Adaptive Skeleton Construction for Accurate DAG Learning

Xianjie Guo, Kui Yu*, Lin Liu, Peipei Li, and Jiuyong Li

Abstract—Directed acyclic graph (DAG) learning plays a key role in causal discovery and many machine learning tasks. Learning a DAG from high-dimensional data always faces scalability problems. A local-to-global DAG learning approach can be scaled to high-dimensional data, however, existing local-to-global DAG learning algorithms employ either the AND-rule or the OR-rule for constructing a DAG skeleton. Simply using either rule, existing local-to-global methods may learn an inaccurate DAG skeleton, leading to unsatisfactory DAG learning performance. To tackle this problem, in this paper, we propose an Adaptive DAG Learning (ADL) algorithm. The novel contribution of ADL is that it can simultaneously and adaptively use the AND-rule and the OR-rule to construct an accurate global DAG skeleton. We conduct extensive experiments on both benchmark and real-world datasets, and the experimental results show that ADL is significantly better than some existing local-to-global and global DAG learning algorithms.

Index Terms—Directed acyclic graph, Causal discovery, Local-to-global structure learning, Local skeleton learning

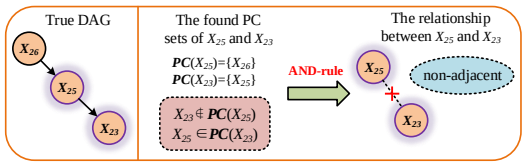
1 Introduction

Directed Acyclic Graph (DAG) learning as the foundation of a Bayesian network (BN) plays a vital part in causal discovery [1], [2], explainable model [3], [4], and many machine learning applications [5], [6]. In a DAG, nodes denote variables while edges represent the dependence relationship between the variables [7]. For instance, an edge $X_1 \rightarrow X_2$ in a DAG represents that X_1 is a parent of X_2 and X_2 is a child of X_1.

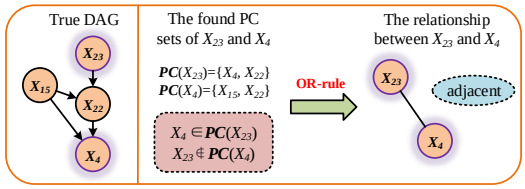
Although many algorithms have been proposed for DAG learning, existing methods still face scalability or inaccuracy problems [8]. Thus developing accurate and efficient DAG learning algorithms is one of the research hotspots in causal discovery and machine learning [9]. Early research efforts focus on learning an entire DAG at once (i.e. global DAG learning methods), but almost all those global learning methods are not scalable to high-dimensional data [10], [11]. Although recent neural DAG learning provide a new research line for DAG learning, these algorithms attempt to learn an entire DAG and they still face computational issues [12], [13].

To alleviate the computational complexity of existing global DAG learning methods, a local-to-global approach has been proposed [14], [15]. The local-to-global approach first learns a local DAG skeleton of each variable. A local DAG skeleton often refers to the parents and children of a variable in a DAG. Then it constructs the global DAG skeleton using the learned local DAG skeletons, and finally orients edges in this skeleton. Existing local-to-global learning approaches can be scaled to high-dimensional data.

However, those local-to-global methods may not ensure accurate DAG skeleton. In a DAG, if there is an edge between X_1 and X_2, X_1 should be in the parent and child (PC) set of X_2 and X_2 also should be in PC set of X_1. Using this symmetrical property and the learned PC sets of the two variables, we can determine whether there is an edge between two variables in a underlying DAG. Due to data noise or small data samples, in the local DAG skeleton learning phase, it always appears that X_1 (or X_2) is in the learned PC set of X_2 (or X_1) but X_2 (or X_1) is not in the learned PC set of X_1 (or X_2). In this case, it is hard to determine whether there is an edge between X_1 and X_2.



(a) Using the AND-rule, MMHC wrongly considers the two adjacent variables as two non-adjacent ones.



(b) Using the OR-rule, SLL+C wrongly considers the two non-adjacent variables as two adjacent ones.

Fig. 1. Either the AND-rule or the OR-rule may lead to an inaccurate DAG skeleton

To tackle this issue, existing local-to-global DAG learning approaches employ either the AND-rule or the OR-rule (please see Definitions 3.6 and 3.7 in Section 3) for constructing DAG skeletons [15], [16]. Specifically, if X_1 is

- *X. Guo, K. Yu, and P. Li are with the Key Laboratory of Knowledge Engineering With Big Data of Ministry of Education, Hefei University of Technology, Hefei 230601, China, also with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China; e-mail: xianjieguo@mail.hfut.edu.cn, {yukui,peipeili}@hfut.edu.cn (*Corresponding author: Kui Yu).*
- *L. Liu and J. Li are with the UniSA STEM, University of South Australia, Adelaide 5095, Australia; e-mail: {Lin.Liu,Jiuyong.Li}@unisa.edu.au.*

*Manuscript received * *, *; revised * *, *.*

in the learned PC set of X_2 but X_2 is not in the learned PC set of X_1, using the AND-rule, existing algorithms consider that X_1 and X_2 have not an edge between them, whereas employing the OR-rule, they determine that there is an edge between X_1 and X_2. However, simply using either rule, these existing learning algorithms may learn an inaccurate DAG skeleton, resulting in inaccurate DAGs.

Fig. 1 gives an example to demonstrate the shortcomings of existing local-to-global DAG learning algorithms. In this example, we first choose the frequently used ALARM Bayesian network for generating a synthetic dataset $\mathcal{D}_{1000\times 37}$ (1000 data samples and 37 variables). Then using the dataset, we run two well-established local-to-global algorithms, MMHC using the AND-rule [15] and SLL+C using the OR-rule [16], as shown in Fig. 1.

Fig. 1(a) shows that using MMHC to learn the PC set of each variable, X_{23} is not in the learned PC set of X_{25} but the learned PC set of X_{23} includes X_{25}. To tackle the issue, MMHC adopts the AND-rule to determine whether there is an edge between X_{25} and X_{23} or not. Thus MMHC wrongly determines that X_{25} and X_{23} are not adjacent. Using SSL+C, from Fig. 1(b), we find that X_4 is in the found PC set of X_{23} while the found PC set of X_4 does not include X_{23}. Using the OR-rule, the SSL+C algorithm wrongly considers that there is an edge between X_{23} and X_4.

Based on the observations discussed above, we find that if MMHC uses the OR-rule in Fig. 1(a) while SLL+C employs the AND-rule in Fig. 1(b), we will get the satisfactory results. Thus a question naturally arises: can we use both the AND-rule and the OR-rule simultaneously and adaptively determine which rule will be used during constructing a global DAG skeleton? Motivated by the question, we summarize the shortcomings of existing methods and our main contributions are shown in Fig. 2. Specifically, in this paper, our main contributions are as follows.

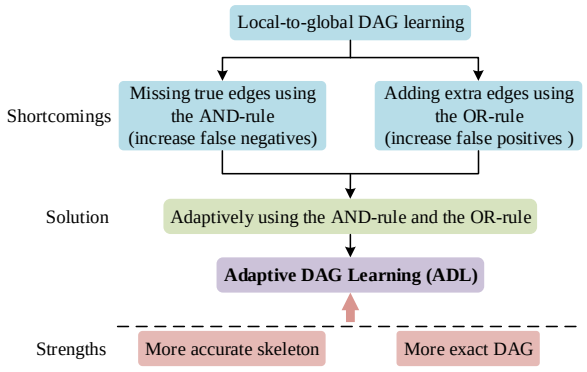


Fig. 2. Shortcomings of existing methods and our main contributions

- We propose a new Adaptive DAG Learning (ADL) algorithm for accurate local-to-global DAG learning. The main idea of ADL is that it designs a novel strategy to adaptively use both the AND-rule and the OR-rule to construct an accurate global skeleton using the learned local skeletons.
- We conduct extensive experiments using 11 benchmark BN datasets and a real-world dataset, to evaluate the proposed algorithm and compare it with the representative and state-of-the-art local-to-global and global DAG learning algorithms. Experimental results have shown that our proposed algorithm outperforms those algorithms compared.

The remainder of this paper is organized as follows. Section 2 reviews the related work and Section 3 gives the notations and definitions. Section 4 describes the proposed ADL algorithm in detail and Section 5 reports the experimental results. Section 6 summarizes the paper.

2 Related Work

Learning a DAG from observational data is crucial for data mining and machine learning [17]. In recent years, Many effective algorithms have been proposed for learning DAGs from observational data [18], and according to the learning scale, these algorithms can be divided into two types, global methods [12], [19] and local-to-global methods [14], [15].

2.1 Global DAG learning algorithms

Traditional global DAG learning algorithms are subdivided into two types: score-based and constraint-based methods [17]. Score-based methods use a scoring function to select the DAG with the best score in the space of DAGs [20], the space of equivalence classes of DAGs [21], or the space of node-ordering [19]. The problem of searching for the best-scored DAG is known to be NP-hard [8]. Fortunately, when a variable ordering of a DAG is given, the problem of finding the best-scored DAG consistent with this ordering is not NP-hard. For instance, the K2 algorithm [22] first takes a random ordering of variables as an input, and then selects a DAG that has the largest posterior probability based on a given dataset. Indeed, if the in-degrees of variables are bounded to k, then the best-scored DAG can be learned with the time complexity of $O(m^{k+1})$, where m is the number of variables. However, the effectiveness and efficiency of K2 are strongly dependent on the initial ordering of variables, and an incorrect initial variable ordering can lead to an incorrectly learned DAG. Recently, Behjati *et al.* attempted to derive a high-quality ordering of variables from a given dataset as an initial input, and proposed two improved K2 algorithms for DAG learning [23], [24].

Constraint-based methods determine the conditional independence relationship between variables by employing conditional independence (CI) tests from data and then construct a DAG that best fits those independence relationships [7]. Common constraint-based algorithms include Peter and Clark (PC) [25] and Fast Causal Inference (FCI) [26] algorithms. However, when there are insufficient sample sizes, the results of CI tests are not reliable, and an error in these CI tests has a cascading effect in the subsequent learning process. Accordingly, in practice, the PC and FCI algorithms may perform poorly. Colombo *et al.* consider the effect of mistaken CI tests arising from limited sample sizes, and point out that the output from the original PC and FCI algorithms is sensitive to the order in which CI tests are performed. Therefore, they modify the original PC and FCI algorithms and create the PC-stable and FCI-stable algorithms for removing this order dependence [10].

However, traditional global DAG learning methods formulate the DAG learning problem as a combinatorial optimization problem and depend on various local heuristics

for enforcing the acyclicity constraint. To avoid the combinatorial constraint, recently, Zheng *et al.* [12] formulate the DAG learning problem as a continuous optimization problem and provide a new research line for DAG learning. In the continuous optimization problem, the DAG is defined as a weighted adjacency matrix optimized by the variations of gradient descent. Based on this work, many continuous optimization based DAG learning methods have been proposed [13], [27], [28], [29], and they have tried different types of neural network models, loss functions and representations of weighted adjacency matrix to improve their performance.

2.2 Local-to-global DAG learning algorithms

In the global DAG learning algorithms mentioned above, traditional algorithms attempt to learn an entire DAG and the search space of DAGs is combinatorial and exponential with the number of variables, resulting in that learning an entire DAG at once is computationally infeasible when the number of variables is large. Likewise, continuous optimization based algorithms also face time-consuming issues since they frequently perform matrix operations, or employ deep neural network models.

To tackle this problem, local-to-global DAG structure learning methods have been proposed and consist of three phases: 1) local DAG skeleton learning; 2) global skeleton construction using either the AND-rule or the OR-rule; 3) edge orientation. Specifically, the first step aims to learn the local skeleton (e.g. PC (parents and children) or MB (Markov blanket)) of each variable independently, and local skeleton learning can significantly reduce the potential DAG search space [30], [31], [32]. Then, the second step utilizes the learned local skeleton to construct the global skeleton using either the AND-rule or the OR-rule. The last step is to orient the undirected edges in the skeleton using CI tests or score functions [20], [33].

In the past two decades, several local-to-global structure learning methods have been proposed. For instance, the GSBN algorithm [34] learns the MB of each variable and constructs the global skeleton based on the AND-rule. It uses CI tests to orient edges. The MMHC algorithm [15] first learns the PC of each variable using CI tests, then constructs the global skeleton using the AND-rule, and finally uses a score function to orient edges. SLL+C/G [16] first finds the MB of each variable using a score-based BN structure learning algorithm (called SLL), then constructs the global skeletons using the OR-rule, and finally SLL+C and SLL+G employ CI tests and score functions to orient edges in the skeleton, respectively. Instead of finding the PC set of each variable in advance, the GGSL algorithm [14] first randomly selects a variable and learns the local DAG structure around the variable using a score-based BN structure learning algorithm, then gradually expands the learned structure until the entire structure is learned.

However, just as we discussed in Section 1, simply using either the AND-rule or the OR-rule for constructing global skeletons, existing local-to-global learning methods may learn an inaccurate DAG skeleton, leading to a unsatisfactory DAG.

3 Notations and Definitions

In this section, we will briefly introduce some basic definitions and notations frequently used in this paper (see Table 1 for a summary of the notations).

TABLE 1
Summary of notations

Notation	Meanings
$\mathcal{D}$	a dataset with m random variables
$\mathcal{X}$	the set of m random variables in $\mathcal{D}$
X_i, X_j	a single variable in $\mathcal{X}$ $(i, j = 1, 2, ..., m)$
$\boldsymbol{S}$	a conditioning set within $\mathcal{X}$
$\mathbb{P}$	a joint probability distribution over $\mathcal{X}$
$\mathcal{G}$	a direct acyclic graph over $\mathcal{X}$
$X_i \not\perp\!\!\!\perp X_j \mid \boldsymbol{S}$	X_i and X_j are conditionally dependent given $\boldsymbol{S}$
$X_i \perp\!\!\!\perp X_j \mid \boldsymbol{S}$	X_i and X_j are conditionally independent given $\boldsymbol{S}$
$\boldsymbol{MB}(X_i)$	Markov blanket of X_i
$\boldsymbol{PC}(X_i)$	a set of parents and children of X_i
$\boldsymbol{SepSet}(X_i, X_j)$	a separation set of X_i from X_j
$P(\cdot\|\cdot)$	conditional probability
$skeleton^*$	the skeleton of a direct acyclic graph

The definition of conditional independence (and dependence) is given as follows.

Definition 3.1 (Conditional Independence)**.** *X_1 and X_2 are conditionally independent given a variable set $\boldsymbol{S}$ if $P(X_1, X_2|\boldsymbol{S}) = P(X_1|\boldsymbol{S})P(X_2|\boldsymbol{S})$; otherwise, X_1 and X_2 are conditionally dependent given $\boldsymbol{S}$.*

Definition 3.2 (Bayesian Network)**.** *[7] Let $\mathbb{P}$ denote the joint probability distribution over a variable set $\mathcal{X}$ of a DAG $\mathcal{G}$. The triplet $<\mathcal{X}, \mathcal{G}, \mathbb{P}>$ is called a Bayesian network iff $<\mathcal{X}, \mathcal{G}, \mathbb{P}>$ satisfies the Markov condition: every variable in $\mathcal{G}$ is independent of any subset of its non-descendants conditioning on its parents.*

Definition 3.3 (D-Separation)**.** *[7] In a DAG $\mathcal{G}$, a path γ is d-separated (or blocked) by a variable set $\boldsymbol{S}\subset\mathcal{X}$ iff 1) γ contains a chain $X_1 \to X_3 \to X_2$ or a fork $X_1 \leftarrow X_3 \to X_2$ with the middle variable $X_3 \in \boldsymbol{S}$, or 2) γ contains an inverted fork (or collider) $X_1 \to X_3 \leftarrow X_2$ with $X_3 \notin \boldsymbol{S}$ and X_3's descendants $\notin \boldsymbol{S}$.*

Two variables X_1 and X_2 are d-separated by a variable set $\boldsymbol{S}$ iff $\boldsymbol{S}$ blocks every path from X_1 to X_2. In this paper, we call such a set $\boldsymbol{S}$ a separation set of X_1 from X_2, denoted as $\boldsymbol{SepSet}(X_1, X_2)$.

Definition 3.4 (Faithfulness)**.** *[35] Given a BN $<\mathcal{X}, \mathcal{G}, \mathbb{P}>$, $\mathbb{P}$ is faithful to $\mathcal{G}$ when for $\forall X_1, X_2 \in \mathcal{X}$, $\exists \boldsymbol{S} \subseteq \mathcal{X} \setminus \{X_1, X_2\}$ d-separates X_1 and X_2 in $\mathcal{G}$ if $X_1 \perp\!\!\!\perp X_2|\boldsymbol{S}$ holds in $\mathbb{P}$.*

Definition 3.4 establishes a relation between a probability distribution $\mathbb{P}$ and its underlying DAG $\mathcal{G}$. In a BN, the faithfulness assumption implies that two variables $X_1, X_2 \in \mathcal{X}$ that are conditionally independent conditioning on a subset $\boldsymbol{S} \subseteq \mathcal{X} \setminus \{X_1, X_2\}$ in $\mathbb{P}$ are d-separated with each other by $\boldsymbol{S}$ in $\mathcal{G}$. Under the assumption, we can use CI tests to find all dependencies or independencies entailed with a BN.

Therefore, by performing CI tests and identifying d-separation relationships among random variables, the entire graph structure can be inferred.

Definition 3.5 (Markov blanket)**.** *[7] Under the faithfulness assumption, the MB of a variable X_1 in $\mathcal{G}$, noted as $\boldsymbol{MB}(X_1)$, is unique and consists of parents, children and spouses of X_1.*

In a BN, sometimes the relations between two variables X_1 and X_2 learned from data are asymmetric (e.g., the PC set of X_1 includes X_2 but X_1 is not in the PC set of X_2). To determine whether they are adjacent, the AND-rule and OR-rule are defined as follows. Clearly the OR-rule is less strict than the AND-rule.

Definition 3.6 (AND-rule). *[31] For a variable X_1 to be adjacent to another variable X_2 in a DAG, both of the following statements hold true: 1) X_1 must be in the PC set of X_2 found by a PC learning algorithm, i.e., $X_1 \in \boldsymbol{PC}(X_2)$ and 2) X_2 must be in the found PC set of X_1, i.e., $X_2 \in \boldsymbol{PC}(X_1)$.*

Definition 3.7 (OR-rule). *[31] Two variables X_1 and X_2 are considered to be adjacent variables if any of the following holds true: 1) X_1 is in the PC set of X_2 found by a PC learning algorithm, i.e., $X_1 \in \boldsymbol{PC}(X_2)$ or 2) X_2 is in the found PC set of X_1, i.e., $X_2 \in \boldsymbol{PC}(X_1)$.*

Proposition 3.1. *[7], [35] In a BN, if there is an edge between variables X_1 and X_2, $\forall \boldsymbol{S} \subseteq \mathcal{X} \setminus \{X_1, X_2\}$, $X_1 \not\perp\!\!\!\perp X_2 | \boldsymbol{S}$ holds.*

Proposition 3.1 states that if X_1 is a parent or a child of X_2, X_1 and X_2 are not conditionally independent conditioning on any variable subsets. Proposition 3.1 is the rationale of learning the PC set of a variable of all existing PC learning algorithms.

4 Our Method

4.1 Overview of the ADL algorithm

In this section, we propose the ADL algorithm to learn a DAG from the dataset $\mathcal{D}$ with the variable set $\mathcal{X}$=$\{X_1, X_2, \cdots, X_m\}$. As shown in Fig. 3, ADL consists of three steps, and our novel contributions lie in Step 2.

- Step 1 learns the PC (Parent-Child) set of each variable in $\mathcal{X}$ using an existing constraint-based PC learning algorithm, such as MMPC [36], HITON-PC [37], and FindPC [38].
- When constructing the global skeleton using the PC sets obtained in Step 1, existing local-to-global DAG learning methods may miss the true edges using the AND-rule, or add the extra edges using the OR-rule. Instead of simply using either rule, in Step 2, we have designed a novel adaptive strategy, and it can simultaneously and adaptively use the AND-rule and OR-rule to construct an accurate global skeleton.
- Step 3 orients edges in the global skeleton using a score-and-search strategy [20], [33].

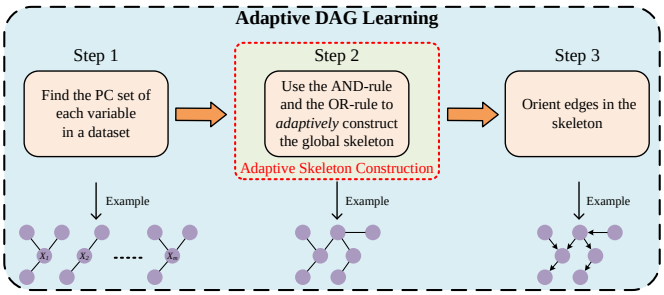


Fig. 3. The framework of the ADL algorithm.

4.2 The adaptive skeleton construction strategy

In this section, we will focus on describing the adaptive skeleton construction strategy of ADL (i.e., Step 2 in Fig. 3). Let $\boldsymbol{PC}(X_i)$ and $\boldsymbol{PC}(X_j)$ represent the learned PC sets of X_i and X_j respectively. Suppose, in Step 1 of ADL, we have learned the PC set of each variable in $\mathcal{X}$=$\{X_1, X_2, \cdots, X_m\}$. For each pair of variables X_i and X_j, according to the learned relations between X_i and X_j, we design the following adaptive strategy to construct a global skeleton, and this strategy consists of two criteria as follows.

Criterion 1: the learned relations between X_i and X_j are symmetrical, i.e., $X_j \in \boldsymbol{PC}(X_i)$ and $X_i \in \boldsymbol{PC}(X_j)$ (*or* $X_j \notin \boldsymbol{PC}(X_i)$ and $X_i \notin \boldsymbol{PC}(X_j)$) hold. Under this situation, either the AND-rule or the OR-rule can be used. For example, if $X_j \in \boldsymbol{PC}(X_i)$ and $X_i \in \boldsymbol{PC}(X_j)$, based on the AND-rule or the OR-rule, X_i and X_j are considered as being adjacent in the true DAG; if $X_j \notin \boldsymbol{PC}(X_i)$ and $X_i \notin \boldsymbol{PC}(X_j)$, based on the AND-rule or the OR-rule, X_i and X_j are determined to be non-adjacent in the true DAG.

The idea behind Criterion 1 is that if both variables consider each other as a direct neighbor, there is a high possibility that these two variables are connected by an edge in the true DAG; similarly, if neither variable considers the other to be its direct neighbor, there is a high possibility that there is no edge between them in the true DAG.

Criterion 2: the learned relations between X_i and X_j are not symmetrical, that is, $X_i \in \boldsymbol{PC}(X_j)$ but $X_j \notin \boldsymbol{PC}(X_i)$ (*or* $X_i \notin \boldsymbol{PC}(X_j)$ but $X_j \in \boldsymbol{PC}(X_i)$). Let $\boldsymbol{SepSet}(X_i, X_j)$ denote a conditioning set that makes X_i and X_j conditionally independent or d-separation. In this case, we discuss this adaptive strategy in the following two aspects.

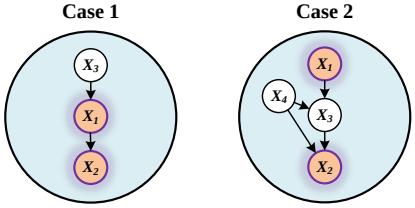


Fig. 4. An example demonstrating how Criteria 2-1 and 2-2 are used.

Criterion 2-1: if $(\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) = \emptyset$, the OR-rule should be used. For instance, in Case 1 of Fig. 4, first, X_1 is added to $\boldsymbol{PC}(X_2)$ and X_3 is added to $\boldsymbol{PC}(X_1)$. Then, due to data issues (e.g., noise or small samples), $\{X_3\}$ may become the set $\boldsymbol{SepSet}(X_1, X_2)$, i.e., $X_1 \perp\!\!\!\perp X_2 | X_3$. Therefore, X_2 cannot be added to $\boldsymbol{PC}(X_1)$, and we obtain $X_2 \notin \boldsymbol{PC}(X_1)$ but $X_1 \in \boldsymbol{PC}(X_2)$. Here, $(\boldsymbol{PC}(X_1) \cap \boldsymbol{PC}(X_2)) \cap \boldsymbol{SepSet}(X_1, X_2) = \emptyset$ holds. Based on Criterion 2-1, we use the OR-rule to determine that X_1 and X_2 are adjacent, which is consistent with the relationship between X_1 and X_2 in the true DAG.

Criterion 2-2: if $(\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) \neq \emptyset$, we should apply the AND-rule. For example, in Case 2 of Fig. 4, when using an existing PC learning algorithm, we may obtain $\boldsymbol{PC}(X_1) = \{X_3, X_2\}$, $\boldsymbol{PC}(X_2) = \{X_3, X_4\}$, and $\boldsymbol{SepSet}(X_1, X_2) = \{X_3, X_4\}$. In this case, $X_2 \in \boldsymbol{PC}(X_1)$ but $X_1 \notin \boldsymbol{PC}(X_2)$, and $(\boldsymbol{PC}(X_1) \cap \boldsymbol{PC}(X_2)) \cap \boldsymbol{SepSet}(X_1, X_2) = \{X_3\} \neq \emptyset$ holds. Based on Criterion 2-2, we employ the AND-rule and determine that X_1 and X_2 are non-adjacent,

which is consistent with the relationship between X_1 and X_2 in the true DAG.

Inspired by the two examples (Case 1 and 2 of Fig. 4) mentioned above, we propose the following Theorem 4.1 for introducing the idea of Criterion 2.

Theorem 4.1. *Suppose all paths between X_i and X_j in $\mathcal{G}$ contain at least one of the following cases: 1) $X_i - X_j$ (e.g. $X_i \leftarrow X_j, X_i \rightarrow X_j$). 2) a chain $X_i \rightarrow \square \rightarrow X_j$ (or $X_i \leftarrow \square \leftarrow X_j$). 3) a fork $X_i \leftarrow \square \rightarrow X_j$. Given a PC learning algorithm for learning $\boldsymbol{PC}(X_i)$ and $\boldsymbol{PC}(X_j))$ from $\mathcal{D}$ generated by $\mathcal{G}$, if $X_i \in \boldsymbol{PC}(X_j)$ but $X_j \notin \boldsymbol{PC}(X_i)$ (or $X_i \notin \boldsymbol{PC}(X_j)$ but $X_j \in \boldsymbol{PC}(X_i)$), the relationship between X_i and X_j in $\mathcal{G}$ is determined as adjacent when $(\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) = \emptyset$ holds.*

Proof. We assume that X_i is not adjacent to X_j, that is, all paths between X_i and X_j contain at least a chain $X_i \rightarrow \square \rightarrow X_j$ (or $X_i \leftarrow \square \leftarrow X_j$) or a fork $X_i \leftarrow \square \rightarrow X_j$, and we let $X_k \in \mathcal{X} \setminus \{X_i, X_j\}$ be the middle variable. Thus, $X_k \in \boldsymbol{PC}(X_i)$ and $X_k \in \boldsymbol{PC}(X_j)$, further, $X_k \in \boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)$. Since $X_i \in \boldsymbol{PC}(X_j)$ and $X_j \notin \boldsymbol{PC}(X_i)$, the PC set of X_i learned by a PC learning algorithm is true, whereas the learned PC set of X_j is false. Clearly, there is a set $\boldsymbol{S} \subseteq \boldsymbol{PC}(X_i)$ that makes $X_i \perp\!\!\!\perp X_j | \boldsymbol{S}$ hold due to $X_j \notin \boldsymbol{PC}(X_i)$. According to Definition 3.4, $\boldsymbol{S}$ is the separation set $\boldsymbol{SepSet}(X_i, X_j)$ that can d-separate X_i and X_j in $\mathcal{G}$. Based on Definition 3.3, $\boldsymbol{SepSet}(X_i, X_j)$ must include the middle variable X_k, i.e., $X_k \in \boldsymbol{SepSet}(X_i, X_j)$. Thus, $X_k \in (\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j)$, and $(\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) \neq \emptyset$, which is against the condition $((\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) = \emptyset)$. Thus, the *assumption* that X_i is not adjacent to X_j does not hold, and X_i is adjacent to X_j.
Summarizing: Theorem 4.1 is true. (Q.E.D) □

Thus, according to Theorem 4.1, the ideas behind Criterion 2-1 and Criterion 2-2 are as follows.

The idea of Criterion 2-1. Given a PC learning algorithm for learning the PC sets of X_i and X_j, and for the learned PC sets, we have $X_i \in \boldsymbol{PC}(X_j)$ but $X_j \notin \boldsymbol{PC}(X_i)$. In this case, if the length of the shortest path between X_i and X_j in the true DAG is equal to or greater than 2, it would not be possible to have a direct edge between X_i and X_j. Otherwise, if there is an edge between X_i and X_j, the path between X_i and X_j should satisfy the *assumption* in Theorem 4.1. Thus, when Theorem 4.1 holds, the OR-rule is used to determine the relationship between X_i and X_j.

The idea of Criterion 2-2. When $X_i \in \boldsymbol{PC}(X_j)$ but $X_j \notin \boldsymbol{PC}(X_i)$, if using the AND-rule, we will assume that X_i and X_j are non-adjacent in the true DAG. With this assumption, there must exist a conditioning set $\boldsymbol{SepSet}(X_i, X_j) \subseteq \boldsymbol{PC}(X_i)$ (or $\boldsymbol{PC}(X_j)$) that d-separates X_i and X_j, i.e., $X_i \perp\!\!\!\perp X_j | \boldsymbol{SepSet}(X_i, X_j)$. Thus when $X_j \notin \boldsymbol{PC}(X_i)$ holds, during the learning of $\boldsymbol{PC}(X_i)$, a conditioning set $\boldsymbol{SepSet}(X_i, X_j) \subseteq \boldsymbol{PC}(X_i)$ is found and it makes X_i and X_j independent. By the symmetry rule of conditional independence tests, when learning the PC set of X_j, if this found set $\boldsymbol{SepSet}(X_i, X_j) \subseteq \boldsymbol{PC}(X_j)$, X_j and X_i are also conditional independent. When X_i and X_j are non-adjacent in the true DAG, but an algorithm finds $X_i \in \boldsymbol{PC}(X_j)$, the explanation is that the set $\boldsymbol{PC}(X_j)$ found by the algorithm only includes a subset of variables within $\boldsymbol{SepSet}(X_i, X_j)$ (i.e., $\exists \boldsymbol{S} \subseteq \boldsymbol{PC}(X_j)$ and $\boldsymbol{S} \subset \boldsymbol{SepSet}(X_i, X_j)$) so that this algorithm cannot find the set $\boldsymbol{SepSet}(X_i, X_j)$ within $\boldsymbol{PC}(X_j)$ to make X_i and X_j independent. Accordingly, under the situation that X_i and X_j are non-adjacent in the true DAG, it is more likely that $(\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) \neq \emptyset$. In other words, under the situation of $X_i \in \boldsymbol{PC}(X_j)$ but $X_j \notin \boldsymbol{PC}(X_i)$, if $(\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) \neq \emptyset$, the AND-rule should be used to determine the relationship between X_i and X_j, and Criterion 2-2 is reasonable.

As for Criterion 2, if the relation between X_i and X_j is represented as $X_i \rightarrow X_k \leftarrow X_j$, $\boldsymbol{SepSet}(X_i, X_j)$ may be an empty set, leading to $(\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) = \emptyset$. In this case, both Criterion 2-1 and Criterion 2-2 cannot be correctly used and it would wrongly determine that there is an edge between X_i and X_j.

4.3 Detailed descriptions of the ADL algorithm

Algorithm 1 gives more details of ADL, and we describe the details in the followings.

Step 1 (Line 1 of Algorithm 1). In this step, any up-to-date PC (parents and children) learning algorithm can be used to learn the PC set for each variable. In our implementation, we employ HITON-PC [37], one of the best PC learning algorithms for this step. Learning a local skeleton for each variable makes ADL scalable to high-dimensional data. Based on Proposition 3.1, when learning the PC set of a target variable (such as X_1), the condition set that makes X_1 and non-PC variable of X_1 (such as X_2) independent can be obtained. Although there are many such condition sets, we only record the condition set with the smallest size, i.e., $\boldsymbol{SepSet}(X_1, X_2)$ (See Definition 3.3 for details). At the end of the step, we obtain the PC sets of all variables, $\boldsymbol{PC}$s= $\{\boldsymbol{PC}(X_1), \boldsymbol{PC}(X_2), \cdots, \boldsymbol{PC}(X_m)\}$.

Algorithm 1 Adaptive DAG Learning (ADL)

Input: $\mathcal{D}$: dataset with the variable set $\mathcal{X}$=$\{X_1, X_2, \cdots, X_m\}$
Output: $\mathcal{G}^*$: the highest scoring DAG
{*Step 1: learn the PC sets of all variables*}
1: $\boldsymbol{PC}$s=$\{\boldsymbol{PC}(X_1), \boldsymbol{PC}(X_2), \cdots, \boldsymbol{PC}(X_m)\}$
{*Step 2: adaptively use the AND-rule and the OR-rule to construct the DAG skeleton*}
2: **for** $i = 1, \ldots, m$; $j = 1, \ldots, (i-1)$ **do**
3: **if** $X_j \in \boldsymbol{PC}(X_i)$ and $X_i \in \boldsymbol{PC}(X_j)$ **then**
4: $skeleton^*(i,j)$=$skeleton^*(j,i)$=1 {*Criterion 1*}
5: **else if** $X_j \notin \boldsymbol{PC}(X_i)$ and $X_i \notin \boldsymbol{PC}(X_j)$ **then**
6: $skeleton^*(i,j)$=$skeleton^*(j,i)$=0 {*Criterion 1*}
7: **else** {*Criterion 2*}
8: **if** $(\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) = \emptyset$ **then**
9: $skeleton^*(i,j)$=$skeleton^*(j,i)$=1 {*Criterion 2-1*}
10: **else**
11: $skeleton^*(i,j)$=$skeleton^*(j,i)$=0 {*Criterion 2-2*}
12: **end if**
13: **end if**
14: **end for**
{*Step 3: orient edges in the skeleton*}
15: $\mathcal{G}^* \xleftarrow{greedy\ search\ and\ scoring} skeleton^*$
16: **return** $\mathcal{G}^*$

Step 2 (Lines 2-14 of Algorithm 1). Using the learned PC sets (i.e. $\boldsymbol{PC}$s), ADL *adaptively* constructs the DAG skeleton by the AND-rule and the OR-rule. Here, the

skeleton of a DAG is denoted as $skeleton^*(i,j)$ (an adjacency matrix where $i,j \in \{1,\ldots,m\}$). Specifically, if $skeleton^*(i,j)=0$, there is no edge between X_i and X_j, while if $skeleton^*(i,j)=1$, X_i and X_j are considered to be adjacent in the skeleton.

For each pair of variables X_i and X_j, at Lines 3 and 5, ADL firstly checks whether the learned relations between X_i and X_j are symmetrical. If so, ADL will apply Criterion 1 to adaptively determine the relationship between X_i and X_j by using the AND-rule or the OR-rule at Line 3-6.

When the learned relations between X_i and X_j are not symmetrical, at Lines 8 to 11, ADL uses Criterion 2 to adaptively determine whether there is an edge between X_i and X_j. Specifically, if $(\boldsymbol{PC}(X_i) \cap \boldsymbol{PC}(X_j)) \cap \boldsymbol{SepSet}(X_i, X_j) = \emptyset$ holds, ADL uses the OR-rule at Line 9. Otherwise, ADL uses the AND-rule at Line 11. By adaptively employing the AND-rule and the OR-rule to determine the relationship between two variables, the ADL algorithm is able to tackle the shortcomings of existing local-to-global DAG learning algorithms.

Step 3 (Line 15 of Algorithm 1). To orient edges in the DAG skeleton achieved at Step 2, at Line 15, ADL uses a Bayesian score criterion, BDeu [20], and a search procedure, hill-climbing [15], [33].

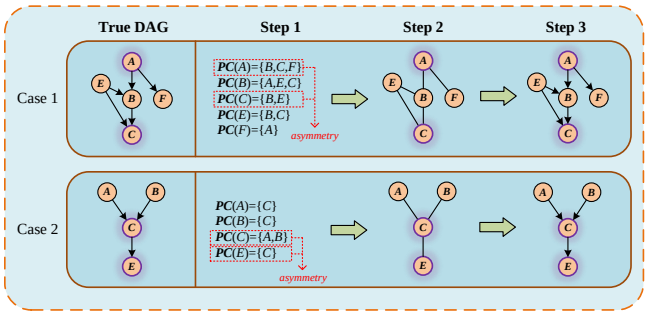


Fig. 5. Two examples of tracing the ADL algorithm.

As shown in Fig. 5, we use two examples to demonstrate how ADL works.

- Case 1. In Step 1 (Line 1 of Algorithm 1), ADL obtains the PC set of each variable in $\mathcal{X}=\{A,B,C,E,F\}$ using the HITON-PC algorithm. However, we find that there is an asymmetrical relationship, i.e., $C \in \boldsymbol{PC}(A)$ but $A \notin \boldsymbol{PC}(C)$. This is because, based on Definition 3.3, $\{B,F\}$ cannot d-separate the path between A and C. Thus, according to Definition 3.4, $A \not\perp\!\!\!\perp C|\{B,F\}$ holds, and C is added to the PC set of A as a false positive. In this case, $\boldsymbol{SepSet}\{A,C\} = \{B,E\}$ since $A \perp\!\!\!\perp C|\{B,E\}$ holds according to Definitions 3.3 and 3.4. Therefore, $(\boldsymbol{PC}(A) \cap \boldsymbol{PC}(C)) \cap \boldsymbol{SepSet}\{A,C\} = \{B\} \neq \emptyset$, and based on Criterion 2-2, A and C are considered to be non-adjacent (Lines 10-11 of Algorithm 1). Then, based on Criterion 1, ADL constructs the complete global skeleton in Step 2. Finally, in Step 3 (Line 15 of Algorithm 1), by orienting the undirected edges in the global skeleton, we get the true DAG.
- Case 2. ADL first finds the PC sets of all variables in Step 1 (Line 1 of Algorithm 1), but $C \in \boldsymbol{PC}(E)$ and $E \notin \boldsymbol{PC}(C)$. Clearly, according to Proposition 3.1, learning the PC set of C encounters an error (i.e. $C \perp\!\!\!\perp E|A$) due to data issues (e.g. noise or small samples). In this case, $\boldsymbol{SepSet}\{C,E\} = \{A\}$. Due to $(\boldsymbol{PC}(C) \cap \boldsymbol{PC}(E)) \cap \boldsymbol{SepSet}\{C,E\} = \emptyset$, based on Criterion 2-1, C and E are determined as adjacent (Lines 8-9 of Algorithm 1). Then according to Criterion 1, the global skeleton is constructed correctly in Step 2. In Step 3 (Line 15 of Algorithm 1), based on the true global skeleton, ADL learns the true DAG by a score-and-search strategy.

5 Experiments

In this section, we conduct experiments to systematically evaluate the effectiveness of our method. In Section 5.1, we describe the experiment settings, including comparison methods, datasets, evaluation metrics, and implementation details. Section 5.2 and Section 5.3 compare our proposed algorithm with 12 other DAG learning algorithms on 11 benchmark BN datasets and a real dataset, respectively.

5.1 Experiment setting

5.1.1 Comparison methods.

We compare the ADL algorithm with five representative local-to-global DAG learning algorithms, including GSBN[1] [34], MMHC[2] [15], SLL+C/G[3] [16] and GGSL [14], and seven well-established and state-of-the-art global DAG learning algorithms, including K2[4] [22], OBS [19], Improved-K2 [24], NOTEARS[5] [12], DAG-GNN[6] [13], GOLEM[7] [28] and DAG-NoCurl[8] [29].

5.1.2 Datasets.

TABLE 2
Summary of benchmark BNs

Network	Num. Vars	Num. Edges	Max In/out-Degree	Min/Max |PCset|	Variable Domain
Child	20	25	2/7	1/8	2-6
Alarm	37	46	4/5	1/6	2-4
Child3	60	79	3/7	1/8	2-6
Alarm3	111	149	4/5	1/6	2-4
Insurance5	135	284	5/8	1/10	2-5
Alarm5	185	265	4/6	1/8	2-4
Insurance10	270	556	5/8	1/11	2-5
Alarm10	370	570	4/7	1/9	2-4
Pigs	441	592	2/39	1/41	3-3
Gene	801	972	4/10	0/11	3-5
Munin	1041	1397	3/69	1/69	1-15

1. The source codes of GSBN are available at https://github.com/kuiy/CausalLearner.
2. The source codes of MMHC are available at http://mensxmachina.org/en/software/probabilistic-graphical-model-toolbox.
3. The source codes of SLL+C/G are available at https://www.cs.helsinki.fi/u/tzniinim/uai2012.
4. The source codes of K2 are available at https://github.com/bayesnet/bnt.
5. The implementation is publicly available at https://github.com/xunzheng/notears.
6. The code is available at https://github.com/fishmoon1234/DAG-GNN.
7. The code is available at https://github.com/ignavierng/golem.
8. The code is available at https://github.com/fishmoon1234/DAG-NoCurl.

Several benchmark BN datasets. We first evaluate our algorithm and its rivals on eleven benchmark BNs, using the datasets provided from existing works [15]. Each BN consists of two groups data containing 5 datasets with 500, and 1000 data instances, respectively. The details of the eleven benchmark BNs are presented in Table 2[9].

A real dataset. We also compare the proposed method with its rivals on a real biological dataset, Sachs [39]. Sachs is a protein signaling network expressing the level of different proteins and phospholipids in human cells. It is commonly viewed as a benchmark graphical model with 11 nodes (cell types) and 17 edges. In our experiments, we utilize the observational data with 7466 samples.

For the experiments on the synthetic datasets, please see the Supplementary Material.

5.1.3 Evaluation metrics.

We evaluate the performance of ADL and its rivals from three aspects: structure errors, structure correctness and efficiency. The *SHD* (Structural Hamming Distance) and *Ar_F1* metrics as shown below are used to measure structure error and structure correctness, respectively. The running time is utilized as the efficiency measure of the algorithms.

- *SHD* (Structural Hamming Distance) is the sum of the values of *Miss*, *Extra*, *Reverse* and *Undirected*, where *Miss* is the number of missing edges in the DAG learned by an algorithm against the true DAG, *Extra* is the number of extra edges in the learned DAG, *Reverse* is the number of edges with wrong directions according to the true DAG, and *Undirected* is the number of undirected edges in the learned DAG. In our experiments, we randomly orient the edges that cannot be oriented by a DAG learning algorithm, thus the value of *Undirected* is always 0 and Undirected metric is not shown in the experimental results. The smaller the value of SHD the better.
- Ar_F1=$\frac{2*Ar\_Precision*Ar\_Recall}{Ar\_Precision+Ar\_Recall}$. The $Ar\_Precision$ denotes the number of correctly predicted arrowheads in the output divided by the number of edges in the output of an algorithm, while the Ar_Recall denotes the number of correctly predicted arrowheads in the output divided by the number of true arrowheads in a test DAG. Compared to SHD, Ar_F1 not only considers erroneous edges, but also correct edges. A larger value of Ar_F1 is better.

On benchmark BN datasets, for each algorithm, we report the average results of these metrics over 5 datasets generated by a BN. In Tables 3-8, the symbol "-" denotes that an algorithm does not produce results on the corresponding BN when the running time of the algorithm exceeded 12 hours or there is no enough memory space, and the results are shown in the format of $A \pm B$, where A represents the average of these metrics, and B is the standard deviation. In addition, the best results are highlighted in bold face.

5.1.4 Implementation details.

All experiments were conducted on a computer with Inter Core i9-10900 3.70-GHz CPU, NVIDIA GeForce RTX 3060 GPU and 64-GB memory. The significance level for conditional independence tests is set to 0.01. For continuous optimization based DAG learning methods (i.e., NOTEARS, DAG-GNN, GOLEM and DAG-NoCurl), we adopt 0.3 as the threshold to prune the DAGs obtained from those methods [28]. K2, OBS, Improved-K2, GSBN, MMHC and our algorithm are implemented in MATLAB, SLL+C/G and GGSL are implemented in C++, and NOTEARS, DAG-GNN, GOLEM and DAG-NoCurl are implemented in PYTHON.

9. Those benchmark BNs are publicly available at http://www.bnlearn.com/bnrepository/.

5.2 Results of DAG learning on benchmark data

In this section, we report the experimental results of ADL and its rivals on 11 benchmark BN datasets. Specifically, Section 5.2.1, 5.2.2 and 5.2.3 describe the structure errors, structure correctness and time efficiency, respectively. In Section 5.2.4, we compare BIC (Bayesian information criterion) scores [40] of each algorithm. Finally, we conduct statistical tests for verifying whether ADL is significantly better than other methods in Section 5.2.5.

5.2.1 Structure errors.

From Tables 3 and 4, we can see that for almost all benchmark datasets with both 500 and 1,000 samples, the ADL algorithm achieves a lower SHD value than the other algorithms, which indicates the superiority of our method. The reason is as follows: the adaptive skeleton learning strategy adopted by ADL enables that ADL can reduce the number of missing edges and extra edges simultaneously, further reducing the number of incorrect directed edges.

In the following, we compare ADL to each of its rivals from four metrics, SHD, Reverse, Miss and Extra.

(1) ADL against K2, OBS and Improved-K2. It can be seen that the quality of the structures learned by K2 and OBS is generally worse than that learned by local-to-global DAG learning methods (especially ADL), and is competitive with that learned by continuous optimization based DAG learning methods. Specifically, compared with K2, ADL reduces the value of SHD by approximately 87% and 92% on Child with 1,000 samples and Alarm with 1,000 samples, respectively. Compared with OBS, ADL reduces the value of SHD by approximately 91% on Alarm with 1,000 samples. As ordering-based DAG learning methods, K2 and OBS does not achieve a higher quality of learned structures than ADL, probably since their performance depend on the order of nodes given initially. Although OBS constantly updates the order of nodes in the space of node-ordering and finds the highest scoring DAG consistent with each ordering, it only obtains a local optimal solution. Further, as a novel ordering-based method, Improved-K2 aims to derive a high-quality ordering of variables from a given dataset as an initial input. As a result, Improved-K2 significantly reduces structural errors compared with K2, and is competitive with local-to-global DAG learning methods.

(2) ADL against GSBN. We can see that for almost all benchmark datasets with both 500 and 1,000 samples, ADL is significantly superior to GSBN, since the SHD value of ADL is much smaller than that of GSBN. Specifically, compared with GSBN, ADL reduces the value of SHD by approximately 97% on Pigs with 1,000 samples, and reduces the values of Reverse and Miss by approximately 90% and

TABLE 3
Summaries of wrongly learned edges (1)

	#Sample	500				1000			
Network	Algorithm	SHD (↓)	Reverse (↓)	Miss (↓)	Extra (↓)	SHD (↓)	Reverse (↓)	Miss (↓)	Extra (↓)
Child	K2	28.00±5.06	9.60±3.59	10.20±2.98	8.20±1.45	25.60±2.92	6.00±2.04	13.20±5.23	6.40±0.95
	OBS	26.20±5.06	6.00±0.71	12.80±3.53	7.40±1.37	21.00±4.13	7.20±2.82	8.60±2.52	5.20±1.86
	Improved-K2	13.00±3.25	4.40±1.20	8.20±1.30	0.40±0.55	6.60±0.22	1.40±1.52	4.40±1.34	0.80±0.20
	GSBN	14.40±1.14	**1.60±0.89**	12.80±0.45	**0.00±0.00**	11.40±0.55	1.60±0.55	9.80±0.45	**0.00±0.00**
	MMHC	9.20±3.70	3.80±1.64	4.40±1.34	1.00±1.00	3.40±0.89	**1.00±0.00**	**2.00±1.00**	0.40±0.89
	SLL+C	8.00±3.16	2.40±1.82	4.80±1.10	0.80±0.84	5.20±1.30	1.20±0.84	4.00±0.71	**0.00±0.00**
	SLL+G	10.20±2.39	4.00±1.58	5.80±1.30	0.40±0.55	7.80±1.64	3.60±1.14	4.20±0.84	**0.00±0.00**
	GGSL	11.00±2.45	4.80±2.28	5.60±1.14	0.60±0.55	9.40±1.52	4.80±2.59	4.40±1.67	0.20±0.45
	NOTEARS	22.20±2.69	6.80±2.37	15.00±4.55	0.40±0.06	22.80±3.25	5.40±2.05	16.60±5.25	0.80±0.20
	DAG-GNN	21.80±3.14	8.20±1.24	12.00±1.73	1.60±0.39	22.40±2.72	6.60±2.62	14.80±3.27	1.00±0.30
	GOLEM	39.20±3.02	7.80±2.66	13.40±3.15	18.00±3.87	39.20±3.30	4.80±0.53	15.80±4.51	18.60±5.02
	DAG-NoCurl	41.20±6.70	10.60±3.83	11.20±2.22	19.40±5.72	41.00±5.66	8.80±1.57	13.60±5.28	18.60±4.87
	ADL	**5.80±2.59**	2.40±2.30	**3.20±0.84**	0.20±0.45	**3.20±0.84**	**1.00±0.00**	**2.00±0.71**	0.20±0.45
Alarm	K2	51.20±6.34	14.80±2.85	19.00±3.40	17.40±5.70	54.00±4.19	15.00±5.51	17.60±3.50	21.40±3.61
	OBS	46.40±4.43	14.80±5.89	16.00±5.45	15.60±3.55	45.40±7.03	17.00±2.04	12.20±2.23	16.20±2.99
	Improved-K2	16.60±6.06	1.80±0.57	12.00±0.47	2.80±0.29	9.60±1.53	0.80±0.17	7.80±1.30	1.00±0.26
	GSBN	35.20±0.84	11.00±1.00	23.00±1.22	1.20±0.45	29.60±2.07	12.20±1.30	17.40±1.34	**0.00±0.00**
	MMHC	9.40±3.85	2.00±2.92	**4.60±0.55**	2.80±0.84	7.00±2.92	2.40±2.61	**2.40±0.89**	2.20±0.84
	SLL+C	14.40±1.67	7.00±2.00	5.00±0.71	2.40±0.89	12.20±2.59	8.00±2.55	**2.40±0.55**	1.80±1.48
	SLL+G	15.20±3.70	8.20±5.07	5.00±0.71	2.00±1.00	11.40±3.36	7.40±4.22	**2.40±0.55**	1.60±1.14
	GGSL	18.20±2.28	6.80±3.11	6.00±3.94	5.40±1.82	13.40±2.07	4.60±2.07	6.00±1.73	2.80±1.92
	NOTEARS	29.80±3.47	4.60±0.56	20.40±5.66	4.80±1.79	26.80±3.98	4.80±1.61	19.40±2.60	2.60±1.02
	DAG-GNN	32.00±2.40	5.00±1.92	16.40±5.40	10.60±2.84	30.20±3.39	6.80±1.97	12.00±2.03	11.40±3.12
	GOLEM	64.80±12.91	9.00±3.21	17.40±6.76	38.40±11.66	47.40±3.54	3.60±0.91	19.40±4.31	24.40±3.23
	DAG-NoCurl	99.60±9.97	12.40±2.71	14.80±4.61	72.40±19.37	78.20±13.66	11.40±3.54	13.40±4.85	53.40±6.18
	ADL	**7.20±3.56**	**1.20±2.17**	5.20±1.30	**0.80±0.45**	**4.00±2.35**	**0.60±1.34**	2.80±0.45	0.60±0.89
Child3	K2	77.20±10.77	22.60±8.80	37.80±5.09	16.80±1.94	70.40±12.16	22.20±5.23	30.40±6.08	17.80±2.10
	OBS	66.00±12.35	18.20±4.77	33.40±7.67	14.40±3.78	65.00±8.16	24.60±3.70	24.40±7.94	16.00±3.37
	Improved-K2	37.80±10.90	8.20±0.35	28.60±7.48	1.00±0.20	32.20±0.20	11.60±4.57	18.20±1.22	2.40±0.10
	GSBN	58.00±1.22	12.60±0.55	45.40±1.14	**0.00±0.00**	50.80±1.10	15.20±0.84	35.20±0.84	**0.40±0.55**
	MMHC	33.80±6.06	10.00±3.39	21.40±1.67	2.40±1.52	27.00±3.08	9.20±1.10	16.40±2.07	1.40±1.14
	SLL+C	32.80±4.66	8.80±3.42	**19.40±1.52**	4.60±2.30	28.00±4.42	8.00±3.08	17.60±2.70	2.40±1.14
	SLL+G	32.80±5.31	9.00±2.83	20.60±2.30	3.20±2.17	26.00±5.39	7.00±2.24	17.60±2.70	1.40±1.52
	GGSL	41.60±14.19	7.40±3.36	29.80±19.23	4.40±2.97	31.60±16.91	6.60±4.04	22.40±20.66	2.60±2.07
	NOTEARS	70.40±6.23	**6.60±1.61**	61.00±10.80	2.80±0.59	67.60±10.24	**5.80±1.52**	61.00±18.00	0.80±0.25
	DAG-GNN	68.80±6.59	21.60±3.42	40.00±7.92	7.20±2.62	60.80±6.58	16.00±2.46	43.80±16.55	1.00±0.12
	GOLEM	121.80±23.87	11.40±2.53	57.40±20.28	53.00±15.08	138.60±25.28	17.40±2.03	48.60±16.77	72.60±18.78
	DAG-NoCurl	144.60±18.86	35.20±12.12	35.80±13.23	73.60±16.84	112.20±18.10	43.60±16.87	28.20±6.58	40.40±5.08
	ADL	**30.80±3.70**	9.00±2.83	19.60±2.70	2.20±0.84	**24.40±4.34**	8.20±1.92	**15.80±2.95**	**0.40±0.55**
Alarm3	K2	150.40±24.25	38.20±8.15	69.40±22.24	42.80±16.53	142.20±22.73	37.00±7.53	64.60±24.81	40.60±10.40
	OBS	146.00±11.57	35.40±5.47	72.40±9.26	38.20±9.13	138.00±23.56	45.80±17.14	54.00±18.08	38.20±7.21
	Improved-K2	93.00±33.14	18.60±2.49	61.60±17.22	12.80±1.01	65.60±8.91	14.20±5.41	47.00±17.30	4.40±1.34
	GSBN	115.00±2.00	41.00±2.45	73.20±1.48	**0.80±0.84**	94.40±4.16	35.80±3.63	58.40±2.41	**0.20±0.45**
	MMHC	56.00±8.43	12.20±4.66	**27.40±1.95**	16.40±4.83	47.40±9.32	11.00±5.52	23.20±1.30	13.20±3.96
	SLL+C	91.40±10.90	25.80±3.56	28.80±2.59	36.80±6.50	76.80±8.87	23.60±3.91	**22.00±2.35**	31.20±5.93
	SLL+G	88.00±8.72	31.60±8.23	31.00±1.58	25.40±4.72	77.60±11.46	35.00±10.70	23.00±1.58	19.60±5.03
	GGSL	122.40±6.69	21.20±6.53	61.80±13.52	39.40±8.85	97.80±5.89	15.20±4.97	49.40±5.90	33.20±7.05
	NOTEARS	111.40±7.82	8.00±0.87	95.80±15.57	7.60±1.80	109.40±11.51	7.00±1.18	96.80±19.00	5.60±1.95
	DAG-GNN	114.60±13.30	13.00±4.34	90.00±20.79	11.60±2.68	113.60±12.18	12.80±3.96	93.40±31.65	7.40±1.72
	GOLEM	231.00±34.01	7.00±1.21	115.00±40.53	109.00±38.93	234.00±36.92	13.40±4.97	99.40±37.03	121.20±39.29
	DAG-NoCurl	390.40±38.81	29.80±7.78	59.40±19.52	301.20±61.53	231.80±19.95	29.20±4.59	59.60±8.57	143.00±39.40
	ADL	**41.60±8.71**	**4.80±4.15**	30.20±2.05	6.60±4.22	**38.40±5.94**	**6.80±4.55**	26.20±1.79	5.40±1.14
Insurance5	K2	262.80±23.92	52.40±16.49	167.60±45.81	42.80±9.84	275.40±52.43	59.00±14.62	150.80±44.71	65.60±24.08
	OBS	235.00±35.80	40.80±11.72	157.20±31.19	37.00±12.61	239.40±17.89	46.40±9.18	145.20±56.90	47.80±10.01
	Improved-K2	247.00±27.42	37.00±9.39	193.60±17.23	16.40±2.05	243.20±13.30	61.60±8.23	149.20±14.93	32.40±6.74
	GSBN	234.00±4.24	25.80±2.68	194.60±2.97	13.60±1.14	197.80±1.30	25.80±2.17	169.00±1.41	**3.00±0.71**
	MMHC	168.60±8.76	27.80±6.53	118.60±3.58	22.20±4.27	163.80±4.15	44.20±5.50	98.80±1.64	20.80±4.32
	SLL+C	204.40±3.65	31.20±4.49	131.20±2.49	42.00±2.45	164.80±12.11	26.00±7.91	110.00±2.00	28.80±4.44
	SLL+G	187.20±8.53	30.80±6.22	134.20±2.39	22.20±2.05	159.80±7.36	33.60±7.02	113.00±2.83	13.20±2.77
	GGSL	176.00±8.63	**23.20±2.59**	119.00±13.11	33.80±10.57	148.20±12.66	**18.80±3.27**	109.60±17.16	19.80±3.83
	NOTEARS	258.60±25.91	49.80±5.33	199.80±56.39	**9.00±1.20**	259.00±22.69	46.20±9.22	202.00±74.59	10.80±2.70
	DAG-GNN	268.60±31.69	37.40±6.08	221.80±51.52	9.40±3.64	268.60±19.83	43.60±11.26	213.40±23.39	11.60±4.04
	GOLEM	325.20±24.60	40.00±12.30	203.40±80.08	81.80±15.13	325.20±47.35	30.20±11.52	235.80±26.96	59.20±6.88
	DAG-NoCurl	728.40±108.83	81.00±29.97	153.60±24.27	493.80±161.14	613.00±52.00	90.00±30.50	131.60±37.55	391.40±47.38
	ADL	**154.80±6.10**	25.60±4.34	**118.00±2.45**	11.20±1.92	**139.40±15.14**	36.80±10.38	**97.40±3.21**	5.20±2.59
Alarm5	K2	263.60±46.52	67.20±19.32	130.40±41.93	66.00±22.54	267.60±21.95	77.00±13.19	117.00±20.25	73.60±9.67
	OBS	268.20±45.10	67.00±21.62	137.80±48.14	63.40±9.32	256.00±39.57	71.80±18.30	110.60±21.89	73.60±21.97
	Improved-K2	182.00±13.32	29.80±2.86	125.80±24.61	26.40±0.30	133.40±8.77	28.20±2.87	92.60±4.69	12.60±0.90
	GSBN	209.00±4.47	72.20±3.27	135.60±0.89	**1.20±0.84**	182.00±5.10	66.00±4.85	115.40±3.97	**0.60±0.55**
	MMHC	125.20±6.76	24.40±8.17	62.20±1.30	38.60±3.36	96.00±8.28	15.80±8.17	52.80±1.10	27.40±4.04
	SLL+C	208.60±18.28	46.00±5.29	**61.60±0.89**	101.00±15.98	180.80±8.35	42.20±3.70	**52.00±2.55**	86.60±8.08
	SLL+G	173.60±4.28	58.40±10.06	64.60±1.52	50.60±8.11	154.00±14.66	52.60±14.01	54.80±2.77	46.60±3.58
	GGSL	226.80±13.75	24.60±3.85	152.60±14.50	49.60±14.05	202.20±11.01	24.80±3.96	118.80±18.94	58.60±11.41
	NOTEARS	195.80±34.96	13.40±4.94	174.60±47.97	7.80±2.14	194.00±37.90	13.00±5.08	173.00±23.73	8.00±1.92
	DAG-GNN	208.60±25.66	21.80±2.42	167.00±54.08	19.80±2.90	198.40±28.28	22.60±5.16	163.20±29.40	12.60±4.11
	GOLEM	389.80±43.71	**11.00±2.43**	212.60±47.18	166.20±35.88	379.80±72.91	**9.80±1.73**	225.60±58.64	144.40±55.80
	DAG-NoCurl	993.60±81.78	48.60±9.57	100.00±33.09	845.00±143.85	457.20±36.08	44.20±8.39	112.20±31.13	300.80±48.43
	ADL	**96.00±6.28**	11.40±6.66	69.40±2.41	15.20±4.82	**80.40±6.88**	**9.80±5.17**	60.20±1.48	10.40±2.19

100% respectively on Pigs with both 500 and 1,000 samples. On most datasets, the local skeleton of each variable learned by GSBN has lost many true edges, furthermore, GSBN employs the AND-rule to construct the global skeleton. As a result, GSBN avoids learning some extra edge and achieves a very low the value of Extra, but many true edges are missing from the learned global skeleton, further leading to many incorrectly oriented edges in the DAG.

(3) ADL against MMHC. Using both groups of datasets (with 500 and 1,000 samples respectively), we have the following observations. The quality of DAG learned by ADL outperforms those learned by MMHC on all benchmark datasets, since the value of SHD of ADL is smaller than those of MMHC. In some large-sized BNs (e.g. Insurance5, Insurance10, Pigs and Gene), the SHD value of MMHC is competitive with that of ADL, since they all employ a score-based strategy to orient edges in the global skeleton. Specifically, compared with MMHC, ADL reduces the value of SHD by approximately 43% on Alarm with 1,000 samples, the value of Reverse by approximately 68% on Alarm10 with 500 samples, and the value of Extra by approximately 80% on Child with 500 samples. The reason why the missing edge errors of ADL is higher than that of MMHC on some datasets (such as Alarm10 with 500 samples) is that if a variable has many parents, the size of the learned separation set may be larger than the true separation set in the Step 1 of ADL. According to Lines 8-12 of Algorithm 1, ADL will prefer to execute Criterion 2-2 instead of Criterion 2-

TABLE 4
Summaries of wrongly learned edges (2)

	#Sample	500				1000			
Network	Algorithm	SHD (↓)	Reverse (↓)	Miss (↓)	Extra (↓)	SHD (↓)	Reverse (↓)	Miss (↓)	Extra (↓)
Insurance10	K2	509.00±90.42	89.60±16.85	330.60±63.74	88.80±22.81	516.60±79.87	109.60±37.13	290.60±49.41	116.40±32.62
	OBS	-	-	-	-	-	-	-	-
	Improved-K2	491.20±40.03	74.40±14.89	365.00±38.78	51.80±1.24	400.80±5.23	71.00±5.94	278.80±8.35	51.00±3.65
	GSBN	455.60±6.02	54.00±3.74	374.20±1.79	27.40±1.14	385.80±8.70	55.40±4.45	323.60±3.85	**6.80±2.17**
	MMHC	361.60±6.11	72.00±5.24	235.80±4.32	53.80±4.21	310.20±12.52	68.00±9.03	190.20±5.63	52.00±3.74
	SLL+C	430.60±14.67	65.00±9.97	251.80±2.95	113.80±9.76	-	-	-	-
	SLL+G	376.00±12.71	69.60±7.13	259.80±3.83	46.60±5.94	-	-	-	-
	GGSL	385.80±13.89	**32.20±3.96**	279.40±29.15	74.20±2.95	353.00±11.75	**47.20±8.11**	252.80±25.45	53.00±2.32
	NOTEARS	499.80±71.18	93.20±16.67	389.80±118.57	16.80±2.86	503.20±95.46	95.00±23.70	391.40±71.74	16.80±4.97
	DAG-GNN	501.00±99.66	95.40±31.15	391.20±154.19	**14.40±2.45**	497.00±47.54	104.80±13.98	381.40±72.16	10.80±2.36
	GOLEM	659.00±97.70	81.40±29.07	398.00±157.80	179.60±68.04	663.20±54.34	109.40±24.14	361.20±68.11	192.60±36.97
	DAG-NoCurl	2557.80±248.12	158.40±26.26	284.00±56.16	2115.40±272.43	2066.60±186.36	161.00±16.13	310.40±57.45	1595.20±423.12
	ADL	**332.80±10.06**	74.00±8.77	**234.20±3.96**	24.60±2.07	**280.20±7.92**	75.80±5.17	**190.00±3.08**	14.40±3.13
Alarm10	K2	585.60±66.73	118.60±41.47	315.60±83.88	151.40±58.63	545.60±47.48	146.80±51.40	252.00±85.72	146.80±55.10
	OBS	-	-	-	-	-	-	-	-
	Improved-K2	417.60±37.14	47.60±1.84	302.00±18.83	68.00±12.24	323.60±17.06	37.00±2.31	232.20±39.46	54.40±14.03
	GSBN	457.20±6.61	135.60±5.64	317.20±2.17	**4.40±2.61**	399.80±6.30	127.80±6.69	270.80±6.53	**1.20±0.84**
	MMHC	320.80±7.16	46.60±7.37	**167.00±5.29**	107.20±8.64	261.20±12.28	43.00±9.77	**142.00±3.54**	76.20±5.36
	SLL+C	-	-	-	-	-	-	-	-
	SLL+G	436.00±13.13	114.80±10.43	168.80±6.18	152.40±8.20	-	-	-	-
	GGSL	-	-	-	-	489.60±57.15	23.20±8.65	399.40±68.12	67.00±4.78
	NOTEARS	445.40±51.25	27.80±7.97	394.20±133.63	23.40±7.58	441.80±40.89	26.00±7.09	396.40±111.73	19.40±3.19
	DAG-GNN	487.80±60.43	27.00±8.62	438.80±146.74	22.00±4.62	482.60±77.38	33.40±7.46	425.80±47.02	23.40±4.40
	GOLEM	728.60±63.28	22.60±3.79	451.80±92.62	254.20±47.31	792.00±108.06	26.80±5.63	473.20±152.30	292.00±75.17
	DAG-NoCurl	5899.20±537.87	115.80±40.71	304.00±45.68	5479.40±1373.47	1853.60±159.56	88.80±30.75	264.80±77.15	1500.00±157.25
	ADL	**242.40±3.05**	**14.80±9.86**	180.60±4.39	47.00±5.24	**196.80±8.76**	**20.20±11.34**	149.60±3.05	27.00±5.43
Pigs	K2	495.20±21.35	183.00±14.07	146.20±33.78	166.00±15.97	478.80±54.56	188.20±15.89	111.40±10.57	179.20±41.78
	OBS	-	-	-	-	-	-	-	-
	Improved-K2	26.20±5.43	**0.60±0.89**	11.00±2.86	14.60±4.67	9.60±1.47	**0.20±0.45**	**0.00±0.00**	9.40±0.67
	GSBN	304.20±10.76	73.40±5.50	219.20±5.89	11.60±3.44	282.20±8.50	65.80±11.03	210.00±9.59	6.40±1.14
	MMHC	23.80±6.98	10.60±3.05	**0.00±0.00**	13.20±4.09	10.40±5.50	6.80±3.56	**0.00±0.00**	3.60±2.61
	SLL+C	-	-	-	-	-	-	-	-
	SLL+G	-	-	-	-	-	-	-	-
	GGSL	-	-	-	-	-	-	-	-
	NOTEARS	558.20±70.86	149.80±57.24	405.60±41.27	2.80±0.79	582.60±86.71	138.80±45.54	437.40±155.98	6.40±1.38
	DAG-GNN	592.00±0.00	24.00±0.00	568.00±0.00	**0.00±0.00**	592.00±0.00	34.00±0.00	558.00±0.00	**0.00±0.00**
	GOLEM	436.60±63.74	258.20±90.11	72.20±13.51	106.20±23.45	356.80±56.83	247.00±79.07	29.80±6.89	80.00±17.12
	DAG-NoCurl	1075.20±103.74	207.80±72.35	108.60±32.88	758.80±132.55	584.40±52.82	286.80±56.73	131.20±25.48	166.40±61.29
	ADL	**17.20±3.96**	7.00±1.58	**0.00±0.00**	10.20±3.96	**9.20±3.56**	6.20±3.27	**0.00±0.00**	3.00±1.58
Gene	K2	-	-	-	-	-	-	-	-
	OBS	-	-	-	-	-	-	-	-
	Improved-K2	-	-	-	-	-	-	-	-
	GSBN	480.60±11.15	139.40±8.32	328.40±4.16	12.80±1.64	474.20±10.96	135.00±8.75	325.60±2.51	13.60±3.85
	MMHC	150.80±6.72	60.80±4.02	45.60±1.82	44.40±3.78	108.60±13.61	54.40±12.05	28.60±4.34	25.60±4.39
	SLL+C	-	-	-	-	-	-	-	-
	SLL+G	-	-	-	-	-	-	-	-
	GGSL	246.00±32.85	125.00±36.41	**36.00±5.65**	85.00±10.74	160.00±30.14	129.20±41.52	**2.40±0.54**	28.40±6.32
	NOTEARS	820.00±132.11	60.20±19.43	756.40±247.62	3.40±0.74	811.80±131.55	51.20±7.33	757.20±273.65	3.40±0.42
	DAG-GNN	972.00±0.00	**0.00±0.00**	972.00±0.00	**0.00±0.00**	968.00±155.85	**1.40±0.23**	966.00±293.64	**0.60±0.16**
	GOLEM	1533.20±211.29	152.80±19.41	540.00±200.56	840.40±307.01	554.00±68.99	218.80±45.39	296.40±73.12	38.80±6.86
	DAG-NoCurl	7906.00±1190.45	122.40±21.81	255.60±59.73	7528.00±2658.89	1202.40±151.57	356.80±110.88	350.40±77.33	495.20±76.53
	ADL	**135.80±5.81**	58.00±3.81	45.80±2.28	32.00±3.94	**103.20±12.11**	54.60±13.15	27.80±3.35	20.80±3.56
Munin	K2	-	-	-	-	-	-	-	-
	OBS	-	-	-	-	-	-	-	-
	Improved-K2	-	-	-	-	-	-	-	-
	GSBN	**1284.60±151.28**	**4.00±2.35**	1262.40±231.24	**18.20±5.57**	**1222.20±189.49**	**6.00±1.45**	1187.20±223.78	**29.00±6.31**
	MMHC	1918.80±131.54	230.60±47.28	**1049.00±159.74**	639.20±31.33	1727.40±156.77	259.20±41.91	962.20±78.93	506.00±32.17
	SLL+C	-	-	-	-	-	-	-	-
	SLL+G	-	-	-	-	-	-	-	-
	GGSL	-	-	-	-	-	-	-	-
	NOTEARS	-	-	-	-	-	-	-	-
	DAG-GNN	-	-	-	-	-	-	-	-
	GOLEM	-	-	-	-	-	-	-	-
	DAG-NoCurl	-	-	-	-	-	-	-	-
	ADL	1390.00±141.12	60.00±4.78	1060.00±153.57	270.00±27.48	1320.80±127.88	100.20±7.87	**959.20±98.34**	261.40±39.77

1, leading to that some true edges are removed, i.e., more missing edge errors. However, when the sample size of the datasets is large (such as 1,000 samples), the missing edge errors of ADL is almost the same as that of MMHC.

(4) ADL against SLL+C/G and GGSL. Since SLL+C, SLL+G and GGSL all employ score-based algorithms with high time and space complexity to learn the local skeleton of each variable in a dataset, they do not produce the results on some large-sized networks (such as Alarm10, Pigs and Gene). We can see that the SHD value of SLL+C, SLL+G and GGSL is very close, and is all higher than that of ADL. In some large-sized BNs (e.g. Insurance5, Insurance10, Pigs and Gene), the SHD value of SLL+G, GGSL is competitive with that of ADL, since they all employ a score function to orient undirected edges, which is more effective than using CI tests. Specifically, compared with GGSL, ADL reduces the value of SHD by approximately 66% and 70% on Child with 1,000 samples and Alarm with 1,000 samples, respectively. Compared with SLL+C, ADL reduces the value of Reverse by approximately 93% on Alarm with 1,000 samples, and compared with SLL+G, ADL reduces the value of Reverse by approximately 85% on Alarm3 with 500 samples. On Child3 with 500 and 1,000 samples, SLL+C, SLL+G and GGSL achieve a comparable performance against ADL.

(5) ADL against NOTEARS, DAG-GNN, GOLEM and DAG-NoCurl. NOTEARS is specially designed for linear cases instead of nonlinear cases. In addition, NOTEARS adapts convex optimization approaches, its outcome may fall into local optimal solution. Thus, NOTEARS often achieves a higher value of SHD, Reverse, Miss and Extra than ADL in our experimental results. Although DAG-GNN, GOLEM and DAG-NoCurl can be applied to nonlinear cases by adopting different types of neural network models, loss functions and representations of adjacency matrix, their performance is still poor due to the strong theoretical assumptions. From Tables 3 and 4, we can see that DAG-GNN often achieves a much larger number of missing edges than the other algorithms, leading to that it obtains the inaccurate DAG skeleton and the poor quality of the final DAG. GOLEM and DAG-NoCurl always learn many extra edges compared to the other algorithms on most datasets. Specifically, compared with NOTEARS

and DAG-GNN, ADL reduces the value of SHD by more than 85% on both Child and Alarm with 1,000 samples. Compared with NOTEARS, ADL reduces the value of Miss by approximately 96% on Gene with 1,000 samples, and compared with DAG-GNN, ADL reduces the value of Miss by approximately 86% on Child with 1,000 samples. On Gene with 500 samples, the Extra value of ADL is 96% lower than that of GOLEM, and 99% lower than that of DAG-NoCurl.

5.2.2 Structural correctness.

Through the metrics of structural correctness, i.e. Ar_F1, Ar_Precision and Ar_Recall, Tables 5 and 6 report the quality of DAG learned by different algorithms. We find that on most BNs (such as Child, Alarm, Alarm10, Pigs and Gene), ADL not only achieves fewer structural errors than its rivals in terms of SHD metric, bust also achieves more structural correctness than other algorithms in terms of Ar_F1 metric. In addition, on all datasets, ADL is able to maintain the value of Ar_F1, Ar_Precision and Ar_Recall at consistently high levels against its rivals.

Specifically, for Ar_F1 metric, our method achieves clear improvements of approximately 65% more than K2 on Alarm with 1,000 samples, 59% more than OBS on Alarm with 1,000 samples, 21% more than Improved-K2 on Alarm3 with 500 samples, 56% more than GSBN on Alarm with 500 samples, 10% more than MMHC on Child with 500 samples, 22% more than SSL+C on Alarm5 with 500 samples, 24% more than SSL+G on Alarm3 with 500 samples, 36% more than GGSL on Alarm5 with 500 samples, 67% more than NOTEARS on Child with 1,000 samples, 48% more than DAG-GNN on Alarm10 with 1,000 samples, 65% more than GOLEM on Gene with 500 samples, and 70% more than DAG-NoCurl on Pigs with 1,000 samples. In particular, DAG-GNN is not scalable to high-dimensional data since the DAGs learned by DAG-GNN are almost empty graphs on large-sized networks, such as Pigs and Gene. Compared with its rivals, ADL obtain higher values of Ar_Precision and Ar_Recall on most datasets, since ADL adopts adaptive strategy to construct more accurate global skeletons, that is, some missing edges can be restored and some extra edges are able to be removed.

For Ar_Precision metric, our method achieves clear improvements of approximately 69% more than K2 on Alarm with 1,000 samples, 53% more than OBS on Alarm5 with 1,000 samples, 20% more than Improved-K2 on Alarm3 with 500 samples, 45% more than GSBN on Alarm3 with 500 samples, 16% more than MMHC on Alarm10 with 500 samples, 36% more than SSL+C on Alarm5 with 500 samples, 31% more than SSL+G on Alarm3 with 500 samples, 39% more than GGSL on Alarm3 with 500 samples, 50% more than NOTEARS on Child with 1,000 samples, 43% more than DAG-GNN on Insurance5 with 500 samples, 72% more than GOLEM on Alarm3 with 500 samples, and 70% more than DAG-NoCurl on Pigs with 1,000 samples.

For Ar_Recall metric, our method achieves clear improvements of approximately 62% more than K2 on Alarm with 1,000 samples, 48% more than OBS on Alarm3 with 500 samples, 22% more than Improved-K2 on Alarm3 with 500 samples, 60% more than GSBN on Alarm with 500 samples, 10% more than MMHC on Child with 500 samples, 15% more than SSL+C on Alarm with 1,000 samples, 20% more than SSL+G on Child with 1,000 samples, 36% more than GGSL on Alarm5 with 500 samples, 96% more than NOTEARS on Pigs with 1,000 samples, 68% more than DAG-GNN on Child with 1,000 samples, 60% more than GOLEM on Gene with 500 samples, and 64% more than DAG-NoCurl on Gene with 1,000 samples.

From Tables 5 and 6, we also find that the local-to-global DAG learning methods using the AND-rule (e.g. MMHC) can obtain a higher Ar_Precision value than other methods since the algorithms employing the AND-rule can remove some redundant false directed edges. In contrast, the local-to-global DAG learning methods using the OR-rule (e.g. SLL+C) can achieve a higher Ar_Recall value

TABLE 5
Comparison of correctly learned edge directions (in %) (1)

	#Sample	500			1000		
Network	Algorithm	Ar_F1 (↑)	Ar_Precision (↑)	Ar_Recall (↑)	Ar_F1 (↑)	Ar_Precision (↑)	Ar_Recall (↑)
Child	K2	25.00±1.49	26.09±4.42	24.00±7.32	27.91±2.33	33.33±5.64	24.00±4.38
	OBS	31.11±7.49	35.00±2.66	28.00±2.17	42.55±1.71	45.45±6.76	40.00±1.30
	Improved-K2	64.00±2.39	74.11±1.46	56.32±5.36	78.64±6.27	84.00±4.68	73.92±8.45
	GSBN	56.96±5.67	86.79±7.57	42.40±4.56	67.66±2.53	89.50±3.52	54.40±2.19
	MMHC	72.01±11.57	77.58±11.60	67.20±11.45	90.89±1.92	94.17±3.20	**88.00±4.00**
	SLL+C	77.40±10.52	84.92±11.88	71.20±9.96	86.05±4.69	94.23±4.12	79.20±5.22
	SLL+G	68.13±8.25	77.55±9.15	60.80±7.69	75.05±6.09	82.61±5.60	68.80±6.57
	GGSL	64.86±9.98	72.97±11.02	58.40±9.21	69.12±7.88	76.57±11.13	63.20±6.57
	NOTEARS	22.86±5.02	40.00±5.62	16.00±3.84	23.53±2.85	44.44±2.40	16.00±4.75
	DAG-GNN	25.64±2.15	35.71±8.08	20.00±1.16	27.03±1.23	41.67±3.74	20.00±1.34
	GOLEM	18.18±1.35	16.67±1.64	20.00±1.83	22.64±1.40	21.43±6.78	24.00±2.83
	DAG-NoCurl	13.79±5.47	12.12±5.09	16.00±0.92	14.55±5.33	13.33±8.22	16.00±3.96
	ADL	**82.40±10.69**	**87.95±9.99**	**77.60±11.52**	**91.27±1.80**	**94.85±1.79**	**88.00±2.83**
Alarm	K2	28.89±7.95	29.55±6.33	28.26±6.40	29.17±6.06	28.00±6.56	30.43±3.96
	OBS	35.16±1.01	35.56±7.38	34.78±7.98	35.42±5.74	34.00±1.38	36.96±7.97
	Improved-K2	74.54±4.10	81.23±4.17	68.87±5.80	82.00±2.25	85.90±1.35	78.43±1.38
	GSBN	34.17±2.43	49.60±3.08	26.09±2.17	43.92±4.93	57.23±5.35	35.65±4.51
	MMHC	87.38±7.37	89.20±7.86	85.65±6.98	89.80±5.92	90.08±6.81	89.57±5.41
	SLL+C	76.07±3.73	78.38±4.21	73.91±3.44	77.92±5.29	78.52±6.00	77.39±5.00
	SLL+G	73.67±9.94	76.20±9.85	71.30±10.01	79.30±8.59	79.94±7.68	78.70±9.53
	GGSL	72.64±4.84	73.58±6.97	72.17±6.77	79.77±3.77	82.98±5.89	76.96±3.64
	NOTEARS	57.89±8.38	73.33±7.37	47.83±6.78	61.33±2.00	79.31±2.63	50.00±6.88
	DAG-GNN	58.14±4.61	62.50±1.92	54.35±3.69	61.54±4.40	62.22±6.65	60.87±3.67
	GOLEM	35.40±1.57	29.85±0.75	43.48±8.01	49.48±2.68	47.06±0.80	52.17±5.89
	DAG-NoCurl	26.67±2.91	19.23±2.86	43.48±3.16	33.33±3.94	25.58±4.11	47.83±5.38
	ADL	**90.35±6.67**	**95.08±5.89**	**86.09±7.31**	**94.90±3.99**	**97.32±4.82**	**92.61±3.30**
Child3	K2	29.20±4.24	34.48±5.69	25.32±0.70	37.24±0.98	40.91±3.03	34.18±6.68
	OBS	40.29±7.24	46.67±4.97	35.44±7.33	41.33±6.07	43.66±1.50	39.24±1.54
	Improved-K2	67.11±6.66	77.73±3.30	59.04±5.80	69.44±1.01	75.06±3.74	64.61±4.09
	GSBN	37.29±1.86	62.47±2.04	26.58±1.55	46.43±1.36	64.72±2.08	36.20±1.13
	MMHC	68.48±6.75	79.32±7.72	60.25±6.04	74.67±2.86	83.45±3.01	67.59±3.18
	SLL+C	70.96±5.50	79.25±6.80	**64.30±5.09**	74.78±4.87	83.74±5.68	67.59±4.62
	SLL+G	70.26±5.56	80.23±6.45	62.53±5.26	76.69±5.26	**86.58±5.08**	68.86±5.49
	GGSL	60.87±16.75	79.23±6.92	52.91±20.98	69.95±19.27	85.25±5.06	63.29±23.19
	NOTEARS	24.24±3.28	60.00±4.07	15.19±0.93	26.80±1.24	72.22±0.56	16.46±3.88
	DAG-GNN	28.80±1.92	39.13±5.80	22.78±3.15	34.48±5.74	54.05±6.28	25.32±4.75
	GOLEM	14.29±7.69	14.67±1.45	13.92±8.41	15.38±1.37	13.59±5.55	17.72±1.51
	DAG-NoCurl	9.18±1.82	7.69±2.16	11.39±1.50	9.41±1.57	8.79±1.29	10.13±1.64
	ADL	**71.64±4.83**	**81.80±4.53**	63.80±5.34	**77.07±4.66**	86.41±3.63	**69.62±5.52**
Alarm3	K2	31.00±2.80	34.43±3.82	28.19±4.22	35.04±1.85	38.40±2.07	32.21±3.04
	OBS	31.82±6.61	36.52±7.05	28.19±1.30	35.46±3.03	37.59±2.24	33.56±2.51
	Improved-K2	61.57±3.83	71.02±7.24	54.34±7.16	69.21±3.43	78.31±6.61	62.01±5.52
	GSBN	30.85±1.87	45.44±2.77	23.36±1.46	45.70±3.21	60.34±3.92	36.78±2.74
	MMHC	76.25±4.26	79.35±4.92	73.42±4.07	79.77±4.84	82.73±6.23	77.05±3.66
	SLL+C	61.74±4.19	60.23±4.74	63.36±3.84	67.35±3.63	65.47±4.38	69.40±3.34
	SLL+G	59.12±5.21	60.32±5.69	57.99±4.88	61.82±7.04	62.63±7.75	61.07±6.51
	GGSL	47.83±2.63	52.61±4.62	44.30±4.91	59.81±4.11	63.53±3.17	56.64±5.71
	NOTEARS	43.81±1.92	75.41±3.38	30.87±0.95	44.44±7.64	79.31±6.13	30.87±4.95
	DAG-GNN	42.01±4.68	65.71±3.19	30.87±1.91	41.51±1.98	69.84±2.20	29.53±1.12
	GOLEM	18.49±2.17	18.88±7.74	18.12±5.90	23.13±7.81	21.64±6.15	24.83±4.96
	DAG-NoCurl	22.59±4.25	15.60±3.80	40.94±1.33	31.94±3.01	26.18±1.83	40.94±5.48
	ADL	**83.13±4.27**	**91.03±5.84**	**76.51±3.15**	**83.69±3.74**	**90.48±3.95**	**77.85±3.65**
Insurance5	K2	29.35±6.46	40.88±6.39	22.89±4.99	31.06±8.40	37.69±1.86	26.41±2.56
	OBS	38.84±1.97	53.05±5.28	30.63±2.90	39.57±3.67	50.00±1.09	32.75±5.97
	Improved-K2	47.38±2.55	66.28±5.41	36.87±5.16	48.36±6.68	58.52±7.96	41.21±8.28
	GSBN	32.87±1.60	61.76±3.05	22.39±1.13	44.38±0.77	75.59±1.24	31.41±0.58
	MMHC	58.34±3.25	73.36±3.90	48.45±2.95	57.55±1.74	68.44±1.80	49.65±1.69
	SLL+C	50.79±1.40	62.43±1.78	42.82±1.18	60.82±3.82	73.05±5.24	52.11±3.00
	SLL+G	52.19±3.17	69.20±4.29	41.90±2.55	58.67±3.21	74.55±3.21	48.38±3.02
	GGSL	58.68±2.47	71.61±3.89	**49.93±3.88**	**64.94±4.13**	**80.14±1.04**	**54.79±5.64**
	NOTEARS	19.05±1.57	38.30±2.20	12.68±7.66	19.15±3.72	39.13±8.36	12.68±3.72
	DAG-GNN	14.61±1.07	36.11±2.44	9.15±0.93	15.30±5.47	34.15±1.73	9.86±3.55
	GOLEM	18.39±4.03	25.31±0.61	14.44±7.68	9.69±1.79	17.59±6.56	6.69±7.47
	DAG-NoCurl	11.01±2.07	8.01±1.25	17.61±2.96	15.22±3.31	11.58±1.98	22.18±2.85
	ADL	**60.87±2.43**	**79.21±2.47**	49.44±2.29	62.95±5.43	78.07±6.35	52.75±4.73
Alarm5	K2	29.18±4.15	33.83±1.31	25.66±8.46	29.22±4.75	32.13±7.16	26.79±5.28
	OBS	26.75±3.16	31.94±2.88	23.02±1.00	34.08±3.18	36.84±2.89	31.70±4.12
	Improved-K2	59.05±4.83	69.86±7.46	51.14±2.62	66.60±2.04	75.73±1.61	59.44±6.07
	GSBN	28.92±1.99	43.79±2.96	21.58±1.50	40.26±2.38	55.65±3.01	31.55±2.01
	MMHC	70.45±2.96	73.89±2.80	67.32±3.12	77.84±3.15	81.98±3.34	**74.11±3.10**
	SLL+C	55.33±2.78	51.83±3.48	59.40±2.21	60.50±1.56	57.02±1.57	64.45±1.93
	SLL+G	55.02±2.74	56.55±2.31	53.58±3.26	60.39±5.54	61.34±5.38	59.47±5.70
	GGSL	40.99±4.32	54.34±4.23	33.13±4.72	51.52±3.79	59.37±2.16	45.81±5.87
	NOTEARS	42.98±2.89	79.59±0.87	29.43±4.54	43.29±3.88	79.00±3.38	29.81±4.97
	DAG-GNN	40.31±6.59	65.81±5.55	29.06±1.22	42.22±6.44	70.18±3.89	30.19±3.93
	GOLEM	17.36±1.15	19.18±6.72	15.85±7.74	13.81±1.50	16.85±0.70	11.70±2.82
	DAG-NoCurl	18.35±4.77	11.58±1.37	44.15±7.11	30.36±3.04	24.06±5.73	41.13±8.16
	ADL	**77.44±2.26**	**87.42±3.14**	**69.51±1.82**	**81.22±2.37**	**90.63±3.03**	73.58±1.89

than other methods since the algorithms employing the OR-rule can recover some missed true directed edges. Further, by adaptively using the AND-rule and the OR-rule, our method not only guarantees the high Ar_Precision value, but also ensures the high Ar_Recall value, thus our method can obtain a higher Ar_F1 score than its seven rivals.

On the whole, our method is obviously superior to other algorithms on all *sparse* networks (such as Child, Alarm, Child3, Alarm3, Alarm5, Alarm10 and Gene) and most *dense* networks (such as Insurance10 and Pigs).

TABLE 6
Comparison of correctly learned edge directions (in %) (2)

	#Sample	500			1000		
Network	Algorithm	Ar_F1 (↑)	Ar_Precision (↑)	Ar_Recall (↑)	Ar_F1 (↑)	Ar_Precision (↑)	Ar_Recall (↑)
	K2	31.49±3.20	43.63±2.85	24.64±6.47	33.48±7.99	41.10±4.16	28.24±2.42
	OBS	-	-	-	-	-	-
	Improved-K2	47.96±3.04	64.14±1.45	38.30±8.02	56.69±1.25	68.35±4.70	48.43±4.74
	GSBN	33.40±1.30	61.09±2.27	22.99±0.91	44.51±1.77	73.98±2.39	31.83±1.37
	MMHC	53.37±1.36	66.35±1.14	**44.64±1.42**	61.16±2.03	71.28±2.40	**53.56±1.77**
	SLL+C	49.13±2.03	57.27±2.91	43.02±1.58	-	-	-
Insurance10	SLL+G	50.42±2.09	66.12±2.95	40.76±1.69	-	-	-
	GGSL	53.54±5.59	69.54±2.85	43.53±7.45	56.02±7.89	71.51±3.96	46.04±9.11
	NOTEARS	20.03±0.58	40.44±0.89	13.31±5.84	19.00±6.61	38.67±6.57	12.59±6.43
	DAG-GNN	19.05±5.33	39.11±4.71	12.59±6.34	19.16±6.45	38.38±1.35	12.77±5.95
	GOLEM	17.25±6.16	22.85±6.75	13.85±2.80	18.24±4.21	22.22±2.20	15.47±1.29
	DAG-NoCurl	7.75±1.04	4.78±0.95	20.50±3.67	7.09±1.09	4.62±0.90	15.29±1.81
	ADL	**54.92±2.08**	**71.53±2.48**	44.57±1.78	**61.98±1.27**	**76.29±1.60**	52.19±1.06
	K2	28.07±0.99	33.74±6.74	24.04±3.20	33.27±5.83	37.07±7.66	30.18±4.63
	OBS	-	-	-	-	-	-
	Improved-K2	57.88±5.66	69.88±4.34	49.40±5.61	65.50±7.39	74.89±4.38	58.20±3.65
	GSBN	28.33±1.55	45.56±2.29	20.56±1.17	39.38±1.44	57.06±1.90	30.07±1.22
	MMHC	65.98±0.81	69.85±0.50	62.53±1.11	71.68±1.98	76.36±2.13	67.54±1.94
	SLL+C	-	-	-	-	-	-
Alarm10	SLL+G	50.98±1.69	51.75±1.94	50.25±1.57	-	-	-
	GGSL	-	-	-	36.63±10.56	62.18±8.47	25.96±4.88
	NOTEARS	38.75±5.36	74.87±6.43	26.14±1.34	38.79±6.12	76.68±1.73	25.96±8.13
	DAG-GNN	29.01±1.52	68.18±4.90	18.42±4.38	30.35±4.83	66.67±5.94	19.65±0.79
	GOLEM	20.57±7.62	26.01±6.89	17.02±6.37	14.81±6.97	18.25±6.49	12.46±1.46
	DAG-NoCurl	4.78±0.91	2.63±1.08	26.49±1.21	18.35±4.70	12.07±3.11	38.25±4.87
	ADL	**74.44±1.29**	**85.84±1.20**	**65.72±1.38**	**78.67±1.97**	**89.45±2.21**	**70.21±1.93**
	K2	43.56±4.56	42.88±6.78	44.26±4.12	46.81±5.51	44.39±3.18	49.49±2.81
	OBS	-	-	-	-	-	-
	Improved-K2	86.66±4.86	86.52±5.68	86.81±4.85	87.63±5.87	87.26±6.43	88.00±4.66
	GSBN	61.33±1.50	77.89±2.00	50.57±1.25	64.51±1.60	81.44±2.46	53.41±1.32
	MMHC	97.13±0.82	96.07±1.12	98.21±0.52	98.55±0.75	98.26±0.91	98.85±0.60
	SLL+C	-	-	-	-	-	-
Pigs	SLL+G	-	-	-	-	-	-
	GGSL	-	-	-	-	-	-
	NOTEARS	9.73±1.56	20.11±6.28	6.42±1.38	4.52±3.69	10.56±3.82	2.87±1.95
	DAG-GNN	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
	GOLEM	43.02±5.73	41.85±6.49	44.26±5.17	51.17±5.73	49.14±7.96	53.38±1.81
	DAG-NoCurl	30.21±6.42	22.30±2.38	46.79±6.38	28.71±7.87	27.91±6.86	29.56±5.12
	ADL	**97.97±0.38**	**97.15±0.64**	**98.82±0.27**	**98.70±0.56**	**98.45±0.60**	**98.95±0.55**
	K2	-	-	-	-	-	-
	OBS	-	-	-	-	-	-
	Improved-K2	-	-	-	-	-	-
	GSBN	61.92±1.22	76.81±1.33	51.87±1.12	62.67±1.16	77.49±1.50	52.61±0.96
	MMHC	89.11±0.50	89.16±0.55	89.05±0.50	91.60±1.28	91.74±1.14	91.46±1.45
	SLL+C	-	-	-	-	-	-
Gene	SLL+G	-	-	-	-	-	-
	GGSL	81.28±3.22	79.33±1.89	83.33±4.69	85.42±6.89	84.35±4.54	86.52±8.15
	NOTEARS	26.20±3.43	71.23±3.45	16.05±5.98	27.56±4.02	75.23±2.56	16.87±6.52
	DAG-GNN	0.00±0.00	0.00±0.00	0.00±0.00	1.02±0.56	83.33±7.59	0.51±0.13
	GOLEM	24.96±2.15	22.01±1.19	28.81±6.68	54.33±5.87	64.15±6.22	47.12±5.64
	DAG-NoCurl	2.64±1.15	1.40±0.61	22.43±3.91	25.47±3.85	23.81±2.63	27.37±4.03
	ADL	**89.96±0.43**	**90.61±0.48**	**89.32±0.47**	**91.85±1.28**	**92.18±1.13**	**91.52±1.44**
	K2	-	-	-	-	-	-
	OBS	-	-	-	-	-	-
	Improved-K2	-	-	-	-	-	-
	GSBN	16.90±2.57	**85.62±5.17**	9.38±1.95	24.94±3.25	**85.36±6.74**	14.60±2.85
	MMHC	9.90±0.86	11.96±1.27	8.45±1.57	15.06±2.21	18.70±3.44	12.60±1.91
	SLL+C	-	-	-	-	-	-
Munin	SLL+G	-	-	-	-	-	-
	GGSL	-	-	-	-	-	-
	NOTEARS	-	-	-	-	-	-
	DAG-GNN	-	-	-	-	-	-
	GOLEM	-	-	-	-	-	-
	DAG-NoCurl	-	-	-	-	-	-
	ADL	**27.64±3.37**	45.63±4.57	**19.83±2.33**	**32.25±2.89**	48.35±4.51	**24.19±3.57**

5.2.3 Time efficiency.

Table 7 shows that ADL is significantly faster than K2, OBS, Improved-K2, SLL+C, SLL+G, GGSL, NOTEARS, DAG-GNN, GOLEM and DAG-NoCurl on most datasets (especially high-dimensional datasets), and is competitive with the GSBN and MMHC algorithms. On most datasets, the local-to-global DAG learning algorithms (such as MMHC and ADL) are much more efficient than the global DAG learning algorithms (such as K2 and NOTEARS). In addition, we can see that K2, OBS, Improved-K2, SLL+C, SLL+G and GGSL are not scalable to high-dimensional datasets, thus they do not produce the results on Insurance10, Alarm10, Pigs, Gene and Munin networks with both 500 and 1,000 samples. In practice, only through GPU acceleration can continuous optimization based DAG learning methods (i.e., NOTEARS, DAG-GNN, GOLEM and DAG-NoCurl) be scalable to high-dimensional datasets.

The main time cost of local-to-global DAG learning algorithms lies in the local skeleton learning phase. GSBN, MMHC and ADL use more efficient constraint-based PC/MB learning algorithms to learn the local skeletons, whereas SLL+C/G and GGSL adopt more inefficient score-based PC/MB learning algorithms to discover the local skeleton of each variable. As a result, on all datasets with both 500 and 1,000 samples, GSBN, MMHC and ADL are generally faster than SLL+C/G and GGSL. Although ADL uses a similar orientation strategy to MMHC in Step 3 of ADL, the HITON-PC algorithm [37] employed by ADL is more efficient than the MMPC algorithm [36] employed by MMHC on benchmark BN datasets. Secondly, we have also deeply optimized the ADL codes to make ADL faster than MMHC on many BNs.

GSBN employs the GS algorithm [34] to learn the local skeleton of each variable in a dataset, and the time cost of GS is linear with the size of the learned local skeletons. In contrast, ADL employs the HITON-PC algorithm [37] to learn the local skeletons, and the efficiency of HITON-PC is exponential with the size of the learned local skeletons. However, in practice, ADL is faster than GSBN since the local skeleton size of variables on most BNs is small. In particular, on Pigs BN, ADL is slightly slower than GSBN since there are several variables with a large size of local skeletons. OBS is always slower than K2 since OBS uses an exhaustive approach to find the highest scoring DAG by constantly updating the order of nodes, and none of them can be scalable to high-dimensional data (such as Pigs, Gene and Munin) due to the frequent running of scoring functions. Since DAG-NoCurl is designed based on the graph Hodge theory [41] and solves the resultant unconstrained optimization problem, it is more efficient than other continuous optimization based DAG learning methods (i.e., NOTEARS, DAG-GNN and GOLEM).

From Table 7, we also find that the running time of an algorithm is not always positively correlated with the sample size of a dataset. For example, 1) the running time of NOTEARS and DAG-NoCurl on Pigs with 1,000 samples is faster than that on Pigs with 500 samples since the running time of NOTEARS and DAG-NoCurl depends only on the number of iterations; 2) the running time of GGSL on Insurance10 with 1,000 samples is faster than that on Insurance10 with 500 samples since the running time of GGSL is related to the local skeleton size of the variable initially randomly selected; 3) the running time of K2 on Alarm with 500 samples is slower than that on Alarm with 1,000 samples since the efficiency of this algorithm is highly dependent on the given initial node ordering.

5.2.4 BIC score of each algorithm.

Just using the structure evaluation metrics, SHD and Ar_F1, it is not clear how good the DAGs are as probability models. The ultimate evaluation of the accuracy of a learned DAG

TABLE 7
Running time (in second) of ADL and its rivals.

#Sample	Network	K2	OBS	Improved-K2	GSBN	MMHC	SLL+C	SLL+G	GGSL	NOTEARS	DAG-GNN	GOLEM	DAG-NoCurl	ADL
500	Child	0.22±0.02	4.55±0.44	0.34±0.04	0.21±0.01	0.14±0.01	0.93±0.82	0.68±0.46	1.48±0.93	7.47±1.12	94.91±10.74	46.70±7.52	0.63±0.06	**0.12±0.01**
	Alarm	1.52±0.15	42.46±3.59	1.83±0.21	0.47±0.01	0.34±0.02	1.94±0.33	1.34±0.24	5.71±1.03	19.08±1.60	193.22±30.32	60.90±7.79	3.29±0.41	**0.23±0.02**
	Child3	3.51±0.30	249.39±38.73	5.11±0.47	1.01±0.01	0.73±0.02	6.96±2.29	3.40±0.50	19.62±12.62	20.57±2.72	281.09±29.54	82.53±10.01	2.72±0.30	**0.47±0.02**
	Alarm3	23.96±1.86	3107.95±292.12	28.18±2.26	2.95±0.06	2.01±0.04	23.21±5.71	18.59±4.55	128.19±26.64	81.64±6.72	574.82±50.81	209.18±19.66	20.29±2.27	**1.02±0.02**
	Insurance5	48.57±3.64	7877.24±1262.50	57.03±5.18	3.71±0.05	3.51±0.14	52.22±7.32	39.49±4.91	358.25±24.32	404.97±66.61	647.30±77.08	224.99±26.76	47.51±4.93	**2.05±0.15**
	Alarm5	118.44±18.95	975.05±104.26	138.42±25.37	7.73±0.20	5.14±0.12	121.60±29.04	93.86±9.25	731.55±174.42	404.60±32.82	848.45±125.59	499.40±54.42	98.63±9.29	**2.56±0.06**
	Insurance10	538.45±59.44	-	569.17±66.28	13.95±0.12	13.39±0.25	441.18±236.44	392.27±206.93	2945.78±22.96	2308.24±192.04	1389.93±228.23	1258.58±208.44	247.13±31.51	**7.70±0.37**
	Alarm10	2192.35±166.57	-	2191.00±180.91	48.00±3.75	20.07±0.36	-	1686.52±360.97	-	3082.26±324.61	1900.76±289.14	3427.69±245.22	1085.39±80.65	**9.81±0.21**
	Pigs	7172.24±637.11	-	6339.35±569.32	46.26±1.95	**33.41±0.18**	-	-	-	10808.95±1547.54	2522.60±339.98	3874.82±445.96	1163.99±145.15	48.84±2.43
	Gene	-	-	-	770.45±18.48	108.79±11.54	-	-	13888.01±2159.52	28051.44±2493.65	5984.02±829.85	16061.95±1419.09	5293.93±565.65	**46.37±2.89**
	Munin	-	-	-	410.37±4.65	451.46±7.39	-	-	-	-	-	-	-	**38.48±0.27**
1000	Child	0.25±0.03	5.44±0.80	0.39±0.05	0.27±0.00	0.19±0.02	1.57±0.47	1.31±0.38	2.66±0.70	6.93±0.54	209.98±34.21	50.51±7.45	0.63±0.07	**0.14±0.01**
	Alarm	1.22±0.14	49.89±5.72	1.67±0.28	0.63±0.03	0.36±0.01	2.90±0.23	1.88±0.08	8.54±1.21	13.32±1.34	360.95±43.62	63.15±7.65	5.58±0.85	**0.25±0.00**
	Child3	3.95±0.59	303.71±40.83	5.88±1.21	1.38±0.05	0.88±0.04	8.23±3.57	6.02±2.56	19.20±9.67	25.75±2.78	569.01±86.01	88.03±10.85	3.30±0.35	**0.59±0.02**
	Alarm3	26.61±4.36	3567.20±562.17	31.92±6.87	4.10±0.19	2.12±0.03	28.24±4.27	22.19±2.58	138.14±25.50	88.36±11.05	1034.86±136.86	217.99±28.06	20.07±1.82	**1.18±0.03**
	Insurance5	60.49±6.06	9968.35±1178.93	70.51±8.31	4.75±0.06	4.28±0.10	80.19±5.61	61.68±4.73	452.75±23.89	213.93±19.91	1288.04±198.91	243.84±21.82	40.40±3.74	**2.56±0.10**
	Alarm5	140.97±12.27	1176.19±109.11	160.82±17.54	10.18±0.07	5.44±0.12	126.09±24.80	106.43±20.22	685.92±122.14	332.25±37.73	1718.73±173.78	530.30±86.09	61.46±6.95	**2.88±0.05**
	Insurance10	636.74±56.34	-	693.56±71.46	17.47±0.55	14.94±0.19	-	-	1738.23±315.86	1221.33±205.15	2414.10±274.93	1114.13±90.37	199.92±19.15	**8.54±0.08**
	Alarm10	2543.79±282.04	-	2345.07±237.47	56.79±1.03	20.56±0.15	-	-	6614.82±518.54	1797.46±172.95	3533.80±460.40	3364.58±474.81	534.57±49.27	**10.76±0.11**
	Pigs	8287.27±912.45	-	8100.89±874.02	52.09±1.59	**37.59±0.87**	-	-	-	6290.18±640.83	4767.97±536.00	4084.71±493.38	284.05±22.31	52.21±2.09
	Gene	-	-	-	774.33±55.12	130.50±1.16	-	-	16251.11±4156.89	14627.13±1066.64	9586.93±1561.57	15729.75±2249.87	3480.01±413.64	**52.24±0.56**
	Munin	-	-	-	702.38±10.41	689.72±9.53	-	-	-	-	-	-	-	**49.98±0.79**

is how close it represents the true probability distribution, thus we compare the BIC (Bayesian information criterion) score [40] of each algorithm shown in Table 8. As a commonly used information-theoretic score, BIC score can avoid over-fitting by balancing the goodness of fit with DAG dimensionality given the available data.

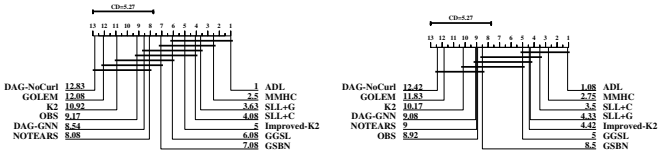
As we can see from Table 8, ADL improves the learning scores by a significant margin over other algorithms on most datasets. In the edge orientation step, as both MMHC and SLL+G use a search strategy similar to ADL to find the optimal structure, so they achieve a comparable performance against ADL on some datasets (such as Child3, Insurance5 and Insurance10). The BIC scores of OBS and Improved-K2 are competing with that of ADL on most datasets, since OBS employs a systematic search of ordering space to find the highest scoring DAG consistent with each ordering and Improved-K2 can obtain a high-quality ordering of variables as an initial input of K2.

5.2.5 *Statistical Tests.*

To give a comprehensive performance comparison between ADL with its rivals, the Friedman test and Nemenyi test [42] are performed.

We first perform the Friedman test at the 0.05 significance level under the null-hypothesis, which states that the performance of all algorithms is the same on all datasets (i.e., the average ranks of all algorithms are equivalent). Then, we perform the Nemenyi test, which states that the performance of two algorithms is significantly different if the corresponding average ranks differ by at least one critical difference (CD). Note that we only perform these tests on the datasets where all algorithms can run the results.

Figs. 6(a) and (b) provide the CD diagrams, where the average rank of each algorithm is marked along the axis (lower ranks to the right). For the SHD metric, we observe that ADL achieves a comparable performance against MMHC, SLL+G, SLL+C, Improved-K2 and GGSL, and it is significantly better than the other algorithms. For Ar_F1 metric, we note that ADL significantly outperforms GSBN, OBS, NOTEARS, DAG-GNN, K2, GOLEM and DAG-NoCurl, and it achieves a comparable performance against the other algorithms. On the whole, ADL is the only algorithm that achieves the lowest rank value on both the SHD and Ar_F1 metrics.



(a) SHD metric (b) Ar_F1 metric

Fig. 6. Comparison of ADL against its rivals with the Nemenyi test. (the lower the rank value, the better the performance.)

5.3 Results of DAG learning on real data

In this section, we compare our proposed algorithm with K2, OBS, Improved-K2, GSBN, MMHC, SLL+C, SLL+G, GGSL, NOTEARS, DAG-GNN, GOLEM and DAG-NoCurl on a real dataset Sachs [39] (Sachs is described in Section 5.1.2), and experimental results are summarized in Table 9.

From Table 9, we can obtain the following observations.

1) Our proposed method achieves the best results on all metrics (i.e., Ar_F1, Ar_Precision, Ar_Recall, SHD, Reverse, Miss and Extra).
2) The existing DAG learning algorithms do not perform well on real data, only our method improves the F1 metric by more than 50%.
3) Continuous optimization based DAG learning methods usually learn a worse structure than other algorithms, this is probably because such methods are usually designed for specific scenarios or with strong assumptions.
4) The local-to-global DAG learning methods (i.e., ADL, GSBN, MMHC, SLL+C, SLL+G and GGSL) outperform the global DAG learning methods, since the local-to-global methods can use the local skeleton learning algorithms to construct an accurate global skeleton.

On the whole, by employing the adaptive skeleton construction strategy, our proposed method not only achieves a good performance on the benchmark datasets, but also obtains the best results on the real dataset.

6 Conclusion

This paper focuses on the issue of local-to-global DAG structure learning. Existing methods construct the skeleton of a DAG by simply using either the AND-rule or the OR-rule,

TABLE 8
Learning BIC scores for different DAG learning algorithms on different datasets.

#Sample	Network	K2	OBS	Improved-K2	GSBN	MMHC	SLL+C	SLL+G	GGSL	NOTEARS	DAG-GNN	GOLEM	DAG-NoCurl	ADL
500	Child	-7.096280E+03	-7.049930E+03	-6.853100E+03	-7.249940E+03	-6.664490E+03	-6.681860E+03	-6.673250E+03	-7.151060E+03	-7.429350E+03	-9.149990E+03	-7.501080E+03	-1.396654E+04	**-6.648140E+03**
	Alarm	-5.983980E+03	-5.921090E+03	-5.920050E+03	-7.176670E+03	-5.970760E+03	-5.950140E+03	-5.932080E+03	-6.912380E+03	-7.779360E+03	-1.339617E+04	-9.019570E+03	-3.050004E+04	**-5.914840E+03**
	Child3	-2.169417E+04	-2.151504E+04	-2.054698E+04	-2.195435E+04	-2.049381E+04	-2.063815E+04	-2.055745E+04	-2.202312E+04	-2.351920E+04	-2.597515E+04	-2.384051E+04	-2.842927E+05	**-2.047510E+04**
	Alarm3	-2.043430E+04	-2.042601E+04	**-2.028200E+04**	-2.201467E+04	-2.081604E+04	-2.117098E+04	-2.070433E+04	-2.597413E+04	-2.301917E+04	-2.321901E+04	-2.813889E+04	-9.659642E+07	-2.067156E+04
	Insurance5	-4.470477E+04	-4.373956E+04	**-4.371751E+04**	-4.826884E+04	-4.521122E+04	-4.543234E+04	-4.425777E+04	-4.940019E+04	-5.113285E+04	-5.094571E+04	-5.366885E+04	-4.300718E+07	-4.497184E+04
	Alarm5	-3.481331E+04	-3.457681E+04	**-3.442752E+04**	-3.741840E+04	-3.528747E+04	-3.727844E+04	-3.584899E+04	-4.709139E+04	-3.891680E+04	-3.915974E+04	-4.765098E+04	-4.578304E+04	-3.504698E+04
	Insurance10	-8.988036E+04	-	-8.910902E+04	-9.709363E+04	-9.063090E+04	-9.269465E+04	**-8.900159E+04**	-1.051684E+05	-1.035574E+05	-1.032492E+05	-1.066020E+05	-5.604573E+07	-9.084952E+04
	Alarm10	-7.073688E+04	-	-6.921966E+04	-7.425935E+04	-6.978447E+04	-	-6.944316E+04	-	-7.657934E+04	-8.004505E+04	-9.006527E+04	-5.695415E+08	**-6.921624E+04**
	Pigs	-1.963104E+05	-	-1.918651E+05	-2.016526E+05	-1.814861E+05	-	-	-	-2.171722E+05	-2.201938E+05	-1.937926E+05	-2.885710E+05	**-1.814518E+05**
	Gene	-	-	-	-2.603939E+05	-2.412176E+05	-	-	-2.516894E+05	-2.895122E+05	-3.198638E+05	-3.016782E+05	-2.965350E+06	**-2.409292E+05**
	Munin	-	-	-	-1.661008E+05	-3.445156E+05	-	-	-	-	-	-	-	**-1.522743E+05**
1000	Child	-1.357099E+04	-1.344080E+04	-1.317795E+04	-1.365749E+04	-1.279651E+04	-1.287878E+04	-1.287143E+04	-1.318454E+04	-1.449615E+04	-1.489761E+04	-1.522671E+04	-1.785008E+04	**-1.278792E+04**
	Alarm	-1.123319E+04	-1.111250E+04	-1.096420E+04	-1.285909E+04	-1.073996E+04	-1.091783E+04	-1.082789E+04	-1.248335E+04	-1.238930E+04	-4.598914E+04	-1.512684E+04	-6.281176E+04	**-1.070551E+04**
	Child3	-4.198156E+04	-4.108299E+04	-3.970753E+04	-4.177459E+04	**-3.967251E+04**	-4.000545E+04	-3.976949E+04	-4.169372E+04	-4.597845E+04	-4.635923E+04	-4.632711E+04	-1.125884E+05	-3.968904E+04
	Alarm3	-3.898577E+04	-3.860234E+04	-3.846657E+04	-4.040642E+04	-3.851964E+04	-3.860444E+04	-3.857278E+04	-4.440421E+04	-4.491321E+04	-4.581132E+04	-5.185772E+04	-6.006728E+04	**-3.837307E+04**
	Insurance5	-8.465015E+04	**-8.323304E+04**	-8.334992E+04	-9.170055E+04	-8.553144E+04	-8.540811E+04	-8.584046E+04	-9.124499E+04	-9.719328E+04	-1.011746E+05	-1.059536E+05	-1.697343E+06	-8.484530E+04
	Alarm5	-6.564761E+04	-6.501115E+04	-6.511653E+04	-6.893632E+04	-6.513066E+04	-6.703756E+04	-6.516028E+04	-7.940382E+04	-7.594617E+04	-7.614681E+04	-9.049469E+04	-1.643380E+06	**-6.498929E+04**
	Insurance10	-1.757178E+05	-	-1.699400E+05	-1.837972E+05	-1.702013E+05	-	-	-2.011592E+05	-1.942892E+05	-1.950971E+05	-1.972078E+05	-1.547118E+07	**-1.698724E+05**
	Alarm10	-1.330759E+05	-	-1.290558E+05	-1.368523E+05	-1.290445E+05	-	-	-1.628922E+05	-1.532749E+05	-1.569444E+05	-1.809160E+05	-1.747659E+05	**-1.286983E+05**
	Pigs	-3.787895E+05	-	-3.490657E+05	-3.925541E+05	-3.487431E+05	-	-	-	-4.359147E+05	-4.541037E+05	-3.649772E+05	-4.084042E+05	**-3.487228E+05**
	Gene	-	-	-	-5.107311E+05	-4.638813E+05	-	-	-4.700560E+05	-5.844175E+05	-6.311532E+05	-5.971217E+05	-6.821043E+05	**-4.635984E+05**
	Munin	-	-	-	-3.070276E+05	-4.323130E+05	-	-	-	-	-	-	-	**-2.782613E+05**

TABLE 9
Results on the real Sachs dataset. (↓ means that the lower, the better while ↑ denotes the higher, the better.)

Method	Ar_F1 (↑)	Ar_Precision (↑)	Ar_Recall (↑)	SHD (↓)	Reverse (↓)	Miss (↓)	Extra (↓)
K2	24.24%	25.00%	23.53%	19	6	7	6
OBS	27.59%	33.33%	23.53%	15	6	7	2
Improved-K2	40.00%	46.15%	35.29%	15	**3**	8	4
GSBN	35.71%	45.45%	29.41%	14	4	8	2
MMHC	43.75%	46.67%	41.18%	14	4	**6**	4
SLL+C	41.38%	50.00%	35.29%	14	**3**	8	3
SLL+G	35.71%	45.45%	29.41%	14	4	8	2
GGSL	28.57%	36.36%	23.53%	16	4	9	3
NOTEARS	21.43%	27.27%	17.65%	18	4	10	4
DAG-GNN	16.00%	25.00%	11.76%	18	**3**	12	3
GOLEM	21.43%	27.27%	17.65%	18	4	10	4
DAG-NoCurl	18.18%	18.75%	17.65%	23	4	10	9
ADL	**55.17%**	**66.67%**	**47.06%**	**10**	**3**	**6**	**1**

which seriously deteriorates DAG learning quality. To alleviate this problem, we propose a new local-to-global DAG structure learning algorithm, ADL. ADL can adaptively use the AND-rule and the OR-rule to construct the skeleton of a DAG for accurate DAG learning. Experiments have shown that the proposed ADL algorithm outperforms five existing local-to-global DAG learning algorithms and seven global DAG learning algorithms in terms of the quality and efficiency of DAG learning. In addition, the idea of adaptive skeleton learning of ADL can be embedded in other local-to-global DAG learning algorithms. Therefore, future work may consider expanding the adaptive skeleton learning strategy to a framework that can effectively improve the performance of existing local-to-global DAG learning algorithms.

ACKNOWLEDGMENTS

This work was supported in part by the National Key Research and Development Program of China (under grant 2021ZD0111801), in part by the Australian Research Council (under grants DP200101210, DP230101122).

REFERENCES

[1] Y. Zeng, S. Shimizu, R. Cai, F. Xie, M. Yamamoto, and Z. Hao, "Causal discovery with multi-domain LiNGAM for latent factors," in *Proceedings of International Joint Conference on Artificial Intelligence*, 2021, pp. 2097–2103.

[2] R. Cai, J. Qiao, K. Zhang, Z. Zhang, and Z. Hao, "Causal discovery with cascade nonlinear additive noise model," in *Proceedings of International Joint Conference on Artificial Intelligence*, 2019, pp. 1609–1615.

[3] C. Zhang, H. Zhang, W. Xie, N. Liu, K. Wu, and L. Chen, "Where to: Crowd-aided path selection by selective Bayesian network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 1072–1087, 2021.

[4] K. Zhang, M. Gong, P. Stojanov, B. Huang, Q. Liu, and C. Glymour, "Domain adaptation as a problem of inference on graphical models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4965–4976, 2020.

[5] P. Cui, Z. Shen, S. Li, L. Yao, Y. Li, Z. Chu, and J. Gao, "Causal inference meets machine learning," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3527–3528.

[6] M. Prosperi, Y. Guo, M. Sperrin, J. S. Koopman, J. S. Min, X. He, S. Rich, M. Wang, I. E. Buchan, and J. Bian, "Causal inference and counterfactual prediction in machine learning for actionable healthcare," *Nature Machine Intelligence*, vol. 2, no. 7, pp. 369–375, 2020.

[7] J. Pearl, *Probabilistic reasoning in intelligent systems*. Elsevier, 2014, vol. 88, no. 3.

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

[9] B. Huang, K. Zhang, J. Zhang, J. D. Ramsey, R. Sanchez-Romero, C. Glymour, and B. Schölkopf, "Causal discovery from heterogeneous/nonstationary data." *Journal of Machine Learning Research*, vol. 21, no. 89, pp. 1–53, 2020.

[10] D. Colombo, M. H. Maathuis *et al.*, "Order-independent constraint-based causal structure learning." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3741–3782, 2014.

[11] S. M. Lee and S. B. Kim, "Parallel simulated annealing with a greedy algorithm for Bayesian network structure learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1157–1166, 2020.

[12] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "Dags with no tears: Continuous optimization for structure learning," *Advances in Neural Information Processing Systems*, vol. 31, pp. 9472–9483, 2018.

[13] Y. Yu, J. Chen, T. Gao, and M. Yu, "DAG-GNN: DAG structure learning with graph neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7154–7163.

[14] T. Gao, K. Fadnis, and M. Campbell, "Local-to-global Bayesian network structure learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1193–1202.

[15] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," *Machine Learning*, vol. 65, no. 1, pp. 31–78, 2006.

[16] T. Niinimaki and P. Parviainen, "Local structure discovery in Bayesian networks," in *Proceedings of Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2012, pp. 634–643.

[17] M. J. Vowels, N. C. Camgoz, and R. Bowden, "D'ya like DAGs? A survey on structure learning and causal discovery," *ACM Computing Surveys (CSUR)*, 2021.

[18] K. Zhang, B. Schölkopf, P. Spirtes, and C. Glymour, "Learning causality and causality-related learning: some recent progress," *National Science Review*, vol. 5, no. 1, pp. 26–29, 2018.

[19] M. Teyssier and D. Koller, "Ordering-based search: a simple and effective algorithm for learning Bayesian networks," in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 2005, pp. 584–590.

[20] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Machine Learning*, vol. 20, no. 3, pp. 197–243, 1995.

[21] D. M. Chickering, "Optimal structure identification with greedy search," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 507–554, 2002.

[22] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, no. 4, pp. 309–347, 1992.

[23] S. Behjati and H. Beigy, "An order-based algorithm for learning structure of Bayesian networks," in *Proceedings of International Conference on Probabilistic Graphical Models*. PMLR, 2018, pp. 25–36.

[24] ——, "Improved K2 algorithm for Bayesian network structure learning," *Engineering Applications of Artificial Intelligence*, vol. 91, p. 103617, 2020.

[25] P. Spirtes and C. Glymour, "An algorithm for fast recovery of sparse causal graphs," *Social Science Computer Review*, vol. 9, no. 1, pp. 62–72, 1991.

[26] P. Spirtes, C. Meek, and T. Richardson, "Causal inference in the presence of latent variables and selection bias," in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 1995, pp. 499–506.

[27] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen, "D-vae: A variational autoencoder for directed acyclic graphs," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[28] I. Ng, A. Ghassami, and K. Zhang, "On the role of sparsity and DAG constraints for learning linear DAGs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 943–17 954, 2020.

[29] Y. Yu, T. Gao, N. Yin, and Q. Ji, "DAGs with no curl: An efficient DAG structure learning approach," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 156–12 166.

[30] K. Yu, X. Guo, L. Liu, J. Li, H. Wang, Z. Ling, and X. Wu, "Causality-based feature selection: Methods and evaluations," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–36, 2020.

[31] X. Guo, K. Yu, L. Liu, F. Cao, and J. Li, "Causal feature selection with dual correction," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[32] X. Guo, K. Yu, F. Cao, P. Li, and H. Wang, "Error-aware Markov blanket learning for causal feature selection," *Information Sciences*, vol. 589, pp. 849–877, 2022.

[33] J. A. Gámez, J. L. Mateo, and J. M. Puerta, "Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood," *Data Mining and Knowledge Discovery*, vol. 22, no. 1, pp. 106–148, 2011.

[34] D. Margaritis and S. Thrun, "Bayesian network induction via local neighborhoods," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

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

[36] I. Tsamardinos, C. F. Aliferis, and A. Statnikov, "Time and sample efficient discovery of Markov blankets and direct causal relations," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 673–678.

[37] C. F. Aliferis, I. Tsamardinos, and A. Statnikov, "HITON: a novel Markov Blanket algorithm for optimal variable selection," in *AMIA Annual Symposium Proceedings*, vol. 2003. American Medical Informatics Association, 2003, p. 21.

[38] X. Wu, B. Jiang, K. Yu, H. Chen *et al.*, "Accurate Markov boundary discovery for causal feature selection," *IEEE Transactions on Cybernetics*, vol. 50, no. 12, pp. 4983–4996, 2019.

[39] K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan, "Causal protein-signaling networks derived from multiparameter single-cell data," *Science*, vol. 308, no. 5721, pp. 523–529, 2005.

[40] J. Suzuki, "Learning Bayesian belief networks based on the minimum description length principle: basic properties," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 82, no. 10, pp. 2237–2245, 1999.

[41] X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye, "Statistical ranking and combinatorial Hodge theory," *Mathematical Programming*, vol. 127, no. 1, pp. 203–244, 2011.

[42] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

Xianjie Guo received the B.S. degree from Anhui Normal University, Wuhu, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China.

His current research interests include causal discovery and federated learning.

Kui Yu (Member, IEEE) received the Ph.D. degree in computer science from the Hefei University of Technology, Hefei, China, in 2013.

From 2015 to 2018, he was a Research Fellow of computer science with the University of South Australia, Adelaide, SA, Australia. From 2013 to 2015, he was a Post-Doctoral Fellow with the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada. He is currently a Professor with the School of Computer Science and Information Engineering, Hefei University of Technology. His main research interests include causal discovery and machine learning.

Lin Liu received the B.S. and M.S. degrees in electronic engineering from Xidian University, Xi'an, China, in 1991 and 1994, respectively, and the Ph.D. degree in computer systems engineering from the University of South Australia, Adelaide, SA, Australia.

She is currently a Professor with the University of South Australia, Adelaide, SA, Australia. Her research interests include data mining, machine learning, causal inference, and bioinformatics.

Peipei Li received the B.S., M.S., and Ph.D. degrees from the Hefei University of Technology, in 2005, 2008, and 2013, respectively. She was a Research Fellow at the Singapore Management University, from 2008 to 2009. She was a student intern at Microsoft Research Asia from August 2011 to December 2012. She is currently an Associate Professor with the Hefei University of Technology, China. Her research interests are in data mining and knowledge engineering.

Jiuyong Li (Member, IEEE) received the Ph.D. degree in computer science from Griffith University, Brisbane, QLD, Australia, in 2002.

He is currently a Professor with the University of South Australia, Adelaide, Australia. His main research interests include data mining, causal discovery and inference, and bioinformatics. His research work has been supported by eight Australian Research Council Discovery projects and many industry and government projects.