Bootstrap-Based Layerwise Refining for Causal Structure Learning

Guodu Xiang, Hao Wang, Kui Yu, *Member, IEEE*, Xianjie Guo, Fuyuan Cao, and Yukun Song

***Abstract*—Learning causal structures from observational data is critical for causal discovery and many machine learning tasks. Traditional constraint-based methods first adopt conditional independence (CI) tests to learn a global skeleton layer by layer and then orient the undirected edges to obtain a causal structure. However, the reliability of these statistical tests largely depends on the quality of data samples. In real-life scenarios, the presence of data noise or limited samples often makes many CI tests unreliable at each layer in the skeleton learning phase, leading to an inaccurate skeleton. As the number of layers increases, the inaccurate skeleton will continue to impair the skeleton construction of subsequent layers. Furthermore, an unreliable skeleton hampers the skeleton orientation procedure, resulting in an unsatisfactory causal structure. In this article, we propose a Bootstrap-based layerwise refining (BLR) algorithm for causal structure learning, which includes two new procedures to solve the above problems. First, BLR utilizes a novel layerwise skeleton refining procedure to construct the global skeleton layer by layer based on the bootstrap sampling. Second, BLR employs a collective skeleton orientation procedure that incorporates scoring techniques to collectively orient the global skeleton. The experimental results show that BLR outperforms the state-of-the-art methods on the benchmark Bayesian Network datasets.**

***Impact Statement*—Discovering causal relationships between various phenomena helps us understand how the world works and how events are generated. However, correctly identifying causal relationships between variables remains a significant challenge. Our proposed method for learning causal structures provides a new way to solve this problem based on ensemble learning. We use a layerwise refining strategy to obtain a reliable skeleton and then orient it using scoring techniques. Our algorithm outperforms current state-of-the-art methods in accuracy and stability on multiple datasets. This approach is expected to play an important role in causal inference in social sciences, biomedicine, natural language processing, and more.**

Manuscript received 10 May 2023; revised 26 September 2023; accepted 28 October 2023. Date of publication 6 November 2023; date of current version 21 June 2024. This work was supported by the National Key Research and Development Program of China under Grant 2021ZD0111801 and in part by the National Natural Science Foundation of China under Grants 62376087 and 62176082. This article was recommended for publication by Associate Editor Aaron Baughman upon evaluation of the reviewers' comments. *(Corresponding author: Kui Yu.)*

Guodu Xiang, Hao Wang, Kui Yu, Xianjie Guo, and Yukun Song are with the Key Laboratory of Knowledge Engineering with the Big Data of Ministry of Education, Hefei University of Technology, Hefei 230601, China, and also with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China (e-mail: xgd600600@mail.hfut.edu.cn; jsjxwangh@hfut.edu.cn; yukui@hfut.edu.cn; xianjieguo@mail.hfut.edu.cn; Yukun_Song@mail.hfut.edu.cn).

Fuyuan Cao is with the School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China (e-mail: cfy@sxu.edu.cn).

Digital Object Identifier 10.1109/TAI.2023.3329786

***Index Terms*—Bootstrap sampling, causal structure learning, directed acyclic graph, layerwise refining.**

I. Introduction

CAUSAL structure learning aims to discover a reliable directed acyclic graph (DAG) from observational data. A DAG can well describe the causal relationships between variables since a directed edge $X_1 \rightarrow X_2$ in a DAG implies that X_1 is the cause of X_2 and X_2 is the effect of X_1 [1], [2], [3]. Causal structures have been widely used in various fields, such as biology [4], medical imaging [5], [6], machine learning [7], [8], [9], [10], and many others [11], [12], [13], [14]. Although numerous causal structure learning methods have been proposed in the past few decades, constraint-based methods have an important role in many scenarios, such as learning high-dimensional sparse graphs [15].

Existing constraint-based methods (such as the PC algorithm [16]) determine whether there exists an edge between two variables in a graph by performing conditional independence (CI) tests. Specifically, constraint-based methods mainly consists of three steps: Step 1 learns a graph skeleton using CI tests; Step 2 identifies all possible v-structures in the skeleton; Step 3 orients the remaining undirected edges as many as possible using the Meek's rules [17]. When conducting CI tests, if two variables are conditionally independent given a set of variables, we call this set the separation set. The v-structure is a special structure composed of three nodes and two directed edges (such as $X_i \rightarrow W \leftarrow X_j$ defined in Definition III-2), which can be identified first based on the learned skeleton and the separation set. Previous studies [18], [19] have demonstrated that constraint-based methods can correctly recover edges in graphs when the sample size tends to infinity. However, in real-world applications, data are often limited or contains noise, which may make existing constraint-based methods learn an incorrect causal structure. This can be illustrated in the following two aspects.

First, in the skeleton learning phase, small samples or data noise may produce wrong CI results leading to unreliable skeletons when constructing skeletons layer by layer with the size of the separation set. Inaccurate CI tests in each layer can falsely delete edges and cause cascading errors in subsequent skeleton construction. For example, as shown in Fig. 1(a), the accurate conditional independence relationships between variables in the true DAG with: $X_2 \perp X_5$, $X_2 \perp X_4 | \{X_3\}$, and $X_1 \perp X_3 | \{X_2, X_4, X_5\}$. The process of deleting edges layer by layer

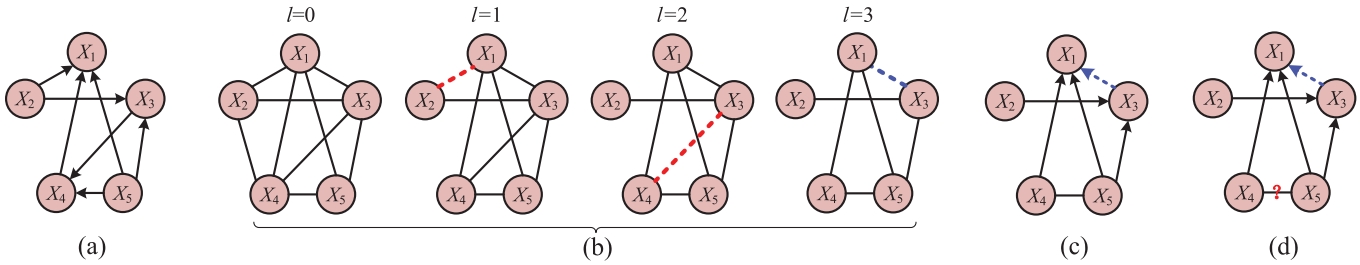

Fig. 1. Example of the inaccurate causal structure learned by the constraint-based methods due to the wrong CI results. (a) True DAG. (b) Constructing a skeleton layer by layer (red dashed edge is mistakenly deleted, and blue dashed edge is mistakenly kept). (c) Identifying all v-structures. (d) Orienting the remaining undirected edges with the Meek's rules.

according to the separation set size is shown in Fig. 1(b), where l represents the layer size, such as $l=0$ denoting the first layer and the size of separation set is 0. If we assume that the wrong CI results $X_1 \perp X_2 | \{X_3\}$ at $l=1$ and $X_3 \perp X_4 | \{X_1, X_5\}$ at $l=2$ are produced (i.e., X_1 and X_2, X_3 and X_4, are both wrongly considered as conditional independence), then edges $X_1 - X_2$ and $X_3 - X_4$ are mistakenly deleted from the skeleton respectively. Then at $l=3$, the adjacent variable set of X_1 except X_3 is $\{X_4, X_5\}$ and the adjacent variable set of X_3 except X_1 is $\{X_2, X_5\}$ in the skeleton. However, the separation set of X_1 and X_3 in the true DAG is $\{X_2, X_4, X_5\}$. Thus, conditioning on the set $\{X_4, X_5\}$ or the set $\{X_2, X_5\}$ does not make X_1 and X_3 conditionally independent. As a result, the edge $X_1 - X_3$ is falsely kept in the skeleton. Therefore, the wrong CI results of the previous layers have an impact on the construction of the subsequent layers' skeleton as the number of layers increases, leading to cascading errors. Finally, in the skeleton construction phase, in the example of Fig. 1(b), two edges are mistakenly deleted and an edge is wrongly retained in the final skeleton, resulting in an inaccurate skeleton.

Second, in the skeleton orienting phase, an inaccurate skeleton hinders the identification of the v-structures and further hampers the orientation of the remaining undirected edges. Fig. 1(c) illustrates that the v-structure $X_3 \rightarrow X_4 \leftarrow X_5$ in the true DAG cannot be identified in an inaccurate skeleton due to falsely deleting the edge $X_3 - X_4$ at $l=2$. Then, as shown in Fig. 1(d), since the v-structure $X_3 \rightarrow X_4 \leftarrow X_5$ is not identified, when applying Meek's rules to orient the remaining edges, the edge direction of $X_4 - X_5$ cannot be determined.

Based on the above discussions, constraint-based methods face the issues of inaccurate skeleton construction and incorrect skeleton orientation due to unreliable CI tests. To tackle these issues, our contributions can be summarized as follows:

1) We propose a Bootstrap-based layerwise refining (BLR) algorithm for causal structure learning, which includes the layerwise skeleton refining (LSkeR) and collective skeleton orientation (CSkeO) procedures to tackle the problem of unreliable CI tests.
2) The LSkeR procedure presents a novel layerwise skeleton refining strategy by utilizing bootstrap sampling. LSkeR learns and refines the skeletons on sampled datasets at each layer for achieving a much more accurate skeleton at the current layer and reducing the impact of unreliable CI tests on skeleton learning at subsequent layers.
3) The CSkeO procedure proposes an effective collective skeleton orientation strategy by aggregating all of the learned DAGs using a scoring method on sampled datasets for improving the accuracy of edge orientations.
4) We conduct extensive experiments to evaluate the effectiveness of BLR on eight benchmark Bayesian network (BN) datasets, and the experimental results show that our algorithm outperforms other causal structure learning methods.

The rest of this article is organized as follows: Section II reviews related work of causal structure learning. Section III introduces basic notations and definitions. Section IV proposes our method. Section V presents and analyzes experimental results and Section VI concludes and discusses our work.

II. Related Work

In this section, we briefly introduce the literature related to causal structure learning methods. The existing causal structure learning methods can be roughly divided into two types: combinatorial optimization methods and continuous optimization methods.

Existing combinatorial optimization methods can be subdivided into constraint-based, score-based, and hybrid methods. Constraint-based methods rely on CI tests to discover the causal relationships between variables. The PC algorithm [16] is a classical constraint-based algorithm. And there are many derivatives of the PC algorithm, such as PC-stable [20], Consistent-PC [21], and others [22], [23], [24]. These algorithms are inspired and improved by the PC algorithm to improve the robustness of v-structure recognition. For example, PC-stable eliminates the order dependency issues and Consistent-PC solves the problem of separation set inconsistency. These algorithms learn causal structures from the datasets without latent variables (satisfying the causal sufficiency assumption, as shown in Definition III-6). Moreover, there are other constraint-based methods, such as FCI [1], RFCI [15], and FCI-soft [25], which could learn causal structures from the datasets containing latent variables. Nevertheless, the prominent disadvantage of constraint-based methods is that the CI tests usually require a large amount of data samples, and the accuracy of the CI tests is easily hindered by noisy data.

Score-based methods learn causal structures by adopting a scoring function to evaluate how well each graph fits the data and a search strategy to find the best graph structures in the potential graph space. GES [26] is a representational score-based method, which adopts the BDeu [27] scoring function and the greedy search strategy. There are many other scoring functions and search strategies [28]. Different scoring functions can be combined with different search strategies to form different methods, such as GIES [29] and THPs [30]. However, score-based methods generally suffer from a large search space, and the search space grows exponentially as the number of nodes increases, leading to practical inefficiencies.

Hybrid methods combine the constraint-based method and score-based method to discover DAGs efficiently. Specifically, the hybrid method first uses a constraint-based method to restrict the search space of the graph and then adopts a score-based method to find the DAG with the highest score. MMHC [31] is a classic hybrid method, which first constructs a local skeleton using CI tests, and then adopts search-and-score techniques for orientation to discover causal structures. BCSL [32] eliminates asymmetric edges in the learned local structure to improve the accuracy of causal structure.

Continuous optimization methods are proposed for causal structure learning in recent years. These methods usually solve problems by using gradient descent methods to optimize various highly parameterized networks [33]. NOTEARS [34] is the first method to formulate the causal structure learning problem as a continuous optimization problem over real matrices and solves the problem using augmented Lagrangian methods. However, this method is designed for the linear Structural Equation Model (SEM), which means that the relationships between variables are linear. Subsequently, many researchers have used neural networks to study the nonlinear relationships between variables [35], [36], [37], [38], [39]. Among them, DAG-GNN [35] employs a variational auto-encoder to reconstruct the data for fitting the generative mechanism and derive the true graph structure. DAG-NoCurl [39] learns causal structures by first finding the initial cyclic solution of the optimization problem and then using the Hodge decomposition of the graph and projecting the cyclic graph onto the gradient of the potential function. Continuous optimization methods have a strong dependence on the generative model of the data and the distribution of noise. Therefore, the application scenarios of the continuous optimization methods are different, the performance in practice is also unstable, and it is difficult to be widely used in real scenarios.

To compare the differences of the above algorithms more intuitively, we summarize their characteristics in Table I. "Sufficiency" refers to whether an algorithm satisfies the causal sufficiency assumption. These algorithms may produce different results, where DAG is a directed acyclic graph, PAG is a partially ancestral graph (a Markov equivalence class of DAGs with latent and selection variables in the acyclic case) and CPDAG is a completely partial directed acyclic graph (in the graph all v-structures have been identified and other edges oriented using the Meek's rules [17]).

TABLE I
COMPARISON OF THE RELATED ALGORITHMS

Algorithm	Year	Type	Sufficiency	Output
PC [16]	1989	Constraint	Yes	CPDAG
PC-stable [20]	2014	Constraint	Yes	CPDAG
Consistent-PC [21]	2019	Constraint	Yes	CPDAG
FCI [1]	2000	Constraint	No	PDAG
RFCI [15]	2012	Constraint	No	PDAG
GES [26]	2002	Score	Yes	CPDAG
GIES [29]	2012	Score	Yes	CPDAG
MMHC [31]	2006	Hybrid	Yes	DAG
BCSL [32]	2022	Hybrid	Yes	DAG
NOTEARS [34]	2018	Continuous optimization	Yes	DAG
DAG-GNN [35]	2019	Continuous optimization	Yes	DAG
DAG-NoCurl [39]	2021	Continuous optimization	Yes	DAG

TABLE II
SUMMARY OF NOTATIONS

Notation	Meaning
$\boldsymbol{X}$	A set of variables
$P(\boldsymbol{X})$	The joint probability distribution of $\boldsymbol{X}$
S	A skeleton (a undirected graph)
G	A directed acyclic graph
$\boldsymbol{S}$	A set of condition variables
$\boldsymbol{D_s}$	A set of sampled datasets
A, B	The adjacency matrix representing a graph
A_{ij}, B_{ij}	The element at row i and column j of the adjacency matrix
X_i, X_j, W	Random variables (nodes)
$\boldsymbol{Pa}(G, X_i)$	The parent node set of X_i in G
$\boldsymbol{adj}(S, X_i)$	The adjacency variable set of X_i in S
$\boldsymbol{adj}(S, X_i)\backslash\{X_j\}$	The adjacency variable set of X_i in S except X_j
$\boldsymbol{X}\backslash\{X_i, X_j\}$,	A set of variables $\boldsymbol{X}$ except X_i and X_j
$X_i \perp X_j \mid \boldsymbol{S}$	X_i and X_j are independent under the set $\boldsymbol{S}$
$X_i \not\perp X_j \mid \boldsymbol{S}$	X_i and X_j are dependent under the set $\boldsymbol{S}$
N	The number of sampled datasets
M	The number of experiments
n	The number of samples in the dataset
m	The number of variables in the dataset
d	The maximum degree in the graph
ε	The aggregation threshold
α	The significance level for CI test
$\lvert\cdot\rvert$	The size of the condition set

III. BACKGROUND

In this section, we introduce some basic notations and definitions. In Table II, we summarize the notations commonly used in this article.

Definition III-1: (Conditional Independence [40]). Random variables X_i and X_j are conditionally independent if given a variable set $\boldsymbol{S}$ that $P(X_i, X_j \mid \boldsymbol{S}) = P(X_i \mid \boldsymbol{S})\, P(X_j \mid \boldsymbol{S})$, which is denoted as $X_i \perp X_j \mid \boldsymbol{S}$. Otherwise, X_i and X_j are conditionally dependent, which is denoted as $X_i \not\perp X_j \mid \boldsymbol{S}$.

Definition III-2: (V-structure [1]). A structure containing three nodes X_i, X_j, W is called a v-structure, where X_i and X_j have a common child node W and there is no edge between X_i and X_j, i.e., $X_i \rightarrow W \leftarrow X_j$.

Definition III-3: (Blocked Path [41]). A path between X_i and X_j in a DAG is blocked by $\boldsymbol{S}$ needs to satisfy any of the following conditions:

1) if the path contains a chain structure $X_i \rightarrow W \rightarrow X_j$ or a fork structure $X_i \leftarrow W \rightarrow X_j$, W should be included in $\boldsymbol{S}$;

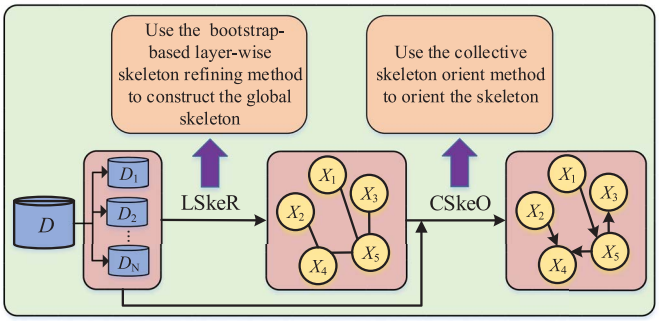

Fig. 2. Framework of the BLR algorithm.

2) if the path contains a v-structure $X_i \to W \leftarrow X_j$, neither W nor W's descendant nodes should be included in $\boldsymbol{S}$.

Definition III-4: (D-Separation [41]). X_i and X_j are d-separated by a set $\boldsymbol{S}$ if all paths between X_i and X_j in the DAG are blocked by the set $\boldsymbol{S}$.

Definition III-5: (Separation Set [41]). A set $\boldsymbol{S}$ is called the separation set of X_i and X_j, if $\exists \boldsymbol{S} \subseteq \boldsymbol{X} \setminus \{X_i, X_j\}$ d-separates X_i and X_j, i.e., X_i and X_j are conditionally dependent conditioning on the set $\boldsymbol{S}$.

Definition III-6: (Causal Sufficiency [1]). A variable set $\boldsymbol{X}$ satisfies the causal sufficiency assumption if the direct cause variables of any two variables of $\boldsymbol{X}$ exist in $\boldsymbol{X}$. That is, there are no latent variables in the variable set $\boldsymbol{X}$.

Definition III-7: (Faithfulness [1]). Given a DAG G, for any two nodes X_i and X_j in $\boldsymbol{X}$ and a set $\boldsymbol{S} \subseteq \boldsymbol{X} \setminus \{X_i, X_j\}$, X_i and X_j are conditionally independent under the set $\boldsymbol{S}$ in the data distribution correspond to X_i and X_j being d-separated by the set $\boldsymbol{S}$ in G, then the joint probability distribution $P(\boldsymbol{X})$ is said to be faithful to G while G is faithful to $P(\boldsymbol{X})$.

The faithfulness makes the conditional independence in data distributions equivalent to the d-separation in the graphical models. Thus, the faithfulness makes it possible to learn causal structures from a given dataset.

IV. Proposed Algorithm

A. Overview

In this section, we propose a BLR algorithm for accurately recovering the underlying DAG from observational data. The BLR algorithm is divided into two main procedures: LSkeR and CSkeO. The LSkeR procedure learns a skeleton using a layerwise skeleton refining method based on the bootstrap sampling technique. The CSkeO procedure collectively orients the skeleton learned at the LSkeR procedure from the sampled datasets using the scoring function methods. The framework of the BLR algorithm is shown in Fig. 2, and the details of the above two procedures of the BLR algorithm are described in Sections IV-B and IV-C.

B. The Layerwise Skeleton Refining (LSkeR) Procedure

The goal of the LSkeR procedure is to construct a reliable global skeleton from the dataset D. As shown in Fig. 3, LSkeR

Algorithm 1: The LSkeR procedure

Input: Dataset D with the variable set $\boldsymbol{X}$, significance level α, number of sampled datasets N, and aggregating threshold ε
Output: Global skeleton S^*, a set of sampled datasets $\boldsymbol{D_s}$

1: // Stage 1: Data sampling
2: **for** $k = 1 : N$ **do**
3: Bootstrap sampling dataset D_k from D
4: **end for**
5: $\boldsymbol{D_s} = \{D_1, D_2, ..., D_N\}$
6: // Stage 2, Stage 3 and Stage 4
7: $l = 0$, $S^{-1} = S^c$
8: **while** all pairs of adjacent vertices (X_i, X_j) in S^{l-1} satisfy $\left|\boldsymbol{adj}\left(S^{l-1}, X_i\right) \setminus \{X_j\}\right| \geq l$ **do**
9: // Stage 2: Learning skeletons at the l-th layer
10: **for** $k = 1 : N$ **do**
11: $S_k^l = S^{l-1}$
12: Remove all edges $X_i - X_j$ from S_k^l if $\exists \boldsymbol{S} \subseteq \boldsymbol{adj}\left(S^l, X_i\right) \setminus \{X_j\}$ with $|\boldsymbol{S}| = l$ such that $X_i \perp X_j | \boldsymbol{S}$ using D_k under the α
13: **end for**
14: // Stage 3: Refining the skeleton at the l-th layer
15: Sum the adjacency matrices A_1^l, A_2^l,..., A_N^l corresponding to S_1^l, S_2^l, ..., S_N^l respectively, and let $\tilde{A}^l = \sum_{i=1}^N A_i^l$
16: Judge every element $\tilde{A}_{ij}^l$ in $\tilde{A}^l$ based on the Eq. (2) to obtain S^l represented by A^l
17: // Stage 4: Go to the (l+1)-th layer
18: $l = l + 1$
19: **end while**
20: $S^* = S^{l-1}$
21: **return** S^*, $\boldsymbol{D_s}$

first samples N datasets from the original dataset using the bootstrap sampling, then it learns skeletons from the N sampled datasets using a layerwise refining strategy. The idea of the layerwise skeleton refining strategy is described as follows. Starting from the fully connected skeleton, using a layer-by-layer strategy, LSkeR learns N skeletons from each sampled dataset independently and refines the learned skeleton by aggregating these N skeletons at each layer. The initial value of the layer begins with 0 and it increases with 1 each time. At the lth ($l \geq 1$) layer, LSkeR takes the skeleton aggregated at the $(l-1)$th layer as the input skeleton and learns new skeletons on each sampled dataset independently based on this input skeleton. This procedure is repeated until the size of the adjacency variables of all variables in the aggregated skeleton is less than the value of the current layer. The pseudocode of the LSkeR procedure is shown in Algorithm 1.

Stage 1: Data sampling (Lines 1–5). LSkeR samples N datasets $D_1, D_2, ..., D_N$ that have the same size as the original dataset D. The set $\boldsymbol{D_s}$ saves the N sampled datasets for orienting skeletons in the CSkeO procedure.

In a dataset containing n samples, using the bootstrap sampling method to sample a dataset, when $n \to \infty$, approximately 36.8% samples of the original dataset do not appear in the sampled dataset [42]. Based on the sampling theory, each sampled

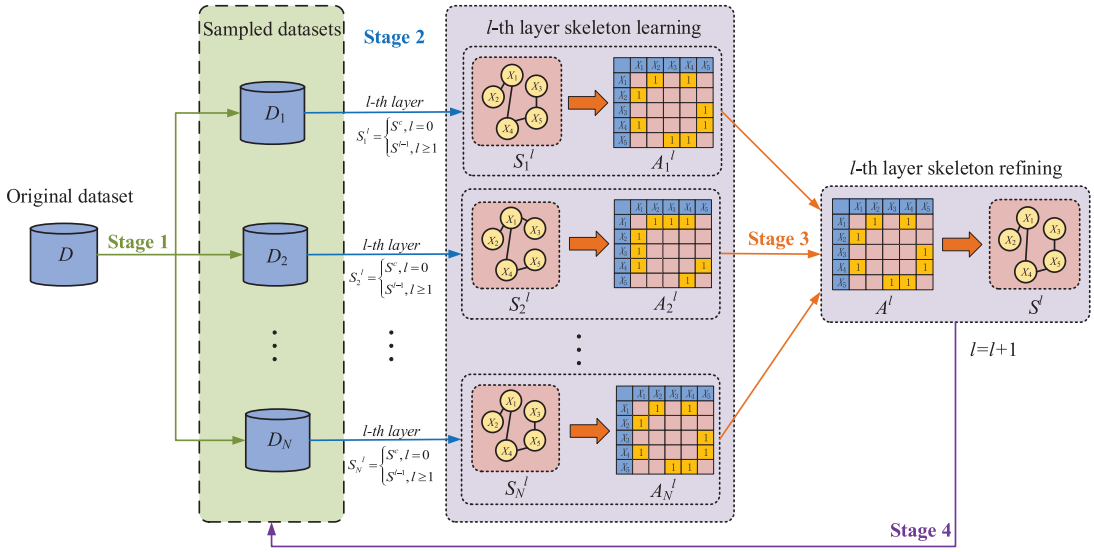

Fig. 3. LSkeR procedure of the BLR algorithm.

dataset contains at least 63.2% of the samples in the original dataset, and the other 36.8% of the samples are repeated when the original data samples tend to infinity. The bootstrap sampling method improves the diversity of data samples by sampling different datasets. The diversity of data ensures that we can mitigate the impact of unreliable CI tests on skeleton learning by aggregating the skeletons learned from the N sampled datasets.

Stage 2: Learning skeletons at the lth layer (Lines 9–13). At the lth layer, taking the skeleton S^{l-1} learned from the (l–1)th layer as the initial skeleton (when $l=0$, taking the fully connected skeleton S^c as the initial skeleton), N sampled datasets $D_1, D_2, ..., D_N$ are used to learn N skeletons $S_1^l, S_2^l, \ldots, S_N^l$ independently. The process of learning the skeleton S_k^l on the kth $(1 \le k \le N)$ sampled dataset D_k at the lth layer is described as follows.

For each node X_i in the skeleton S^{l-1}, we let $\boldsymbol{adj}(S^{l-1}, X_i)$ denote the adjacency node set of X_i in the skeleton S^{l-1}. Stage 2 considers each node in the adjacency node set of X_i i.e., $\forall X_j \in \boldsymbol{adj}\left(S^{l-1}, X_i\right)$ and computes whether X_i and X_j are independent using the dataset D_k. The edge $X_i - X_j$ in the skeleton S^{l-1} is removed if there exists a subset $\boldsymbol{S} \subseteq \boldsymbol{adj}\left(S^{l-1}, X_i\right) \setminus \{X_j\}$ with $|\boldsymbol{S}| = l$ such that $X_i \perp X_j | \boldsymbol{S}$ holds. Once the conditional independence of each node X_i in the skeleton S^{l-1} and its adjacent node X_j among all possible separation sets $\boldsymbol{S} \subseteq \boldsymbol{adj}\left(S^{l-1}, X_i\right) \setminus \{X_j\}$ with $|\boldsymbol{S}| = l$ are checked, and the skeleton S_k^l is obtained using the dataset D_k at the lth layer.

Stage 3: Refining skeleton at the lth layer (Lines 14–16). After finishing the skeleton learning at the lth layer, the N skeletons $S_1^l, S_2^l, \ldots, S_N^l$ are obtained in Stage 2 and Stage 3 aggregates these skeletons to get a global skeleton S^l for the next-layer skeleton learning. The LSkeR procedure sets a threshold ε to determine whether there exists an edge between X_i and X_j in S^l based on the N skeletons $S_1^l, S_2^l, \ldots, S_N^l$. If more than ε of the skeletons contain an edge between X_i and X_j, then an edge exists in S^l between X_i and X_j. We set the value of the threshold ε to $N/2$. In Section V.C.3, we conduct a sensitivity analysis on the parameter ε to discuss why the value of ε is set to $N/2$.

Specifically, we refine the skeleton at the lth layer as follows. The adjacency matrices $A_1^l, A_2^l, \ldots, A_N^l$ represent the skeletons $S_1^l, S_2^l, \ldots, S_N^l$ learned from the N sampled datasets at the lth layer, respectively. $A_{k;ij}^l$ is the ith row and j column element of matrix A_k^l $(k = 1, 2, \ldots, N)$. $A_{k;ij}^l = 1$ means that there is an edge between i and j, otherwise there is no edge between i and j. $\tilde{A}^l$ is the sum of the N matrices as shown in (1). $\tilde{A}_{ij}^l$ means the total number of the edge between i and j existing in the skeletons $S_1^l, S_2^l, \ldots, S_N^l$.

$$\tilde{A}^l = \sum_{i=1}^{N} A_i^l. \tag{1}$$

The aggregated skeleton S^l is represented by the aggregated matrix A^l. $A_{ij}^l = 1$ means that there is an edge between i and j in S^l, otherwise there is no edge between i and j in S^l. Equation (2) shows that A^l is obtained from $\widetilde{A}^l$ based on the ε. If the number of the edge between i and j in $\widetilde{A}^l$ is larger than ε, then there is an edge between i and j in the aggregated skeleton. Otherwise, there is no edge between i and j in the aggregated skeleton. Thus, we get the aggregated skeleton S^l at the lth layer.

$$A_{ij}^l = \begin{cases} 1 & \widetilde{A}_{ij}^l > \varepsilon \\ 0 & \widetilde{A}_{ij}^l \le \varepsilon \end{cases}. \tag{2}$$

Stage 4: Go to the (l+1)th layer (Lines 17–18). The value of the layer increases by 1, starting from the lth layer to the (l+1)th layer.

Stage 2, Stage 3, and Stage 4 are repeated until the sizes of the adjacency variables of all variables in the skeleton S^{l-1} are less than l. At this time, the skeleton S^{l-1} is represented as S^*, and S^* is the final skeleton.

LSkeR improves the accuracy of skeleton learning in two ways. On the one hand, we leverage the bootstrap sampling method to obtain N diverse sampled datasets for the LSkeR

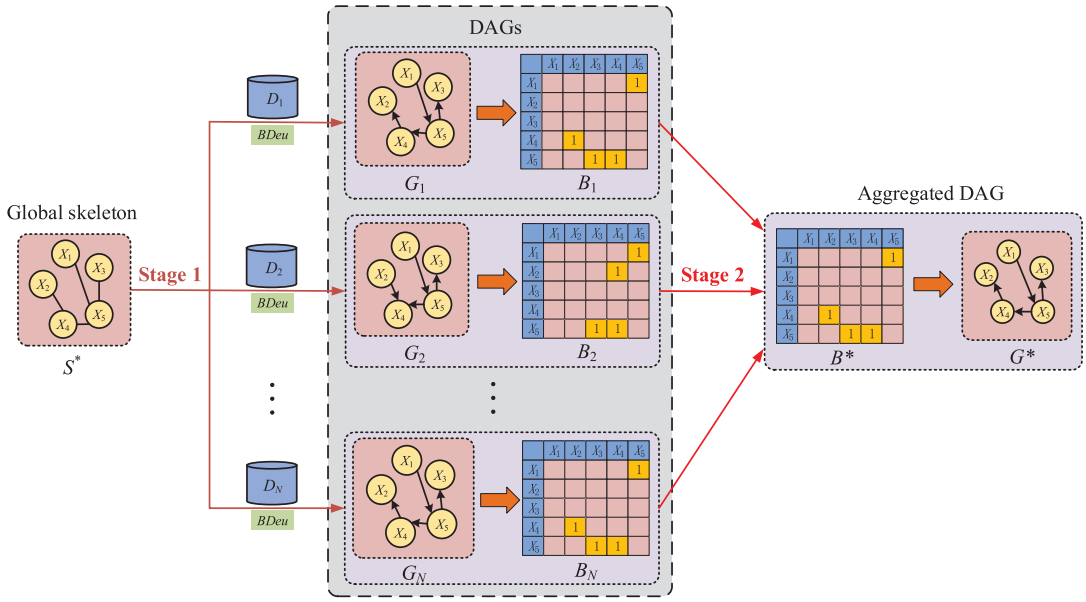

Fig. 4. CSkeO procedure of the BLR algorithm.

procedure. These sampled datasets come from the same original dataset. They have similar data distributions among them, and there are some distribution differences, which can be aggregated to learn the skeleton. By aggregating N skeletons, we can improve skeleton accuracy by overcoming the influence of incorrect CI tests on skeleton learning. On the other hand, at each layer of the skeleton learning, we use the skeleton aggregated from N sampled datasets in the previous layer as the prior skeleton for the current-layer skeleton learning. In this way, a high-quality skeleton is provided for LSkeR to construct a subsequent skeleton on each sampled dataset, then the skeletons learned at each layer are more reliable, reducing cascading errors in the skeleton learning phase and leading to an accurate final skeleton.

C. The Collective Skeleton Orientation (CSkeO) Procedure

The CSkeO procedure aims to orient the undirected edges in S^* learned from the LSkeR procedure to obtain a DAG. Fig. 4 shows that the BDeu scoring function is used to learn the DAGs with the highest scores on sampled datasets independently and all these learned DAGs are aggregated to get the final DAG. The pseudocode of the CSkeO procedure is shown in Algorithm 2.

Stage 1: Scoring the skeleton for orientation using sampled datasets independently (Lines 1–4). Based on the skeleton S^*, the CSkeO procedure adopts the BDeu scoring function on the N sampled datasets $D_1, D_2, ..., D_N$ sampled from the LSkeR procedure for orientation. The edges in the skeleton are continuously adjusted through the hill-climbing search strategy, until obtaining the N DAGs $G_1, G_2, ..., G_N$ with the highest BDeu scores.

The BDeu [27] scoring function of DAG G on the corresponding dataset D is defined as:

$$\mathrm{BDeu}(G,D)=\log P(G)+\sum_{i=1}^{m}\sum_{j=1}^{q_i}\left[\log\frac{\Gamma\left(\frac{N'}{q_i}\right)}{\Gamma\left(N_{ij}+\frac{N'}{q_i}\right)}+\sum_{k=1}^{r_i}\log\frac{\Gamma\left(N_{ijk}+\frac{N'}{r_iq_i}\right)}{\Gamma\left(\frac{N'}{r_iq_i}\right)}\right], \tag{3}$$

Algorithm 2: The CSkeO procedure

Input: Global skeleton S^*, Sampled datasets set $\boldsymbol{D_s}$, the number of sampled datasets N, and aggregating threshold ε
Output: DAG G^*

1: // Stage 1: Scoring the skeleton for orientation using the sampled datasets independently
2: **for** $k = 1 : N$ **do**
3: Obtain the G_k with the highest $BDeu(G_k, D_k)$ through the hill-climbing search process
4: **end for**
5: // Stage 2: Aggregating all directed edges in the learned DAGs for orientation
6: Sum the adjacency matrices B_1, B_2,..., B_N corresponding to G_1, G_2, ..., G_N respectively, and let $\widetilde{B} = \sum_{i=1}^{N} B_i$
7: **for** every element $\tilde{B}_{ij}$ in $\widetilde{B}$ **do**
8: **if** $\tilde{B}_{ij} > \varepsilon$ **then**
9: $B^*_{ij} = 1$
10: **else**
11: $B^*_{ij} = 0$
12: **end if**
13: **end for**
14: **return** G^* represented by B^*

where $P(G)$ is a prior probability of the specific graph structure and this probability can be ignored because it is usually assumed to be the same for all graphs. Γ is the Gamma function, i is the index of the m variables, q_i represents the combination of values of the node X_i's parents, j is the index of q_i, r_i represents the possible values of the node X_i, and k is the index of r_i. In addition, N_{ijk} represents the number of instances when the node X_i takes the kth value, and its parents take the jth combination of values in the dataset D, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ represents the total number of instances where the parents of X_i have the jth combination of values in the dataset D, and N' is

the equivalent sample size (ESS) that represents the confidence we have in the prior parameters.

Stage 2: Aggregating all directed edges in the learned DAGs for orientation (Lines 5–14). Stage 2 aggregates all directions of the edges in N DAGs $G_1, G_2, ..., G_N$ to obtain the final DAG G^*. The CSkeO procedure sets the same threshold ε to determine the edge direction of $X_i - X_j$ based on the N DAGs learned from Stage 1. If more than ε edges go from X_i to X_j in these DAGs, then the edge direction of $X_i - X_j$ goes from X_i to X_j in G^*, otherwise, the edge direction of $X_i - X_j$ is from X_j to X_i in G^*. Here, we set the value of the threshold ε to $N/2$ because the skeleton is oriented using the same sampled datasets from LSkeR.

Specifically, N learned DAGs $G_1, G_2, \ldots, G_N$ are oriented by the scoring function from N sampled datasets. N learned DAGs are represented by N adjacency matrices $B_1, B_2, \ldots, B_N$, respectively. $B_{k;ij} = 1(k = 1, 2, ..., N)$ means an directed edge from i to j in the DAG G_k. Otherwise, there is no edge from i to j. $\widetilde{B}$ is the sum of the N adjacency matrices. $\tilde{B}_{ij}$ means the total number of the directed edge from i to j in the DAGs $B_1, B_2, \ldots, B_N$.

If $\tilde{B}_{ij}$ is larger than ε, then $B^*_{ij} = 1$. $B^*_{ij} = 1$ means that there is a directed edge from i to j in the aggregated DAG. Otherwise, $B^*_{ij} = 0$, there is no edge from i to j in the aggregated DAG. Therefore, the aggregated matrix B^* of N adjacency matrices is obtained. Correspondingly, we get the DAG G^* represented by B^*.

There are two main benefits of using the scoring function orienting. First, using a scoring function to orient edges does not need to consider the separation sets (Definition III-5) that play a significant role in the skeleton orienting stage for the constraint-based methods. The separation sets of variables in each sampled dataset may be different during the skeleton learning phase, so using a scoring function to orient the skeleton avoids finding separation sets that are inconsistent with the final skeleton. Second, based on a reliable skeleton, it is more accurate and efficient to use a scoring function to orient the edges in the skeleton. The given skeleton reduces the search space of the scoring function, and scoring a reliable skeleton can improve the accuracy of orientation. The advantage of CSkeO is that avoids orientation bias caused by a low-quality dataset. Aggregating the DAGs learned from multiple sampled datasets can improve the robustness of skeleton orientation, making the aggregated DAG more accurate.

V. Experiments

In this section, we evaluate the effectiveness of our algorithm by comparing it with seven state-of-the-art causal structure learning algorithms on eight benchmark BN datasets. Moreover, we conduct multiple experiments and set different parameters to evaluate the stability of our algorithm based on the experimental results.

A. Experiment Settings

1) Datasets: We conduct experiments on eight benchmark BNs, which are provided from existing works [31]. These eight benchmark BNs are Child, Child3, Child5, Alarm3, Child10, Alarm10, Pigs, and Link. These eight benchmark BNs are derived from real-life structures, as described on the website[1]. The details of the eight BNs are presented in Table III. The number of nodes in these networks varies from 20 to 724, verifying the effectiveness of the algorithm under different network sizes. For each network, we randomly generated datasets with sample sizes of 200, 300, 500, 1000, and 5000 for experiments.

TABLE III
Details of Bayesian Networks

Network	Num. Vars	Num. Edges	Max In/Out-Degree	Min/Max |PCset|
Child	20	25	2/7	1/8
Child3	60	79	3/7	1/8
Child5	100	126	2/7	1/8
Alarm3	111	149	4/5	1/6
Child10	200	257	2/7	1/8
Alarm10	370	570	4/7	1/9
Pigs	441	592	2/39	1/41
Link	724	1125	3/14	0/17

2) Comparison Methods: We choose the following seven algorithms for comparison: PC [16], PC-stable [20], GES [26], MMHC [31], BCSL [32], NOTEARS [34], DAG-GNN [35], and DAG-NoCurl [39]. PC and PC-stable are constraint-based methods. GES is a score-based method. MMHC and BCSL are hybrid methods. They are well-established combinatorial optimization methods. NOTEARS, DAG-GNN, and DAG-NoCurl are continuous optimization methods. More introduction about above algorithms is as follows:

- PC [16]: PC is a well-known and widely used constraint-based causal structure learning algorithm.
- PC-stable [20]: PC-stable is a well-established causal structure learning algorithm and partially solves the order dependency problem of PC.
- GES [26]: GES is a well-known score-based causal structure learning algorithm.
- MMHC [31]: MMHC is a well-established algorithm for learning large-scale causal structures.
- BCSL [32]: BCSL eliminates asymmetric edges in learned skeleton by using a data sampling technique, then uses the scoring function and the search strategy to orient edges in the local skeleton.
- NOTEARS [34]: NOTEARS is the first algorithm to use gradient descent to learn causal structures.
- DAG-GNN [35]: DAG-GNN learns causal structures using a variational auto-encoder.
- DAG-NoCurl [39]: DAG-NoCurl learns causal structures based on the graph Hodge theory.

3) Evaluation Metrics: We mainly evaluate the BLR algorithm in terms of the accuracy and structure error of the learned DAG. Therefore, we use four metrics for evaluation, which are Arc_P, Arc_R, Arc_F1, and $SHD(StructuralHammingDistance)$.

TP denotes the number of the true edges that are present in both the learned DAG and the true DAG, FP denotes the number of the edges that are present in the learned DAG but

[1]https://www.bnlearn.com/bnrepository/.

TABLE IV
RESULTS ON THE CHILD AND CHILD3 NETWORKS (↑ MEANS THE HIGHER THE BETTER; ↓ REPRESENTS THE LOWER THE BETTER)

Samples	Networks / Metrics / Algorithms	Child Arc_F1% (↑)	Child Arc_P% (↑)	Child Arc_R% (↑)	Child SHD(↓)	Child3 Arc_F1% (↑)	Child3 Arc_P% (↑)	Child3 Arc_R% (↑)	Child3 SHD(↓)
200	PC	22.22	20.69	24.00	31	19.05	17.98	20.25	99
	PC-stable	22.22	20.69	24.00	31	14.91	14.63	15.19	97
	GES	41.79	33.33	56.00	34	32.65	27.35	32.65	114
	MMHC	60.87	66.67	**56.00**	13	44.60	51.67	39.24	63
	BCSL	57.77	65.00	52.00	13	50.00	59.65	43.03	50
	NOTEARS	24.39	31.25	20.00	23	27.85	27.85	27.85	92
	DAG-GNN	23.81	29.41	20.00	24	25.00	36.59	18.99	77
	DAG-NoCurl	14.29	12.90	16.00	40	11.32	9.02	15.19	164
	BLR (ours)	**69.28**	**90.99**	**56.00**	**11.2**	**64.01**	**86.47**	**50.89**	**41**
300	PC	27.45	26.92	28.00	25	25.93	25.30	26.58	82
	PC-stable	19.61	19.23	20.00	27	22.50	22.22	22.78	84
	GES	58.62	51.52	**68.00**	21	37.56	31.36	46.84	103
	MMHC	65.31	66.67	64.00	13	57.97	67.80	50.63	45
	BCSL	65.21	71.24	60.00	12	58.99	68.33	51.89	43
	NOTEARS	29.27	37.50	24.00	19	30.99	34.92	27.85	74
	DAG-GNN	28.57	50.00	20.00	20	30.00	43.90	22.78	66
	DAG-NoCurl	17.78	20.00	16.00	29	14.38	14.86	13.92	104
	BLR (ours)	**77.56**	**90.66**	**68.00**	**8.8**	**63.48**	**77.69**	**53.67**	**38.4**
500	PC	32.65	33.33	32.00	20	31.94	35.38	29.11	64
	PC-stable	32.65	33.33	32.00	20	31.94	35.38	29.11	64
	GES	38.89	29.79	56.00	37	36.55	30.51	45.57	100
	MMHC	75.00	78.26	72.00	7	65.22	76.27	56.96	41
	BCSL	72.34	77.27	68.00	8	68.61	81.03	59.49	35
	NOTEARS	29.27	37.50	24.00	21	26.77	35.42	21.52	70
	DAG-GNN	26.32	38.46	20.00	22	22.86	46.15	15.19	69
	DAG-NoCurl	12.50	13.04	12.00	33	15.94	18.64	13.92	93
	BLR(ours)	**81.61**	**88.16**	**76.00**	**6**	**75.54**	**91.05**	**64.56**	**28.6**
1000	PC	41.67	43.48	40.00	16	38.62	42.42	35.44	57
	PC-stable	37.50	39.13	36.00	17	38.62	42.42	35.44	57
	GES	44.16	32.69	68.00	37	41.38	33.87	53.16	93
	MMHC	32.50	35.22	60.00	10	69.44	76.92	63.29	31
	BCSL	59.57	63.63	56.00	11	64.74	75.00	56.96	36
	NOTEARS	35.00	46.67	28.00	20	29.03	40.00	22.78	64
	DAG-GNN	31.58	46.15	24.00	21	31.67	46.34	24.05	63
	DAG-NoCurl	17.39	19.05	16.00	29	11.11	14.89	8.86	88
	BLR (ours)	**77.83**	**87.08**	**70.40**	**7.4**	**73.60**	**86.40**	**64.20**	**29.2**
5000	PC	64.00	64.00	64.00	9	65.82	65.82	65.82	28
	PC-stable	56.00	56.00	56.00	11	65.82	65.82	65.82	28
	GES	47.37	35.29	72.00	34	45.69	34.64	67.09	105
	MMHC	89.80	91.67	88.00	3	84.97	87.84	82.28	14
	BCSL	**96.00**	**96.00**	**96.00**	**1**	87.89	88.46	**87.34**	**11**
	NOTEARS	29.27	37.50	24.00	20	26.77	35.42	21.52	66
	DAG-GNN	25.64	35.71	20.00	21	31.58	51.43	22.78	64
	DAG-NoCurl	13.64	15.79	12.00	29	10.85	14.00	8.86	92
	BLR (ours)	**96.00**	**96.00**	**96.00**	**1**	**88.78**	**91.58**	85.97	13

not in the true DAG, and FN denotes the number of the edges that are present in the true DAG but not in the learned DAG. The specific definitions are as follows:

- $Arc\_P = \frac{TP}{TP+FP}$. The number of the true directed edges in the learned DAG divided by the total number of the edges in the learned DAG.
- $Arc\_R = \frac{TP}{TP+FN}$. The number of the true directed edges in the learned DAG divided by the total number of the edges in the true DAG.
- $Arc\_F1 = 2 \cdot \frac{Arc\_P \cdot Arc\_R}{Arc\_P + Arc\_R}$. Arc_F1 considers both of the Arc_P and Arc_R to comprehensively evaluate the performance of the algorithm.
- $SHD(Structural\ Hamming\ Distance)$. SHD is the total number of the wrong edges in the learned DAG. SHD is defined as the sum of the number of the extra edges, the missing edges, and the reverse edges in the learned DAG.

For these four evaluation metrics, the larger values of Arc_F1, Arc_P, and Arc_R, the better the effect; the smaller value of SHD, the better the effect.

4) Implementation Details: Our algorithm, PC, PC-stable, MMHC, and BCSL are implemented in MATLAB, GES is implemented in R, and NOTEARS, DAG-GNN, and DAG-NoCurl are implemented in Python. All experiments are conducted on a computer with Windows 10, Intel(R) Core(TM) i9-10900F 2.80GHz CPU, and 32GB memory. The significance level α is set 0.01 for the algorithms employing CI tests for causal structure learning. Following the experimental settings in NOTEARS, we adopt 0.3 as the threshold to prune edges in a DAG for NOTEARS, DAG-GNN, and DAG-NoCurl. Since the distributions of the N datasets sampled from the original dataset at different times vary, the experimental results may fluctuate. Therefore, we run our algorithm five times on all datasets and use the average result as the final result. In the experiment, we set N and the aggregation threshold ε to 10 and 5 for each dataset, respectively.

B. Experiment Results

Tables IV–VII show the experimental results of the above eight networks, and we can get the following findings according to the experimental results.

TABLE V
RESULTS ON THE CHILD5 AND ALARM3 NETWORKS (↑ MEANS THE HIGHER THE BETTER; ↓ REPRESENTS THE LOWER THE BETTER)

Samples	Networks / Metrics / Algorithms	Child5 Arc_F1% (↑)	Child5 Arc_P% (↑)	Child5 Arc_R% (↑)	Child5 SHD (↓)	Alarm3 Arc_F1% (↑)	Alarm3 Arc_P% (↑)	Alarm3 Arc_R% (↑)	Alarm3 SHD(↓)
200	PC	19.62	18.71	20.63	157	30.48	34.17	27.52	131
	PC-stable	19.53	19.23	19.84	150	26.22	29.66	23.49	135
	GES	26.67	20.08	39.68	243	32.17	23.79	49.66	286
	MMHC	61.33	69.70	**54.76**	73	33.21	67.36	**65.10**	80
	BCSL	63.50	78.82	53.17	65	68.8	85.14	57.71	66
	NOTEARS	22.73	21.74	23.81	162	51.64	66.32	42.28	108
	DAG-GNN	14.11	11.50	18.25	240	32.63	29.67	36.24	207
	DAG-NoCurl	9.19	6.23	17.46	393	15.49	9.71	38.26	601
	BLR (ours)	**65.89**	**88.68**	52.03	**62.2**	**73.89**	**94.90**	60.45	**63.2**
300	PC	24.70	24.80	24.60	129	41.73	50.48	35.57	108
	PC-stable	20.97	21.31	20.63	132	33.73	42.00	28.19	119
	GES	34.29	26.79	47.62	197	36.65	27.65	54.36	260
	MMHC	54.94	59.81	50.79	82	71.28	73.57	**69.13**	71
	BCSL	55.04	65.21	47.61	76	**72.06**	80.81	59.73	**62**
	NOTEARS	28.05	32.63	24.60	118	49.38	63.83	40.27	109
	DAG-GNN	25.56	42.59	18.25	111	38.63	43.57	33.20	144
	DAG-NoCurl	14.97	13.10	17.46	210	15.01	10.42	26.85	429
	BLR (ours)	**66.61**	**83.82**	**55.41**	**59.4**	72.01	**95.68**	58.23	64.2
500	PC	30.13	31.86	28.57	105	43.77	50.00	38.93	98
	PC-stable	27.62	29.20	26.19	108	40.15	47.27	35.90	104
	GES	38.90	29.71	56.35	192	32.98	24.21	51.68	280
	MMHC	72.81	81.37	65.87	46	69.69	72.46	67.11	69
	BCSL	70.85	80.44	62.69	50	81.67	94.7	71.81	44
	NOTEARS	30.48	38.10	25.40	107	50.86	71.08	39.60	104
	DAG-GNN	21.82	46.15	14.29	110	41.35	47.01	36.91	138
	DAG-NoCurl	17.48	22.50	14.29	132	23.20	17.73	33.56	310
	BLR (ours)	**77.22**	**90.62**	**67.41**	**43.2**	**84.24**	**99.82**	**72.41**	**40.4**
1000	PC	40.17	42.48	38.10	88	59.09	67.83	52.35	71
	PC-stable	40.17	42.48	38.10	88	56.49	65.49	49.66	75
	GES	41.23	31.76	58.73	179	47.69	37.40	65.77	193
	MMHC	73.28	80.19	67.46	42	86.62	91.11	**82.55**	36
	BCSL	73.12	82.17	65.87	43	85.5	95.83	77.18	37
	NOTEARS	30.62	38.55	25.40	103	51.95	73.17	40.27	99
	DAG-GNN	25.32	62.50	15.87	108	41.51	69.84	29.53	114
	DAG-NoCurl	18.54	24.05	15.08	130	37.63	39.13	36.24	154
	BLR (ours)	**75.62**	**85.62**	**68.01**	**41.2**	**87.03**	**99.62**	77.03	**35**
5000	PC	65.61	65.35	65.87	44	66.19	71.32	61.74	58
	PC-stable	64.03	63.78	64.29	46	64.49	70.08	59.73	61
	GES	36.41	26.22	59.52	217	49.53	38.18	70.47	189
	MMHC	82.11	84.17	80.16	25	80.41	80.95	79.87	44
	BCSL	87.4	88.67	**88.09**	**18**	**89.92**	96.9	**93.89**	**25**
	NOTEARS	32.00	43.24	25.40	99	53.28	76.25	40.94	99
	DAG-GNN	33.51	52.54	24.60	97	45.81	66.67	34.90	109
	DAG-NoCurl	16.22	25.42	11.90	123	40.73	44.44	37.58	142
	BLR (ours)	**89.46**	**93.46**	85.90	19.20	88.90	**97.24**	82.24	28.2

BLR versus PC and PC-stable. PC and PC-stable have comparable accuracy and structural error on all networks, while BLR outperforms them with higher accuracy and fewer structural errors across all networks. When the sample size is small, the results of the CI tests are tend to be unreliable. Incorrect CI test results lead to many errors in the process of learning and orienting the skeleton, resulting in low accuracy and large structural error. BLR adopts layerwise skeleton refining and collective scoring orientation strategies based on the bootstrap sampling method, which not only reduces the structural error of the learning skeleton but also improves the orientation accuracy.

As the sample size increases, the results of the CI tests gradually become more reliable, the accuracy of the skeleton learning and edges orientation increases, and the structural error decreases. However, BLR has advantages in high accuracy and low structural error at this time. In conclusion, BLR significantly improves the accuracy of the skeleton learning and edges orientation compared to constraint-based methods, especially when the sample size is small.

BLR versus GES, MMHC, and BCSL. To reach the highest score, GES usually keeps more edges in the graph, which results in large structural errors. Thus, GES almost always has a larger value of SHD than the constraint-based methods. The advantage of retaining more edges is that may have higher recall. For example, as shown in Table VII, GES has the highest recall on the Link network. When the sample size is small, on the networks of the Child, Child3, Child5, and Alarm3, GES is more accurate than the constraint-based methods, but on the Child10, Alarm10, Pigs, and Link networks, GES is less accurate than the constraint-based methods. As the sample size increases, the accuracy of GES cannot catch up with the constraint-based methods. In addition, the accuracy of GES is not as good as BLR.

MMHC and BCSL have high accuracy and few structural errors on all networks. With an increase in sample size, the accuracy of these two algorithms steadily improves and even reaches 100% on the Pigs network with 5000 samples. In comparison to MMHC, although the Arc_R of MMHHC is better

TABLE VI
RESULTS ON THE CHILD10 AND ALARM10 NETWORKS (↑ MEANS THE HIGHER THE BETTER; ↓ REPRESENTS THE LOWER THE BETTER)

Samples	Networks / Metrics / Algorithms	Child10 Arc_F1% (↑)	Child10 Arc_P% (↑)	Child10 Arc_R% (↑)	Child10 SHD(↓)	Alarm10 Arc_F1% (↑)	Alarm10 Arc_P% (↑)	Alarm10 Arc_R% (↑)	Alarm10 SHD(↓)
200	PC	22.83	21.36	24.51	320	27.24	32.37	23.51	543
	PC-stable	19.85	19.13	20.62	315	22.82	27.92	19.30	558
	GES	19.47	12.99	38.91	747	12.39	7.42	37.54	2919
	MMHC	48.80	55.45	43.58	186	60.37	64.60	**56.67**	383
	BCSL	53.17	67.26	43.96	159	**65.48**	89.12	51.75	**297**
	NOTEARS	18.18	15.60	21.79	419	41.11	52.30	33.86	507
	DAG-GNN	15.88	14.03	18.29	456	16.51	64.29	9.47	528
	DAG-NoCurl	2.30	1.23	18.68	3990	2.33	1.21	31.58	15030
	BLR (ours)	**57.02**	**79.68**	**44.89**	**153.8**	64.89	**90.68**	50.23	309
300	PC	25.00	25.10	24.90	266	33.93	42.90	28.07	459
	PC-stable	22.09	22.40	21.79	270	28.26	37.14	22.81	483
	GES	23.75	16.49	42.41	623	19.14	12.00	47.19	2192
	MMHC	45.82	50.95	41.63	202	63.44	66.22	**60.88**	358
	BCSL	51.05	64.11	42.41	163	**71.8**	94.3	58.07	**256**
	NOTEARS	22.80	23.46	22.18	303	44.89	61.21	35.44	449
	DAG-GNN	21.84	41.76	14.79	240	29.82	67.70	19.12	485
	DAG-NoCurl	7.48	4.85	16.34	968	2.73	1.41	41.04	16730
	BLR (ours)	**60.89**	**80.30**	**49.01**	**136**	69.23	**94.90**	54.67	273.4
500	PC	36.13	39.27	33.46	194	43.66	53.57	36.84	395
	PC-stable	32.49	35.48	29.96	203	30.97	49.07	32.46	418
	GES	28.11	19.97	47.47	544	24.50	16.19	50.35	1692
	MMHC	71.74	81.28	64.20	101	70.12	74.17	**66.49**	293
	BCSL	69.81	82.88	60.31	106	**76.82**	96.8	63.68	**213**
	NOTEARS	28.91	36.97	23.74	225	47.33	69.86	35.79	417
	DAG-GNN	25.21	47.83	17.12	225	25.96	69.47	15.96	495
	DAG-NoCurl	14.26	13.77	14.79	384	8.45	4.94	29.12	3521
	BLR (ours)	**77.85**	**91.58**	**67.70**	**87**	74.89	**97.02**	61.01	229.6
1000	PC	42.56	45.37	40.08	172	49.95	60.65	42.46	334
	PC-stable	40.66	43.56	38.13	176	46.83	57.54	39.47	346
	GES	32.93	23.90	52.92	473	32.14	22.52	56.14	1265
	MMHC	71.25	78.93	**64.92**	96	71.16	75.20	67.54	263
	BCSL	70.92	81.72	62.64	96	**81.76**	**98.27**	**70**	**176**
	NOTEARS	28.71	39.46	22.57	213	49.01	72.51	37.02	405
	DAG-GNN	25.08	64.52	15.56	208	30.18	69.18	19.3	485
	DAG-NoCurl	18.65	27.91	14.01	250	27.97	23.75	34.04	926
	BLR (ours)	**72.45**	**84.24**	63.67	**95**	81.02	98.24	68.89	182.4
5000	PC	62.75	63.24	62.26	97	60.71	68.97	54.21	265
	PC-stable	60.00	60.47	59.53	104	59.09	67.65	52.46	271
	GES	34.89	24.96	57.98	457	40.92	30.52	62.11	917
	MMHC	86.23	88.52	84.05	42	79.85	84.81	75.44	174
	BCSL	81.52	84.23	78.98	54	83.89	95.31	74.91	146
	NOTEARS	30.62	41.89	24.12	208	49.41	76.47	36.49	396
	DAG-GNN	28.41	55.68	19.07	213	39.63	58.36	30.00	458
	DAG-NoCurl	16.58	24.81	12.45	260	41.74	54.86	33.68	472
	BLR (ours)	**89.24**	**92.68**	**85.90**	**39.4**	**85.24**	**97.24**	**75.46**	**140.4**

than BLR on a small number of networks, most of the metrics are better than MMHC on all networks and BLR always have lower SHD. Compared to BCSL, most metrics of BCSL are better than BLR on the Child10, Alarm10, and Pigs networks, but on other networks and datasets of 5000 samples, the accuracy of BLR is higher than BCSL.

BLR versus NOTEARS, DAG-GNN, and DAG-NoCurl. On all networks, BLR consistently maintains a large advantage over the continuous optimization methods in terms of accuracy and structural error. On the small- and medium-sized networks with small sample sizes, the accuracy of the continuous optimization methods is not much different from the combinatorial optimization methods as shown in Tables IV–VI. Although the accuracy improvement is not as fast as the combinatorial optimization methods, it increases steadily with the increase of the sample size.

On the large network of the Pigs and Link, as shown in Table VII, the performance of the continuous optimization methods is unsatisfactory. The accuracy of NOTEARS is not very high, and the accuracy doesn't increase with the increase of the sample size but has a downward trend; the accuracy of DAG-GNN is low or even 0 when the sample size is small, and the accuracy gradually improves as the sample size increases; the accuracy of DAG-NoCurl is not only low but also the value of SHD has large fluctuations with the change of sample size, showing its unstable properties. Therefore, compared with the continuous optimization methods, BLR has great advantages in both the accuracy and stability of causal structure learning.

Overall, BLR outperforms other well-established algorithms in terms of the accuracy and structural error for learning causal structures on the above benchmark datasets.

C. Experiment Analysis

1) Analysis of Distribution Differences Between Sampled Datasets and Original Dataset: Bootstrap sampling is a random sampling method with replacement that increase data diversity and generate sampled datasets with similar distribution

TABLE VII
RESULTS ON THE PIGS AND LINK NETWORKS (↑ MEANS THE HIGHER THE BETTER; ↓ REPRESENTS THE LOWER THE BETTER)

Samples	Networks / Metrics / Algorithms	Pigs Arc_F1% (↑)	Pigs Arc_P% (↑)	Pigs Arc_R% (↑)	Pigs SHD(↓)	Link Arc_F1% (↑)	Link Arc_P% (↑)	Link Arc_R% (↑)	Link SHD(↓)
200	PC	41.03	37.17	45.78	458	15.42	12.76	19.47	2200
	PC-stable	40.89	37.39	45.10	447	15.64	13.37	18.84	2089
	GES	26.36	15.24	**97.30**	3204	7.13	4.18	**24.27**	6975
	MMHC	89.45	86.69	92.40	98	23.76	31.39	19.11	1258
	BCSL	94.46	**93.83**	95.10	**51**	28.99	44.69	20.97	1079
	NOTEARS	50.51	51.22	49.83	355	1.07	15.15	8.27	1422
	DAG-GNN	0.00	0.00	0.00	592	0.00	0.00	0.00	1125
	DAG-NoCurl	4.37	2.35	30.91	7876	0.58	0.31	4.36	16608
	BLR (ours)	**94.46**	92.90	96.02	55.8	**31.23**	**61.89**	20.88	**1004.8**
300	PC	45.88	40.93	52.20	446	14.43	11.34	19.82	2417
	PC-stable	42.15	37.80	47.64	464	14.39	11.60	18.93	2306
	GES	34.97	21.32	97.30	2127	14.57	9.75	**28.80**	3681
	MMHC	93.87	92.05	95.78	61	24.46	32.64	19.56	1222
	BCSL	96.21	**95.81**	96.62	**37**	32.13	49.00	23.91	1058
	NOTEARS	44.28	44.13	44.43	399	14.58	25.44	10.22	1213
	DAG-GNN	0.00	0.00	0.00	592	0.00	0.00	0.00	1125
	DAG-NoCurl	2.24	1.35	6.42	3284	0.46	0.24	4.36	20985
	BLR (ours)	**96.24**	94.46	**97.90**	40.2	**34.01**	**63.67**	23.48	**981.4**
500	PC	62.07	61.81	62.33	228	25.07	25.53	24.62	1456
	PC-stable	55.77	55.63	55.91	264	24.70	25.65	23.82	1432
	GES	45.29	29.28	100.00	1430	20.00	13.70	**36.98**	3216
	MMHC	96.99	95.87	98.14	25	29.12	36.40	24.27	1180
	BCSL	**98.57**	**98.16**	98.98	**11**	**37.01**	53.62	28.26	1014
	NOTEARS	42.64	41.75	43.58	415	17.36	35.73	11.47	1080
	DAG-GNN	0.00	0.00	0.00	592	0.00	0.00	0.00	1125
	DAG-NoCurl	22.25	15.54	39.19	1373	1.03	0.54	12.00	25709
	BLR (ours)	97.46	96.24	**99.24**	24.2	35.67	**60.01**	25.23	**994.2**
1000	PC	75.93	75.67	76.18	145	29.46	28.89	30.04	1435
	PC-stable	74.03	73.91	74.16	155	27.96	28.10	27.82	1421
	GES	55.11	38.13	99.32	954	24.29	16.61	**45.16**	3050
	MMHC	98.57	98.16	98.99	11	34.84	41.98	29.78	1121
	BCSL	**99.49**	**99.32**	**99.66**	**4**	37.24	51.82	29.06	1008
	NOTEARS	40.91	39.53	42.40	427	15.42	35.24	9.87	1070
	DAG-GNN	0.30	0.13	0.17	592	0.00	0.00	0.00	1126
	DAG-NoCurl	31.56	30.15	33.13	567	1.29	0.70	8.27	14152
	BLR (ours)	99.02	98.24	99.46	10.2	**38.67**	**62.23**	28.23	**952.4**
5000	PC	99.41	99.16	99.66	5	55.63	78.70	43.02	676
	PC-stable	99.66	99.49	99.83	3	55.19	79.33	42.31	669
	GES	71.98	56.22	100.00	461	20.23	12.86	**47.47**	4110
	MMHC	**100.00**	**100.00**	**100.00**	**0**	52.36	63.00	44.80	843
	BCSL	**100.00**	**100.00**	**100.00**	**0**	59.26	79.61	47.20	683
	NOTEARS	40.99	38.82	43.41	436	17.03	37.46	11.02	1058
	DAG-GNN	9.79	14.33	7.43	584	0.86	1.19	0.44	1128
	DAG-NoCurl	30.86	30.48	31.25	546	3.91	2.55	8.44	4604
	BLR (ours)	**100.00**	**100.00**	**100.00**	**0**	**59.67**	**88.46**	45.01	**659.2**

characteristics to the original dataset. Therefore, these sampled datasets can be used to train causal structure learning models to identify causal relationships in the original dataset. We conduct visual analyses of the data distribution to illustrate the similarity between the sampled datasets and the original dataset. Furthermore, we calculate the distance between the sampled datasets and the original dataset, revealing slight distribution differences between them.

We use the Child network with 500 samples. As shown in Fig. 5, we visualize the distribution of the original dataset and the distributions of five datasets sampled through bootstrap sampling. Each sampled dataset has the similar distribution as that of the original dataset.

To quantify the distribution differences between the original and sampled datasets of the Child and Child3 network, we utilize the maximum mean discrepancy (MMD). MMD is a widely used quantitative measure of the difference between

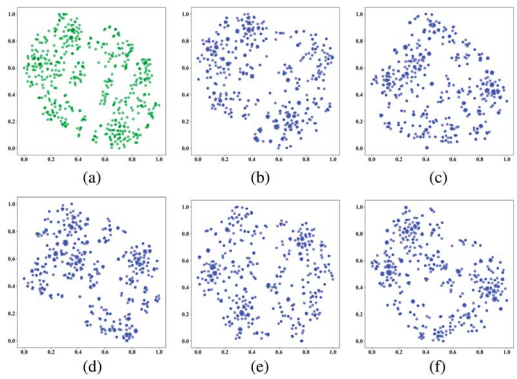

Fig. 5. Sample distribution of the original and the sampled datasets of Child network. In each subfigure, the points represent the feature representations of the data samples. The positions of the points represent the distribution of data samples in the feature space. (a) Original dataset, (b) sampled dataset 1, (c) sampled dataset 2, (d) sampled dataset 3, (e) sampled dataset 4, and (f) sampled dataset 5.

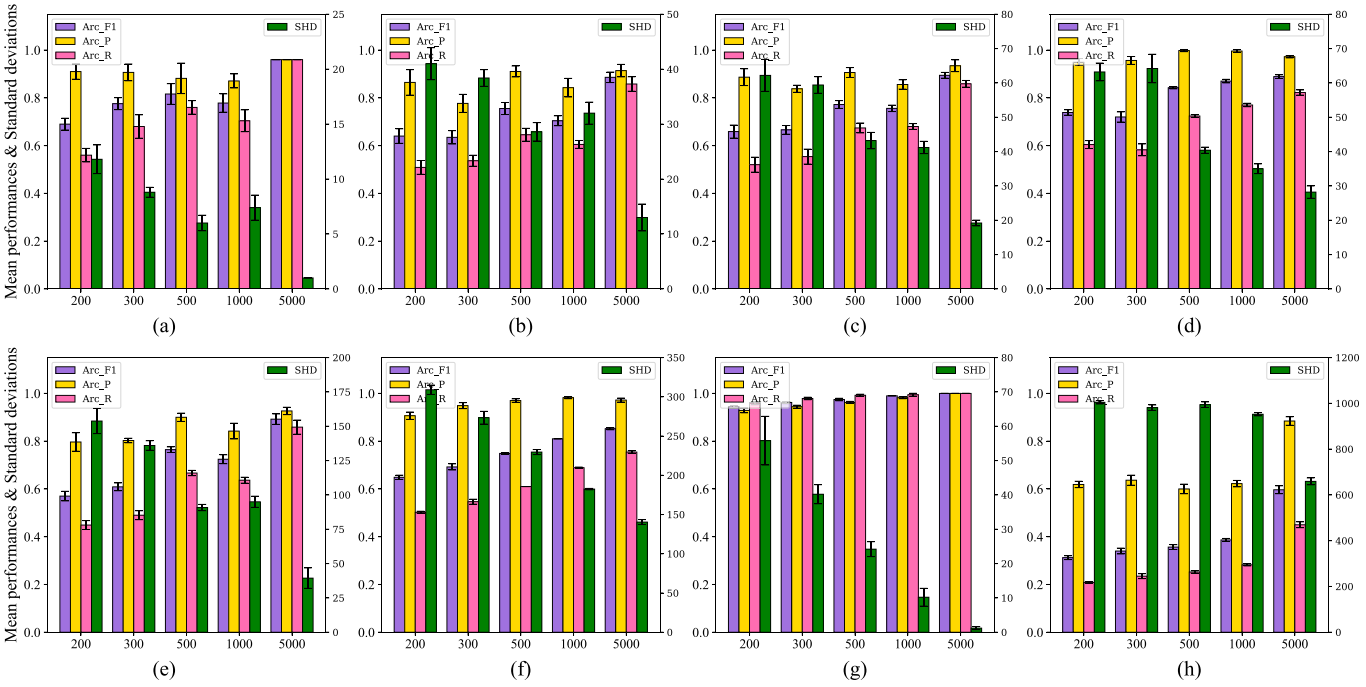

Fig. 6. Mean performances and standard deviations (error bars) of BLR on different networks. The left y-axis is the metrics of Arc_F1, Arc_P, Arc_R, the right y-axis is the metric of SHD, and the x-axis is the number of samples of the network. The low standard deviation of the evaluation metrics shows that BLR has good stability on all datasets. (a) Child network, (b) Child3 network, (c) Child5 network, (d) Alarm3 network, (e) Child10 network, (f) Alarm10 network, (g) Pigs network, and (h) Link network.

TABLE VIII
MMD BETWEEN ORIGINAL AND TEN SAMPLED DATASETS

		MMD									
Networks	Samples	1	2	3	4	5	6	7	8	9	10
Child	200	0.221	0.220	0.217	0.221	0.219	0.218	0.222	0.222	0.219	0.218
	300	0.222	0.222	0.218	0.223	0.221	0.22	0.224	0.224	0.221	0.219
	500	0.217	0.217	0.214	0.218	0.216	0.215	0.219	0.219	0.216	0.215
	100	0.221	0.221	0.217	0.222	0.220	0.219	0.223	0.223	0.220	0.219
	500	0.219	0.220	0.215	0.220	0.218	0.217	0.221	0.222	0.218	0.217
Child3	200	0.227	0.225	0.226	0.229	0.229	0.230	0.226	0.226	0.224	0.230
	300	0.227	0.225	0.226	0.229	0.229	0.230	0.226	0.226	0.224	0.230
	500	0.226	0.224	0.225	0.228	0.228	0.229	0.225	0.226	0.224	0.229
	100	0.227	0.225	0.226	0.229	0.228	0.230	0.226	0.226	0.224	0.230
	500	0.226	0.225	0.226	0.228	0.228	0.229	0.226	0.226	0.224	0.229

two probability distributions in dataset comparison. Given two datasets D_i and D_j, MMD is defined as (4):

$$\mathrm{MMD}(D_i, D_j) = \left\| \frac{1}{n_i} \sum_{k=1}^{n_i} \Phi\left(X_k^i\right) - \frac{1}{n_j} \sum_{k=1}^{n_j} \Phi\left(X_k^j\right) \right\|_{\mathcal{H}}, \quad (4)$$

where n_i and n_j are the number of samples in D_i and D_j, $\Phi(X)$ is a feature map kernel function from X to $\mathcal{H}$ which is the reproducing kernel Hilbert space (RKHS) being high-dimensional or even infinite-dimensional.

In Table VIII, we show the MMDs of the original and 10 sampled datasets on two networks. While the MMDs may be somewhat different for different networks, the MMDs between the original and sampled datasets are similar on the same network. Therefore, the bootstrap sampling method improves the diversity of data samples. The skeletons can be learned separately from each sampled dataset and aggregated to improve the accuracy of causal structure learning.

2) Stability Analysis of Experimental Results: To verify the stability of our method, we conduct multiple experiments on all datasets under the same parameter settings. Specifically, we set the number of sampled datasets to 10 ($N = 10$) and the aggregation threshold to 5 ($\varepsilon = 5$). We conducted five experiments on each dataset of all networks and analyzed all metrics of the experimental results. The mean performances and standard deviations of all metrics were depicted in Fig. 6. The calculation formulas of the mean and standard deviation are shown in (5) and (6), where the notation M is the number of experiments, and the notation E is a metric. It can be instantiated as Arc_F1, Arc_P, Arc_R, and SHD.

$$\mathrm{Mean}_E = \frac{1}{M} \sum_{i=1}^{M} E_i. \quad (5)$$

$$\mathrm{Std}_E = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (E_i - \mathrm{Mean}_E)^2}. \quad (6)$$

In Fig. 6, the height of each bar represents the mean of the results, while the error bars represent the standard deviation of the results. For convenience, we used a dual y-axis coordinate system in the subfigure. The x-axis is the number of samples of each network. The metrics of Arc_F1, Arc_P, and Arc_R are displayed on the left y-axis in the subfigure, while the metric of SHD is displayed on the right y-axis. The title of each subfigure is the network name. As depicted in Fig. 6, our experimental results exhibit small standard deviations, which indicate that our experimental results are stable and demonstrate the robustness of our algorithm.

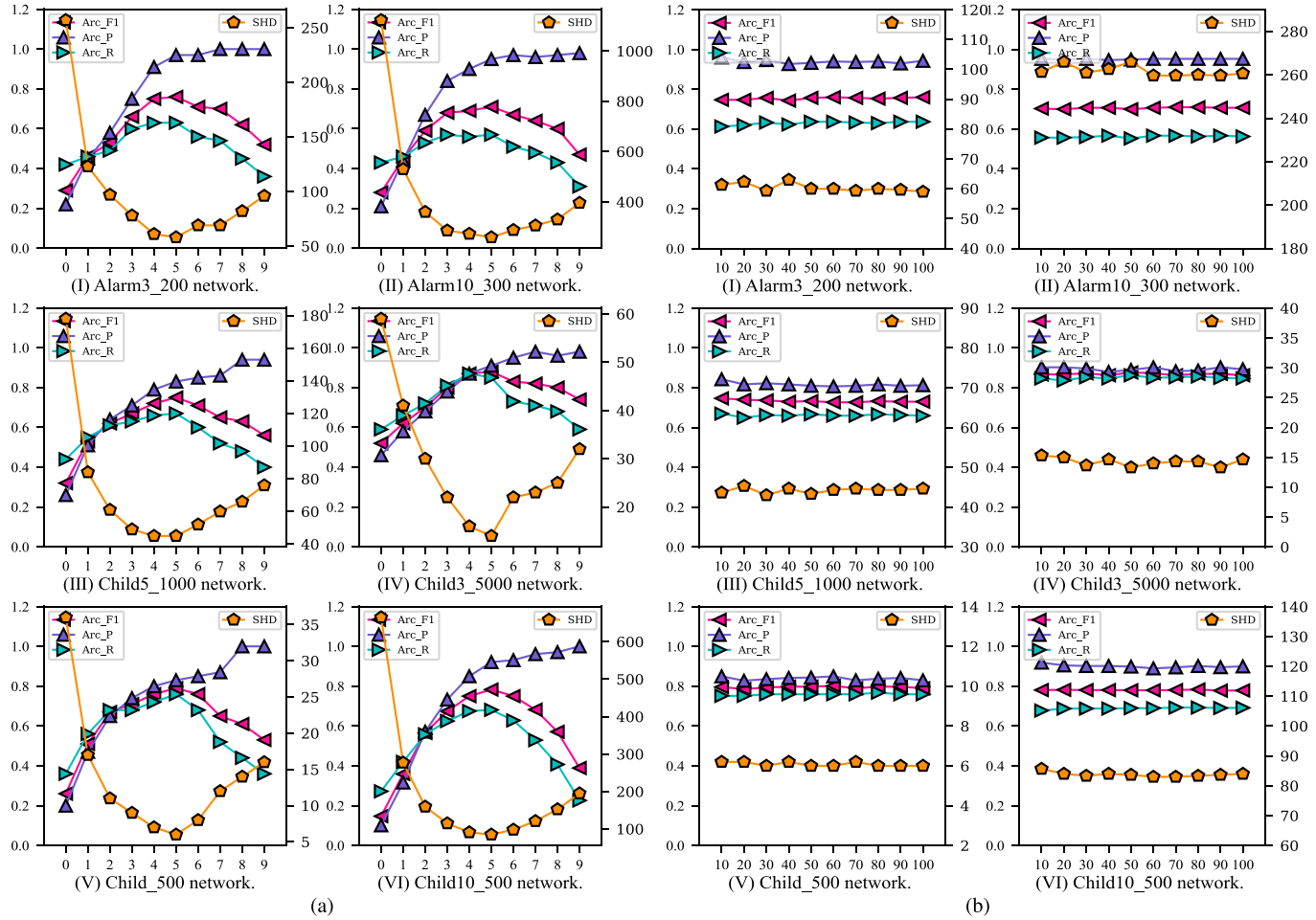

Fig. 7. Sensitivity analysis of parameter ε and N of BLR under different networks In (a), the left y-axis is the metrics of Arc_F1, Arc_P, and Arc_R, the right y-axis is the metric of SHD, and the x-axis is the value of ε. (a) Shows that most of the metrics are the best when $\varepsilon = N/2$. In (b), the left y-axis is the metrics of Arc_F1, Arc_P, and Arc_R, the right y-axis is the metric of SHD, and the x-axis is the value of N. (b) Shows that the metrics are stable as the value of N changes. (a) The curve of the metrics changing with $\varepsilon(N = 10)$. (b) The curve of the metrics changing with $N(\varepsilon = N/2)$.

3) Sensitivity Analysis of Parameters: In the experiments, we need to determine the number N of sampled datasets for bootstrap sampling and set an appropriate threshold ε for aggregation. Therefore, in this section, we analyze the influence of the number N and the setting of ε on the results. To this end, we chose six different networks with varying sample sizes and conducted experiments with different threshold settings under the same number of N to find the optimal threshold setting method. Then, we set different values of N using the optimal threshold ε setting method to analyze the stability of BLR on the number of sampled datasets.

Specifically, we conduct experiments by setting N to 10 and varying the threshold ε from 0 to 9 to determine the optimal threshold setting method for ε based on the experimental results. The curve of the experimental metrics changing with ε is shown in Fig. 7(a). In addition, we set $\varepsilon = N/2$ and set the value of N from 10, 20, to 100 to verify the stability of our algorithm with respect to different values of N. The curve of the experimental metrics changing with N is shown in Fig. 7(b). As in the previous experiments, we conduct five experiments on each dataset and took the average of the results as the final result. The title of each subfigure is the name of the network and the corresponding sample size. In each subfigure, the results for Arc_F1, Arc_P, and Arc_R are displayed on the left y-axis and the result for SHD is displayed on the right y-axis. In Fig. 7(a), the x-axis is the value of ε. In Fig. 7(b), the x-axis is the value of N. Through analysis, we can draw the following conclusions:

1) Under the same value of N, the learned DAG has a large value of SHD when the value of ε is small. Since the value of ε is too small, many edges are remained in the final skeleton. After orienting the skeleton using the scoring function, there are many extra edges in the DAG. At the same time, the values of Arc_F1, Arc_P, and Arc_R are very low. The value of SHD decreases as the value of ε increases since the number of edges in the final skeleton decreases, and then the extra edges in the DAG after orienting the skeleton decrease. The values of Arc_F1, Arc_P, and Arc_R also increase as the value of ε increases until the optimal threshold $N/2$ is reached. When the value of ε is close to $N/2$, the SHD reaches its minimum value, while the Arc_F1 and Arc_R reach their maximum values. The value of SHD increases when the value of ε continues to increase,

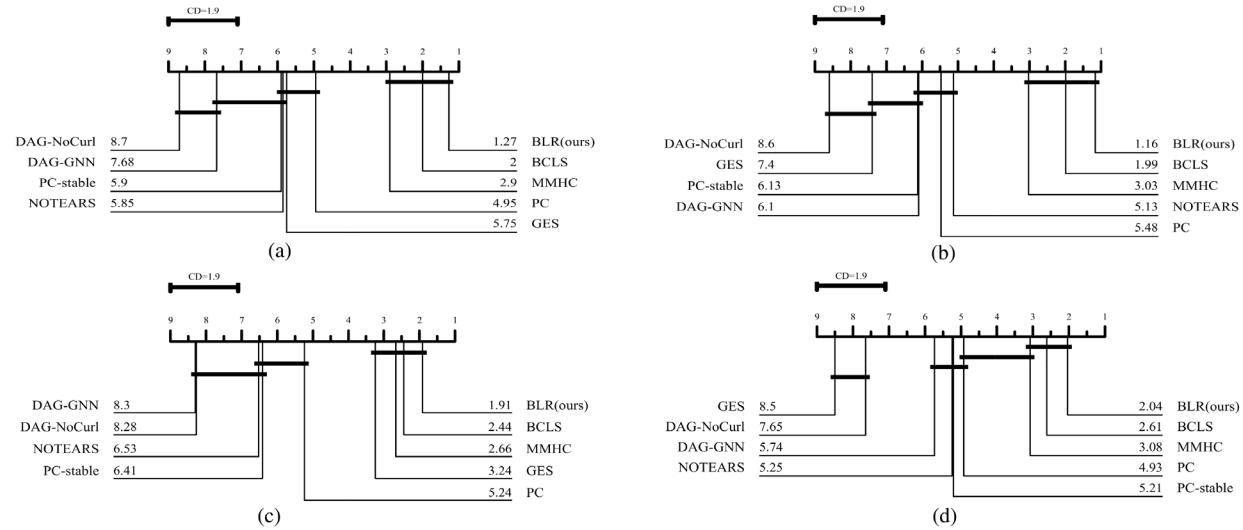

Fig. 8. Comparison of BLR against its rivals with the Nemenyi test (the lower the rank value, the better the performance). (a) Arc_F1 metric, (b) Arc_P metric, (c) Arc_R metric, and (d) SHD metric.

which is caused by the skeleton containing only a few edges, resulting in missing a lot of edges in the DAG. At this time, the advantage of using scoring for orientation is that the orientation of the undirected edges in the skeleton will be more accurate and efficient, that is, the value of Arc_P will continue to increase. However, the missing edges in the DAG make the low values of Arc_R and Arc_F1. Through the analysis of the experimental results, we know that the optimal threshold setting method is $N/2$. Except for the Arc_P, all evaluation metrics on the benchmark networks are optimal when the value of ε is set to $N/2$.

2) Under the optimal threshold setting method, we examine the impact of the number of sampled datasets on our experimental results. Specifically, we set the value of ε to $N/2$ when the value of N changes. At this point, we can see that each evaluation metrics of the experimental results has a very small fluctuation with the changes of N, which shows that our algorithm is stable to changes of the parameter N.

4) Statistical Test: To comprehensively evaluate the performance of our algorithm, we utilize the Nemenyi test [43] to compare BLR with other algorithms on all datasets. The test states that two algorithms are significantly different if their corresponding average ranks differ by at least one critical difference (CD). Fig. 8 illustrates CD diagrams for the evaluation metrics, where each algorithm's average rank is marked along the axis (lower ranks to the right). As shown in Fig. 8(a)–(d), BLR consistently exhibits the lowest average rank on all evaluation metrics. BLR demonstrates comparable performance with BCSL and MMHC and performs significantly better than NOTEARS, DAG-GNN, DAG-NoCurl, and PC-stable on all evaluation metrics.

VI. Conclusion

In this article, we propose the BLR algorithm for causal structure learning, which aims to address the problem of inaccurate causal structures resulting from incorrect conditional independence tests in constraint-based methods. Specifically, BLR first employs a layerwise refining strategy to construct a reliable skeleton of the DAG. Subsequently, BLR uses a scoring technique to collectively orient the skeleton to improve the accuracy of the orientation. The experimental results show that BLR performs better than other state-of-the-art algorithms on eight benchmark Bayesian Network datasets. However, BLR also has some limitations. For example, BLR cannot be used to discover causal relationships from the datasets with latent variables and has poor efficiency in identifying the local causal relationships of a given target variable.

In future, we plan to combine the bootstrap-based method with other techniques such as deep learning, which has the potential to improve the performance of causal structure learning from the datasets containing latent variables. In addition, we will extend BLR to accurately and efficiently learn the local causal structure of a given target variable. Furthermore, we intend to expand and apply BLR to learn causal relationships in knowledge graphs and images, which can enhance the generalization ability of models.

References

[1] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman, *Causation, Prediction, and Search*. Cambridge, MA, USA: MIT Press, 2000.
[2] R. Cai, J. Qiao, Z. Zhang, and Z. Hao, "Self: Structural equational likelihood framework for causal discovery," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1787–1794.
[3] X. Guo, K. Yu, L. Liu, P. Li, and J. Li, "Adaptive skeleton construction for accurate DAG learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 10526–10539, Oct. 2023.
[4] S. Triantafillou, V. Lagani, C. Heinze-Deml, A. Schmidt, J. Tegner, and I. Tsamardinos, "Predicting causal relationships from biological data: Applying automated causal discovery on mass cytometry data of human immune cells," *Sci. Rep.*, vol. 7, no. 1, pp. 1–11, 2017.
[5] D. C. Castro, I. Walker, and B. Glocker, "Causality matters in medical imaging," *Nat. Commun.*, vol. 11, no. 1, pp. 1–10, 2020.
[6] Z. Chen, Z. Tian, J. Zhu, C. Li, and S. Du, "C-CAM: Causal CAM for weakly supervised semantic segmentation on medical image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11676–11685.

[7] T. Kyono and M. van der Schaar, "Exploiting causal structure for robust model selection in unsupervised domain adaptation," *IEEE Trans. Artif. Intell.*, vol. 2, no. 6, pp. 494–507, Dec. 2021.
[8] K. Zhang, M. Gong, P. Stojanov, B. Huang, Q. Liu, and C. Glymour, "Domain adaptation as a problem of inference on graphical models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 4965–4976.
[9] K. Yu, L. Liu, J. Li, W. Ding, and T. D. Le, "Multi-source causal feature selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 9, pp. 2240–2256, Sep. 2020.
[10] S. Yang, K. Yu, F. Cao, L. Liu, H. Wang, and J. Li, "Learning causal representations for robust domain adaptation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 3, pp. 2750–2764, Mar. 2023.
[11] Y. Akkem, S. K. Biswas, and A. Varanasi, "Smart farming using artificial intelligence: A review," *Eng. Appl. Artif. Intell.*, vol. 120, 2023, Art. no. 105899.
[12] M. Li, P.-Y. Huang, X. Chang, J. Hu, Y. Yang, and A. Hauptmann, "Video pivoting unsupervised multi-modal machine translation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3918–3932, Mar. 2023.
[13] H. Zhang et al., "Attribute-guided collaborative learning for partial person re-identification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 12, pp. 14144–14160, Dec. 2023.
[14] L. Zhang et al., "TN-ZSTAD: Transferable network for zero-shot temporal activity detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3848–3861, Mar. 2023.
[15] D. Colombo, M. H. Maathuis, M. Kalisch, and T. S. Richardson, "Learning high-dimensional directed acyclic graphs with latent and selection variables," *Ann. Statist.*, vol. 40, no. 1, pp. 294–321, 2012.
[16] P. Spirtes, C. Glymour, and R. Scheines, "Causality from probability," in *Evolving Knowledge in Natural Science and Artificial Intelligence*, pp. 181–199, 1989.
[17] C. Meek, "Causal inference and causal explanation with background knowledge," in *Proc. Conf. Uncertainty Artif. Intell.*, 1995, pp. 403–410.
[18] M. Kalisch and P. Bühlman, "Estimating high-dimensional directed acyclic graphs with the PC-algorithm." *J. Mach. Learn. Res.*, vol. 8, no. 3, pp. 613–636, 2007.
[19] J. Zhang and P. Spirtes, "Strong faithfulness and uniform consistency in causal inference," in *Proc. Conf. Uncertainty Artif. Intell.*, 2002, pp. 632–639.
[20] D. Colombo et al., "Order-independent constraint-based causal structure learning." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3741–3782, 2014.
[21] H. Li, V. Cabeli, N. Sella, and H. Isambert, "Constraint-based causal structure learning with consistent separating sets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 14257–14266.
[22] E. Giudice, J. Kuipers, and G. Moffa, "The dual PC algorithm for structure learning," in *Proc. Int. Conf. Probab. Graphical Models.*, 2022, pp. 301–312.
[23] A. Sondhi and A. Shojaie, "The reduced PC-algorithm: Improved causal structure learning in large random networks." *J. Mach. Learn. Res.*, vol. 20, no. 164, pp. 1–31, 2019.
[24] X. Qi, X. Fan, H. Wang, L. Lin, and Y. Gao, "Mutual-information-inspired heuristics for constraint-based causal structure learning," *Inf. Sci.*, vol. 560, pp. 152–167, 2021.
[25] M. Kocaoglu, A. Jaber, K. Shanmugam, and E. Bareinboim, "Characterization and learning of causal graphs with latent variables from soft interventions," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 14369–14379.
[26] D. M. Chickering, "Optimal structure identification with greedy search," *J. Mach. Learn. Res.*, vol. 3, no. Nov, pp. 507–554, 2002.
[27] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Mach. Learn.*, vol. 20, no. 3, pp. 197–243, 1995.
[28] M. J. Vowels, N. C. Camgoz, and R. Bowden, "D'ya like DAGs? A survey on structure learning and causal discovery," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–36, 2021.
[29] A. Hauser and P. Bühlmann, "Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 2409–2464, 2012.
[30] R. Cai, S. Wu, J. Qiao, Z. Hao, K. Zhang, and X. Zhang, "THPs: Topological Hawkes processes for learning causal structure on event sequences," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2022, doi: 10.1109/TNNLS.2022.3175622.
[31] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The Max–Min hill-climbing Bayesian network structure learning algorithm," *Mach. Learn.*, vol. 65, no. 1, pp. 31–78, 2006.
[32] X. Guo, Y. Wang, X. Huang, S. Yang, and K. Yu, "Bootstrap-based causal structure learning," in *Proc. ACM Int. Conf. Inf. Knowl. Manag.*, 2022, pp. 656–665.
[33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[34] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "DAGs with no tears: Continuous optimization for structure learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 9492–9503.
[35] Y. Yu, J. Chen, T. Gao, and M. Yu, "DAG-GNN: DAG structure learning with graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7154–7163.
[36] X. Zheng, C. Dan, B. Aragam, P. Ravikumar, and E. Xing, "Learning sparse nonparametric DAGs," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2020, pp. 3414–3425.
[37] S. Lachapelle, P. Brouillard, T. Deleu, and S. Lacoste-Julien, "Gradient-based neural DAG learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–23.
[38] I. Ng, S. Zhu, Z. Fang, H. Li, Z. Chen, and J. Wang, "Masked gradient-based causal structure learning," in *Proc. SIAM Int. Conf. Data Min.*, 2022, pp. 424–432.
[39] Y. Yu, T. Gao, N. Yin, and Q. Ji, "DAGs with no curl: An efficient DAG structure learning approach," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12156–12166.
[40] A. Zanga, E. Ozkirimli, and F. Stella, "A survey on causal discovery: Theory and practice," *J. Approx. Reasoning*, vol. 151, pp. 101–129, 2022.
[41] M. Glymour, J. Pearl, and N. P. Jewell, *Causal Inference in Statistics: A Primer*. Hoboken, NJ, USA: Wiley, 2016.
[42] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Boca Raton, FL, USA: CRC Press, 1994.
[43] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.