structural side information, which can be either diamond-freeness, or an upper bound on the clique number. In this case, **RSL** outputs $\mathcal{H}[\overline{\mathbf{V}}]$ and a set of separating sets $\boldsymbol{\mathcal{S}}_{\overline{\mathbf{V}}}$ for non-neighbor vertices in $\overline{\mathbf{V}}$. The **RSL** consists of three main sub-algorithms: **FindRemovable**, **FindNeighbors**, and **UpdateMb**. It begins by calling **FindRemovable** in line 5 to find a vertex $X \in \overline{\mathbf{V}}$ such that the resulting graph after removing X from the vertex set, $\mathcal{G}[\overline{\mathbf{V}} \setminus \{X\}]$, remains a perfect map of $P_{\overline{\mathbf{V}} \setminus \{X\}}$. Afterwards, in line 6, **FindNeighbors** identifies the neighbors of X in $\mathcal{G}[\overline{\mathbf{V}}]$ and a set of separating sets for X and each of its non-neighbors in this graph. In lines 7 and 8, **RSL** updates the Markov boundaries and calls itself to learn the remaining graph after removing vertex X, i.e., $\mathcal{G}[\overline{\mathbf{V}} \setminus \{X\}]$, respectively. The two functions **FindRemovable** and **FindNeighbors** take advantage of the provided side information, as we shall discuss later.

As mentioned above, it is necessary for $\mathcal{G}[\overline{\mathbf{V}}]$ to remain a perfect map of $P_{\overline{\mathbf{V}}}$ at each iteration. This cannot be guar-

[3] See Appendix A (arxiv.org/abs/2112.10884) for the definition.

[4] Three vertices X, Y, and Z form a *v-structure* if $X \rightarrow Y \leftarrow Z$ while X and Z are not neighbors.

[5] It is a graph with both directed and undirected edges.

[6] In practice, the finite sample data at hand is used instead of $P_{\overline{\mathbf{V}}}$.

anteed if X is chosen arbitrarily. (Mokhtarian et al. 2021) introduced the notion of removability in the context of causal graphs and showed that removable variables are the ones that preserve the perfect map assumption after the distribution is marginalized over them. In this work, we introduce a similar concept in the context of BN structure recovery.

Definition 1 (Removable)**.** *Suppose $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a DAG and $X \in \mathbf{V}$. Vertex X is called removable in $\mathcal{G}$ if the d-separation relations in $\mathcal{G}$ and $\mathcal{G}[\mathbf{V} \setminus \{X\}]$ are equivalent over $\mathbf{V} \setminus \{X\}$. That is, for any vertices $Y, Z \in \mathbf{V} \setminus \{X\}$ and $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y, Z\}$,*

$$Y \perp_{\mathcal{G}} Z|\mathbf{S} \iff Y \perp_{\mathcal{G}[\mathbf{V}\setminus\{X\}]} Z|\mathbf{S}. \qquad (1)$$

Proposition 1. *Suppose $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a perfect map of $P_{\mathbf{V}}$. For each variable $X \in \mathbf{V}$, $\mathcal{G}[\mathbf{V} \setminus \{X\}]$ is a perfect map of $P_{\mathbf{V}\setminus\{X\}}$ if and only if X is a removable vertex in $\mathcal{G}$.*

All proofs appear in Appendix B.

Markov boundary (Mb): Our proposed algorithm uses the notion of Markov boundary.

Definition 2 (Mb)**.** *Suppose $P_{\mathbf{V}}$ is the joint distribution on $\mathbf{V}$. The Mb of $X \in \mathbf{V}$, denoted by $Mb_{\mathbf{V}}(X)$, is a minimal set $\mathbf{S} \subseteq \mathbf{V} \setminus \{X\}$ s.t. $X \perp\!\!\!\perp_{P_{\mathbf{V}}} \mathbf{V} \setminus (\mathbf{S} \cup \{X\})|\mathbf{S}$. We denote $(Mb_{\mathbf{V}}(X)\colon\ X \in \mathbf{V})$ by $Mb_{\mathbf{V}}$.*

Definition 3 (co-parent)**.** *Two non-neighbor variables are called co-parents in $\mathcal{G}$, if they share at least one child. For $X \in \mathbf{V}$, the set of co-parents of X is denoted by $CP_{\mathcal{G}}(X)$.*

If $\mathcal{G}$ is a perfect map of $P_{\mathbf{V}}$, for every vertex $X \in \mathbf{V}$, $Mb_{\mathbf{V}}(X)$ is unique (Pearl 1988) and it is equal to

$$Mb_{\mathbf{V}}(X) = Pa_{\mathcal{G}}(X) \cup Ch_{\mathcal{G}}(X) \cup CP_{\mathcal{G}}(X). \qquad (2)$$

The subroutines **FindRemovable** and **FindNeighbors** need the knowledge of Mbs to perform their tasks. Several constraint-based and scored-based algorithms have been developed in literature such as TC (Pellet and Elisseeff 2008), GS (Margaritis and Thrun 1999), and others (Tsamardinos et al. 2003) that can recover the Mbs of a set of random variables. Initially, any of the aforementioned algorithms could be used in **ComputeMb** to find $Mb_{\mathbf{V}}$ and pass it to the **RSL**. After eliminating a removable vertex X, the Mbs of the remaining graph will change. Therefore, we need to update and pass $Mb_{\overline{\mathbf{V}}\setminus\{X\}}$ to the next recall of **RSL**. This is done by function **UpdateMb** in line 7 of Algorithm 1. We propose Algorithm 2 for **UpdateMb** and prove its soundness and complexity in Proposition 2. Further discussion about this algorithm is presented in Appendix D.

Proposition 2. *Suppose $\mathcal{G}[\overline{\mathbf{V}}]$ is a perfect map of $P_{\overline{\mathbf{V}}}$ and X is a removable variable in $\mathcal{G}[\overline{\mathbf{V}}]$. Algorithm 2 correctly finds $Mb_{\overline{\mathbf{V}}\setminus\{X\}}$ by performing at most $\binom{|N_{\mathcal{G}[\overline{\mathbf{V}}]}(X)|}{2}$ CI tests.*

Learning BN with Known Upper Bound on the Clique Number

In this section, we consider the BN structure learning problem when we are given an upper bound m on the clique number of the underlying BN and propose algorithms 3 and 4 to efficiently find removable vertices along with their neighbors.

Algorithm 2: Updates Markov boundaries (Mbs).

```
UpdateMb($X$, $P_{\overline{\mathbf{V}}}$, $N_{\mathcal{G}[\overline{\mathbf{V}}]}(X)$, $Mb_{\overline{\mathbf{V}}}$)
$Mb_{\overline{\mathbf{V}}\setminus\{X\}} \leftarrow (Mb_{\overline{\mathbf{V}}}(Y)\colon\ Y \in \overline{\mathbf{V}} \setminus \{X\})$
for $Y \in Mb_{\overline{\mathbf{V}}}(X)$ do
    Remove $X$ from $Mb_{\overline{\mathbf{V}}\setminus\{X\}}(Y)$.
if $N_{\mathcal{G}[\overline{\mathbf{V}}]}(X) = Mb_{\overline{\mathbf{V}}}(X)$ then
    for $Y, Z \in N_{\mathcal{G}[\overline{\mathbf{V}}]}(X)$ do
        if $Y \perp\!\!\!\perp_{P_{\overline{\mathbf{V}}}} Z|Mb_{\overline{\mathbf{V}}\setminus\{X\}}(Y) \setminus \{Y, Z\}$ then
            Remove $Z$ from $Mb_{\overline{\mathbf{V}}\setminus\{X\}}(Y)$
            Remove $Y$ from $Mb_{\overline{\mathbf{V}}\setminus\{X\}}(Z)$
return $Mb_{\overline{\mathbf{V}}\setminus\{X\}}$
```

We denote the resulting **RSL** with these implementations of **FindRemovable** and **FindNeighbors** by RSL$_\omega$. First, we present a sufficient removability condition in such networks, which is the foundation of Algorithm 3.

Lemma 1. *Suppose $\mathcal{G} = (\overline{\mathbf{V}}, \mathbf{E})$ is a DAG and a perfect map of $P_{\overline{\mathbf{V}}}$ such that $\omega(\mathcal{G}) \leq m$. Vertex $X \in \overline{\mathbf{V}}$ is removable in $\mathcal{G}$ if for any $\mathbf{S} \subseteq Mb_{\overline{\mathbf{V}}}(X)$ with $|\mathbf{S}| \leq m - 2$, we have*

$$\begin{aligned}&\forall Y, Z \in Mb_{\overline{\mathbf{V}}}(X) \setminus \mathbf{S}\colon\\ &\qquad Y \not\perp\!\!\!\perp_{P_{\overline{\mathbf{V}}}} Z|\big(Mb_{\overline{\mathbf{V}}}(X) \cup \{X\}\big) \setminus \big(\{Y, Z\} \cup \mathbf{S}\big),\\ &\textit{and } \forall Y \in Mb_{\overline{\mathbf{V}}}(X) \setminus \mathbf{S}\colon\\ &\qquad X \not\perp\!\!\!\perp_{P_{\overline{\mathbf{V}}}} Y|Mb_{\overline{\mathbf{V}}}(X) \setminus (\{Y\} \cup \mathbf{S}).\end{aligned} \qquad (3)$$

Also, the set of vertices that satisfy Equation (3) *is nonempty.*

Algorithm 3 first sorts the vertices in $\overline{\mathbf{V}}$ based on their Mb size and checks their removability, starting with the vertex with the smallest Mb. This ensures that both the number of CI tests and the size of the conditioning sets remain bounded.

Proposition 3. *Suppose $\mathcal{G}[\overline{\mathbf{V}}]$ is a DAG and a perfect map of $P_{\overline{\mathbf{V}}}$ s.t. $\omega(\mathcal{G}[\overline{\mathbf{V}}]) \leq m$. Algorithm 3 returns a removable vertex in $\mathcal{G}[\overline{\mathbf{V}}]$ by performing $\mathcal{O}(|\overline{\mathbf{V}}|\Delta_{in}(\mathcal{G}[\overline{\mathbf{V}}])^m)$ CI tests.*

We now turn to the function **FindNeighbors**. Recall that the purpose of this function is to find the neighbors of a removable vertex X and its separating sets. Since for every vertex $Y \notin Mb_{\overline{\mathbf{V}}}(X)$, we have $Y \perp\!\!\!\perp_{P_{\overline{\mathbf{V}}}} X|Mb_{\overline{\mathbf{V}}}(X)$, $Mb_{\overline{\mathbf{V}}}(X)$ is a separating set for all vertices outside of $Mb_{\overline{\mathbf{V}}}(X)$. Therefore, it suffices to find the non-neighbors of X within $Mb_{\overline{\mathbf{V}}}(X)$ or equivalently the co-parents of X. Next result characterizes the co-parents of a removable vertex X.

Algorithm 3: Finds a removable vertex.

```
FindRemovable($\overline{\mathbf{V}}$, $P_{\overline{\mathbf{V}}}$, $Mb_{\overline{\mathbf{V}}}$, SideInfo (m))
$\mathbf{X} = (X_1, ..., X_{|\overline{\mathbf{V}}|}) \leftarrow \overline{\mathbf{V}}$
Sort $\mathbf{X}$ s.t.
    $|Mb_{\overline{\mathbf{V}}}(X_1)| \leq |Mb_{\overline{\mathbf{V}}}(X_2)| \cdots \leq |Mb_{\overline{\mathbf{V}}}(X_{|\overline{\mathbf{V}}|})|$.
for $i = 1$ to $|\overline{\mathbf{V}}|$ do
    if (3) holds for $X = X_i$ then
        return $X_i$
```

Algorithm 4: Finds neighbors and separating sets in a graph with bounded clique number.

```
FindNeighbors($X$, $\overline{\mathbf{V}}$, $P_{\overline{\mathbf{V}}}$, $Mb_{\overline{\mathbf{V}}}(X)$, SideInfo(m))
for $Y \in \overline{\mathbf{V}} \setminus Mb_{\overline{\mathbf{V}}}(X)$ do
    Add $\langle X|Mb_{\overline{\mathbf{V}}}(X)|Y\rangle$ to $\boldsymbol{\mathcal{S}}_X$.
for $Y \in Mb_{\overline{\mathbf{V}}}(X)$ do
    if (4) holds then
        Add $\langle X|Mb_{\overline{\mathbf{V}}}(X) \setminus \{Y, Z\}|Y\rangle$ to $\boldsymbol{\mathcal{S}}_X$.
    else
        Add $Y$ to $N_{\mathcal{G}[\overline{\mathbf{V}}]}(X)$.
return $(N_{\mathcal{G}[\overline{\mathbf{V}}]}(X), \boldsymbol{\mathcal{S}}_X)$
```

Lemma 2. *Suppose $\mathcal{G}[\overline{\mathbf{V}}]$ is a DAG and a perfect map of $P_{\overline{\mathbf{V}}}$ with $\omega(\mathcal{G}[\overline{\mathbf{V}}]) \leq m$. Let $X \in \overline{\mathbf{V}}$ be a vertex that satisfies Equation* (3) *and $Y \in Mb_{\overline{\mathbf{V}}}(X)$. Then, $Y \in CP_{\mathcal{G}}(X)$ iff*

$$\exists \mathbf{S} \subseteq Mb_{\overline{\mathbf{V}}}(X) \setminus \{Y\}: \\ |\mathbf{S}| = (m-1), \;\; X \perp\!\!\!\perp_{P_{\overline{\mathbf{V}}}} Y | Mb_{\overline{\mathbf{V}}}(X) \setminus (\{Y\} \cup \mathbf{S}). \quad (4)$$

Algorithm 4 is designed based on Lemma 2. We use $\langle X|\mathbf{Z}|Y\rangle$ to denote that $\mathbf{Z}$ is a separating set for X and Y.

Theorem 1. *Suppose $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a DAG and a perfect map of $P_{\mathbf{V}}$ with $\omega(\mathcal{G}) \leq m$. Then,* ***RSL*** *(Algorithm 1) with sub-algorithms 3 and 4 is sound and complete, and performs $\mathcal{O}(|\mathbf{V}|^2 \Delta_{in}(\mathcal{G})^m)$ CI tests.*

Learning BN Without Side Information

We showed in Theorem 1 that if the upper bound on the clique number is correct, i.e., $\omega(\mathcal{G}) \leq m$, then RSL$_\omega$ learns the DAG correctly. But what happens if $\omega(\mathcal{G}) > m$? In this case, there are two possibilities: either Algorithm 3 fails to find any removables and consequently, RSL$_\omega$ fails or RSL$_\omega$ terminates with output $(\tilde{\mathcal{H}}, \boldsymbol{\mathcal{S}}_{\mathbf{V}})$. Next result shows that the clique number of $\tilde{\mathcal{H}}$ is greater or equal to $\omega(\mathcal{G})$ and thus, it is strictly larger than m.

Proposition 4 (Verifiable)**.** *Suppose $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a DAG with skeleton $\mathcal{H}$ that is a perfect map of $P_{\mathbf{V}}$. If the* ***RSL*** *with sub-algorithms 3 and 4, and input $m > 0$ terminates, then the clique number of the learned skeleton is at least $\omega(\mathcal{G})$.*

This result implies that executing RSL$_\omega$ with input m either outputs a graph with clique number at most m, which is guaranteed to be the true BN, or indicates that the upper bound m is incorrect. As a result, we can design Algorithm 5 using RSL$_\omega$ when no bound on the clique number is given.

Algorithm 5: Learns BN without side information.

```
Input: $\mathbf{V}$, $P_{\mathbf{V}}$
$Mb_{\mathbf{V}} \leftarrow$ ComputeMb($\mathbf{V}$, $P_{\mathbf{V}}$)
for $m$ from 1 to $|\mathbf{V}|$ do
    $\hat{\mathcal{G}} \leftarrow$ RSL($\mathbf{V}$, $P_{\mathbf{V}}$, $Mb_{\mathbf{V}}$, SideInfo(m))
    if RSL terminates and $\omega(\hat{\mathcal{G}}) \leq m$ then
        return $\hat{\mathcal{G}}$
```

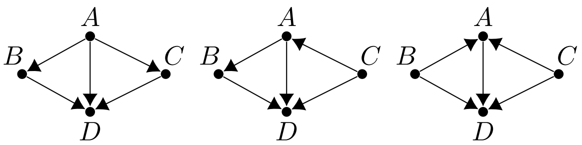


Figure 1: Diamond graphs.

Learning Diamond-free BNs

In this section, we consider a well-studied class of graphs, namely diamond-free graphs. These graphs appear in many real-world applications (see Appendix F). Diamond-free graphs also occur with high probability in a wide range of random graphs. For instance, an Erdos-Renyi graph G(n,p) is diamond-free with high probability, if $pn^{0.8} \to 0$ (See Lemma 5.) Various NP-hard problems such as maximum weight stable set, maximum weight clique, domination and coloring have been shown to be linearly or polynomially solvable for diamond-free graphs (Brandstädt 2004; Dabrowski, Dross, and Paulusma 2017). We show that the structure learning problem for diamond-free graphs is also polynomial-time solvable.

Definition 4 (diamond-free graphs)**.** *The graphs depicted in Figure 1 are called diamonds. A diamond-free graph is a graph that contains no diamond as an induced subgraph.*

Note that triangle-free graphs are a subset of diamond-free graphs. Recall that RSL$_\omega$ with $m = 2$ can lean a triangle-free BN with complexity $\mathcal{O}(|\mathbf{V}|^2 \Delta_{in}(\mathcal{G})^2)$. Herein, we propose new subroutines for **FindRemovable** and **FindNeighbors** with which, RSL can learn diamond-free BNs with the same complexity as triangle-free networks. We start with providing a necessary and sufficient condition for removability in a diamond-free graph.

Lemma 3. *Suppose $\mathcal{G} = (\overline{\mathbf{V}}, \mathbf{E})$ is a diamond-free DAG and a perfect map of $P_{\overline{\mathbf{V}}}$. Vertex $X \in \overline{\mathbf{V}}$ is removable in $\mathcal{G}$ if and only if $\forall Y, Z \in Mb_{\overline{\mathbf{V}}}(X)$:*

$$Y \not\perp\!\!\!\perp_{P_{\overline{\mathbf{V}}}} Z | (Mb_{\overline{\mathbf{V}}}(X) \cup \{X\}) \setminus \{Y, Z\}. \quad (5)$$

Furthermore, the set of removable vertices is nonempty.

Based on Lemma 3, the pseudocode for **FindRemovable** function is identical to Algorithm 3, except that it gets the diamond-freeness as input instead of m and it checks for (5) instead of (3) in line 5.

Similar to RSL$_\omega$, we have the following result.

Proposition 5. *Suppose $\mathcal{G}[\overline{\mathbf{V}}]$ is a diamond-free DAG and a perfect map of $P_{\overline{\mathbf{V}}}$.* ***FindRemovable*** *returns a removable vertex in $\mathcal{G}[\overline{\mathbf{V}}]$ by performing at most $|\overline{\mathbf{V}}|\binom{\Delta_{in}(\mathcal{G}[\overline{\mathbf{V}}])}{2}$ CI tests.*

Analogous to the case with bounded clique number, the next result characterizes the co-parents of a removable vertex in a diamond-free graph.

Lemma 4. *Suppose $\mathcal{G} = (\overline{\mathbf{V}}, \mathbf{E})$ is a diamond-free DAG and a perfect map of $P_{\overline{\mathbf{V}}}$. Let $X \in \overline{\mathbf{V}}$ be a removable vertex in $\mathcal{G}$, and $Y \in Mb_{\overline{\mathbf{V}}}(X)$. In this case, $Y \in CP_{\mathcal{G}}(X)$ if and only if*

$$\exists Z \in Mb_{\overline{\mathbf{V}}}(X) \setminus \{Y\}: \;\; X \perp\!\!\!\perp_{P_{\overline{\mathbf{V}}}} Y | Mb_{\overline{\mathbf{V}}}(X) \setminus \{Y, Z\}. \quad (6)$$

Accordingly, **FindNeighbors** is identical to Algorithm 4, except that diamond-freeness is input to it rather than m and it checks for (6) instead of (4) in line 5.

Theorem 2. *Suppose $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a diamond-free DAG and a perfect map of $P_{\mathbf{V}}$. RSL_D is sound and complete, and performs $\mathcal{O}(|\mathbf{V}|^2 \Delta_{in}(\mathcal{G})^2)$ CI tests.*

A limitation of RSL_D is that diamond-freeness is not verifiable, unlike a bound on the clique number. However, even if the BN has diamonds, RSL_D correctly recovers all the existing edges with possibly extra edges, i.e., RSL_D has no false negative (see Appendix C for details.) Further, as we shall see in our experiments, RSL_D achieves the best accuracy in almost all cases in practice, even when BNs have diamonds.

Discussion

Complexity analysis: Theorems 1 and 2 present the maximum number of CI tests required by Algorithm 1 to learn a DAG with bounded clique number and a diamond-free DAG, respectively. However, this algorithm may perform a CI test several times. We present an implementation of **RSL** in Appendix E that avoids such unnecessary duplicate tests (by keeping track of the performed CI tests, using mere logarithmic memory space) and achieves $\mathcal{O}(|\mathbf{V}|\Delta_{in}(\mathcal{G})^3)$ and $\mathcal{O}(|\mathbf{V}|\Delta_{in}(\mathcal{G})^{m+1})$ CI tests in diamond-free graphs and those with bounded clique number, respectively. Recall that Algorithm 1 initially takes $Mb_{\mathbf{V}}$ as an input, and finding the Mbs requires an additional $\mathcal{O}(|\mathbf{V}|^2)$ number of CI tests.

Due to the recursive nature of **RSL**, the size of conditioning sets in each iteration reduces. Furthermore, since the size of the Mb of a removable variable is bounded by the maximum in-degree[7], **RSL** performs CI tests with small conditioning sets. Having small conditioning sets in each CI test is essential to reduce sample complexity of the learning task. In our experiments, we empirically show that our proposed algorithms outperform the state of the art by having both lower number of CI tests and smaller conditioning sets.

Random BNs: As discussed earlier, diamond-free graphs or BNs with bounded clique numbers appear in some specific applications. Herein, we show that such structures also appear with high probability in networks whose edges appear independently and therefore, are essentially realizations of Erdos-Renyi graphs (Erdős and Rényi 1960).

Lemma 5. *A random graph $\mathcal{G}$ generated from Erdos-Renyi model $G(n, p)$ is diamond-free with high probability when $pn^{0.8} \to 0$ and $\omega(\mathcal{G}) \leq m$ when $pn^{2/m} \to 0$.*

Experiment

In this section, we present a set of experiments to illustrate the effectiveness of our proposed algorithms[8]. We compare the performance of RSL_D and RSL_ω with MARVEL (Mokhtarian et al. 2021), a modified version of PC (Spirtes et al. 2000; Pellet and Elisseeff 2008) that uses Mbs, GS (Margaritis and Thrun 1999), CS (Pellet and Elisseeff 2008), and MMPC (Tsamardinos, Aliferis, and Statnikov 2003b) on both real-world structures and Erdos-Renyi random graphs.

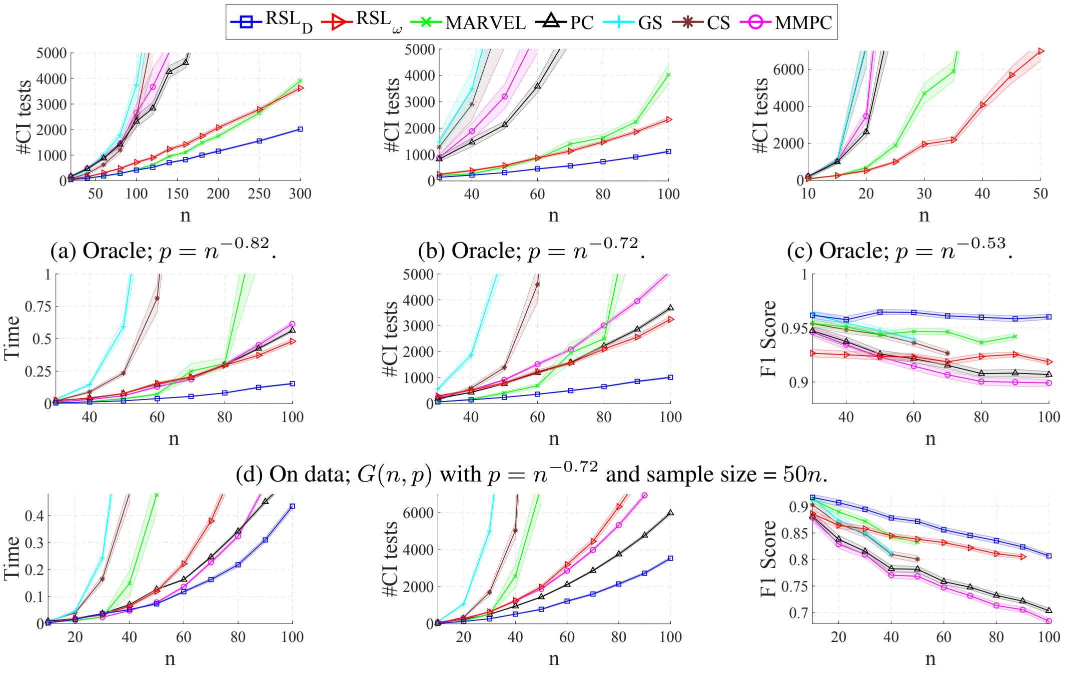
All aforementioned algorithms are Mb based. Thus, we initially use TC (Pellet and Elisseeff 2008) algorithm to compute $Mb_{\mathbf{V}}$, and then pass it to each of the methods for the sake of fair comparison. The algorithms are compared in two settings: I) oracle, and II) finite sample. In the oracle setting, we are working in the population level, i.e., the CI tests are queried through an oracle that has access to the true CI relations among the variables. In the latter setting, algorithms have access to a dataset of finite samples from the true distribution. Hence, the CI tests might be noisy. The samples are generated using a linear model where each variable is a linear combination of its parents plus an exogenous noise variable; the coefficients are chosen uniformly at random from $[-1.5, -1] \cup [1, 1.5]$, and the noises are generated from $\mathcal{N}(0, \sigma^2)$, where σ is selected uniformly at random from $[\sqrt{0.5}, \sqrt{1.5}]$. As for the CI tests, we use Fisher Z-transformation (Fisher 1915) with significance level 0.01 in the algorithms (alternative values did not alter our experimental results) and $\frac{2}{n^2}$ for Mb discovery (Pellet and Elisseeff 2008). These are standard evaluations' scenarios often performed in the structure learning literature (Colombo and Maathuis 2014; Améndola et al. 2020; Mokhtarian et al. 2021; Huang et al. 2012; Ghahramani and Beal 2001; Scutari, Vitolo, and Tucker 2019). We compare the algorithms in terms of runtime, the number of performed CI tests, and the f1-scores of the learned skeletons. In Appendix F, we further report other measurements (average size of conditioning sets, precision, recall, structural hamming distance) of the learned skeletons, and accuracy of the learned separating sets.

Figure 2 illustrates the performance of BN learning algorithms on random Erdos-Renyi $G(n, p)$ model graphs. Each point is reported as the average of 100 runs, and the shaded areas indicate the 80% confidence intervals. Runtime and the number of CI tests are reported after Mb discovery. Figures 2a, 2b and 2c demonstrate the number of CI tests each algorithm performed in the oracle setting, for the values of $p = n^{-0.82}$,$n^{-0.72}$, and $n^{-0.53}$, respectively. In 2a, the graphs are diamond-free with high probability (see Discussion for details). In 2d, $\omega \leq 3$ with high probability, but the graphs are not necessarily diamond-free. In 2c, $\omega \leq 4$, with high probability. We have not included the result of RSL_D in Figure 2c, as the graphs contain diamonds with high probability, and RSL_D has no theoretical guarantee despite of low complexity. Figures 2d and 2e demonstrate the performance of the algorithms in the finite sample setting, when $50n$ and $20n$ samples were available, respectively. Although RSL_D does not have any theoretical correctness guarantee to recover the network (graphs are not diamond-free), both RSL_D and RSL_ω outperform other algorithms in terms of both accuracy and computational complexity in most cases. The lower runtime of MARVEL and MMPC compared to RSL_ω in Figure 2e can be explained through their significantly low accuracy due to skipping numerous CI tests.

Figure 3 illustrates the performance of BN learning algorithms on two real-world structures, namely Diabetes (Andreassen et al. 1991) and Andes (Conati et al. 1997) networks, over a range of different sample sizes. Each point is reported

[7] See Lemma 6 in Appendix B.

[8] The MATLAB implementation of our algorithms is publicly available at https://github.com/Ehsan-Mokhtarian/RSL.



(a) Oracle; $p = n^{-0.82}$.

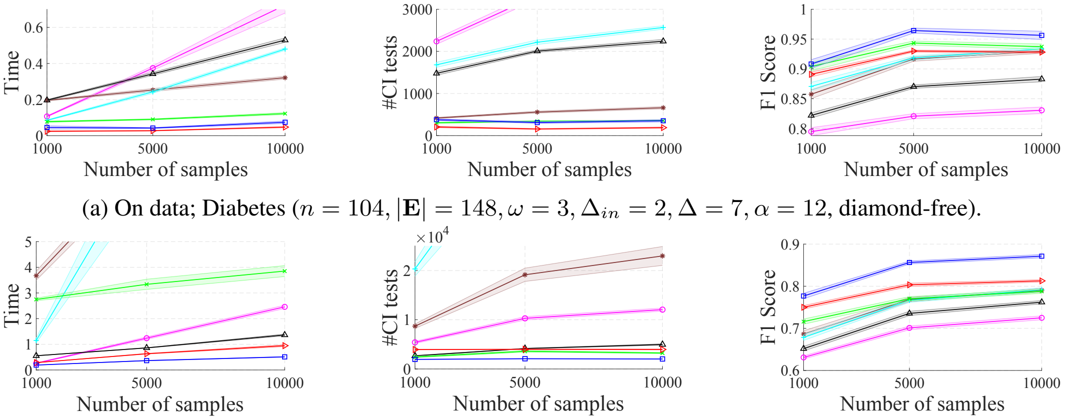
(b) Oracle; $p = n^{-0.72}$.

(c) Oracle; $p = n^{-0.53}$.

(d) On data; $G(n, p)$ with $p = n^{-0.72}$ and sample size = $50n$.

(e) On data; $G(n, p)$ with $p = n^{-0.65}$ and sample size = $20n$.

Figure 2: Performance of various algorithms on random graphs generated from Erdos-Renyi models.



(a) On data; Diabetes ($n = 104, |\mathbf{E}| = 148, \omega = 3, \Delta_{in} = 2, \Delta = 7, \alpha = 12$, diamond-free).

(b) On data; Andes ($n = 223, |\mathbf{E}| = 328, \omega = 3, \Delta_{in} = 6, \Delta = 12, \alpha = 23$, not diamond-free).

Figure 3: Performance of various algorithms on real-world structures.

as the average of 10 runs. As seen in Figures 3a and 3b, both RSL algorithms outperform other algorithms in both accuracy and complexity. Note that although Andes is not a diamond-free graph, RSL$_D$ achieves the best accuracy in Figure 3b. Similar experimental results for five real-world structures in both oracle and finite sample settings along with detailed information about these structures appear in Appendix F.

Conclusion

In this work, we presented the RSL algorithm for BN structure learning. Although our generic algorithm has exponential complexity, we showed that it could harness structural side information to learn the BN structure in polynomial time. In particular, we considered two types of side information about the underlying BN: I) when an upper bound on its clique number is known, and II) when the BN is diamond-free. We provided theoretical guarantees and upper bounds on the number of CI tests required by our algorithms. We demonstrated the superiority of our proposed algorithms in both synthetic and real-world structures. We showed in the experiments that even when the graph is not diamond-free, RSL$_D$ outperforms various algorithms both in time complexity and accuracy.

Acknowledgments

The work presented in this paper was in part supported by Swiss National Science Foundation (SNSF) under grant number 200021_20435.

References

Améndola, C.; Dettling, P.; Drton, M.; Onori, F.; and Wu, J. 2020. Structure Learning for Cyclic Linear Causal Models. In *Conference on Uncertainty in Artificial Intelligence*, 999–1008. PMLR.

Andreassen, S.; Hovorka, R.; Benn, J.; Olesen, K. G.; and Carson, E. R. 1991. A model-based approach to insulin adjustment. In *AIME 91*, 239–248. Springer.

Bartlett, M.; and Cussens, J. 2017. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244: 258–271.

Bodlaender, H. L.; and Möhring, R. H. 1993. The pathwidth and treewidth of cographs. *SIAM Journal on Discrete Mathematics*, 6(2): 181–188.

Brandstädt, A. 2004. (P5, diamond)-free graphs revisited: structure and linear time optimization. *Discrete applied mathematics*, 138(1-2): 13–27.

Chen, E. Y.-J.; Shen, Y.; Choi, A.; and Darwiche, A. 2016. Learning Bayesian networks with ancestral constraints. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2333–2341. Citeseer.

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

Chickering, M.; Heckerman, D.; and Meek, C. 2004. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5.

Claassen, T.; M. Mooij, J.; and Heskes, T. 2013. Learning Sparse Causal Models is not NP-hard. *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*.

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

Conati, C.; Gertner, A. S.; VanLehn, K.; and Druzdzel, M. J. 1997. On-line student modeling for coached problem solving using Bayesian networks. In *User Modeling*, 231–242. Springer.

Dabrowski, K. K.; Dross, F.; and Paulusma, D. 2017. Colouring diamond-free graphs. *Journal of computer and system sciences*, 89: 410–431.

de Morais, S. R.; and Aussem, A. 2010. An Efficient and Scalable Algorithm for Local Bayesian Network Structure Discovery. In *Machine Learning and Knowledge Discovery in Databases*, 164–179.

Elidan, G.; and Gould, S. 2008. Learning Bounded Treewidth Bayesian Networks. *Journal of Machine Learning Research*, 9(12).

Erdős, P.; and Rényi, A. 1960. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5: 17–61.

Fisher, R. A. 1915. Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, 10(4): 507–521.

Flores, M. J.; Nicholson, A. E.; Brunskill, A.; Korb, K. B.; and Mascaro, S. 2011. Incorporating expert knowledge when learning Bayesian network structure: a medical case study. *Artificial intelligence in medicine*, 53(3): 181–204.

Friedman, N.; Linial, M.; Nachman, I.; and Pe'er, D. 2000. Using Bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4): 601–620.

Ghahramani, Z.; and Beal, M. J. 2001. Propagation algorithms for variational Bayesian learning. *Advances in neural information processing systems*, 507–513.

Grüttemeier, N.; and Komusiewicz, C. 2020. Learning Bayesian Networks Under Sparsity Constraints: A Parameterized Complexity Analysis. *arXiv preprint arXiv:2004.14724*.

Huang, S.; Li, J.; Ye, J.; Fleisher, A.; Chen, K.; Wu, T.; Reiman, E.; Initiative, A. D. N.; et al. 2012. A sparse structure learning algorithm for gaussian bayesian network identification from high-dimensional data. *IEEE transactions on pattern analysis and machine intelligence*, 35(6): 1328–1342.

Korhonen, J.; and Parviainen, P. 2013. Exact learning of bounded tree-width Bayesian networks. In *Artificial Intelligence and Statistics*, 370–378. PMLR.

Korhonen, J. H.; and Parviainen, P. 2015. Tractable Bayesian network structure learning with bounded vertex cover number. *Advances in Neural Information Processing Systems*, 28: 622–630.

Li, A.; and van Beek, P. 2018. Bayesian Network Structure Learning with Side Constraints. In *Proceedings of the Ninth International Conference on Probabilistic Graphical Models*, Proceedings of Machine Learning Research, 225–236.

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

McLachlan, S.; Dube, K.; Hitman, G. A.; Fenton, N. E.; and Kyrimi, E. 2020. Bayesian networks in healthcare: Distribution by medical condition. *Artificial Intelligence in Medicine*, 107: 101912.

Mokhtarian, E.; Akbari, S.; Ghassami, A.; and Kiyavash, N. 2021. A Recursive Markov Boundary-Based Approach to Causal Structure Learning. In *The KDD'21 Workshop on Causal Discovery*, 26–54. PMLR.

Nie, S.; Mauá, D. D.; De Campos, C. P.; and Ji, Q. 2014. Advances in learning Bayesian networks of bounded treewidth. *Advances in neural information processing systems*, 27: 2285–2293.

Oyen, D.; Anderson, B.; and Anderson-Cook, C. M. 2016. Bayesian Networks with Prior Knowledge for Malware Phylogenetics. In *AAAI Workshop: Artificial Intelligence for Cyber Security*.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

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

Pellet, J.-P.; and Elisseeff, A. 2008. Using Markov blankets for causal structure learning. *Journal of Machine Learning Research*, 9(Jul): 1295–1342.

Pollino, C. A.; Woodberry, O.; Nicholson, A.; Korb, K.; and Hart, B. T. 2007. Parameterisation and evaluation of a Bayesian network for use in an ecological risk assessment. *Environmental Modelling & Software*, 22(8): 1140–1152.

Ramaswamy, V. P.; and Szeider, S. 2021. Turbocharging Treewidth-Bounded Bayesian Network Structure Learning. In *Proceeding of AAAI-21, the Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Scutari, M. 2014. Bayesian network constraint-based structure learning algorithms: Parallel and optimised implementations in the bnlearn R package. *arXiv preprint arXiv:1406.7648*.

Scutari, M.; Vitolo, C.; and Tucker, A. 2019. Learning Bayesian networks from big data with greedy search: computational complexity and efficient implementation. *Statistics and Computing*, 29(5): 1095–1108.

Sesen, M. B.; Nicholson, A. E.; Banares-Alcantara, R.; Kadir, T.; and Brady, M. 2013. Bayesian networks for clinical decision support in lung cancer care. *PloS one*, 8(12): e82349.

Shimizu, S. 2019. Non-Gaussian methods for causal structure learning. *Prevention Science*, 20(3): 431–441.

Solus, L.; Wang, Y.; and Uhler, C. 2017. Consistency guarantees for greedy permutation-based causal inference algorithms. *arXiv preprint arXiv:1702.03530*.

Sondhi, A.; and Shojaie, A. 2019. The Reduced PC-Algorithm: Improved Causal Structure Learning in Large Random Networks. *Journal of Machine Learning Research*, 20(164): 1–31.

Spirtes, P.; Glymour, C. N.; Scheines, R.; and Heckerman, D. 2000. *Causation, prediction, and search*. MIT press.

Takeishi, N.; and Kawahara, Y. 2020. Knowledge-Based Regularization in Generative Modeling. In *29th International Joint Conference on Artificial Intelligence, IJCAI 2020*, 2390–2396. International Joint Conferences on Artificial Intelligence.

Tsamardinos, I.; Aliferis, C. F.; and Statnikov, A. 2003a. Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 673–678.

Tsamardinos, I.; Aliferis, C. F.; and Statnikov, A. 2003b. Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 673–678.

Tsamardinos, I.; Aliferis, C. F.; Statnikov, A. R.; and Statnikov, E. 2003. Algorithms for large scale Markov blanket discovery. In *FLAIRS conference*, volume 2, 376–380.

Tsamardinos, I.; Brown, L. E.; and Aliferis, C. F. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning*, 65(1): 31–78.

Zheng, X.; Aragam, B.; Ravikumar, P. K.; and Xing, E. P. 2018. DAGs with NO TEARS: Continuous Optimization for Structure Learning. *Advances in Neural Information Processing Systems*, 31.

Zheng, X.; Dan, C.; Aragam, B.; Ravikumar, P.; and Xing, E. 2020. Learning sparse nonparametric DAGs. In *International Conference on Artificial Intelligence and Statistics*, 3414–3425. PMLR.

Zhu, S.; Ng, I.; and Chen, Z. 2020. Causal discovery with reinforcement learning. *ICLR*.