Front. Comput. Sci., 2021, 15(6): 156337
https://doi.org/10.1007/s11704-020-0268-6

RESEARCH ARTICLE

BIC-based node order learning for improving Bayesian network structure learning

Yali LV[1,2], **Junzhong MIAO**[1], **Jiye LIANG**[2], **Ling CHEN**[3], **Yuhua QIAN** (✉)[2,4]

1 Shanxi University of Finance & Economics, Taiyuan 030031, China
2 Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Shanxi University, Taiyuan 030006, China
3 The Center for Artificial Intelligence, University of Technology Sydney, New South Wales 2007, Australia
4 Institute of Big Data Science & Industry, Shanxi University, Taiyuan 030006, China

Abstract Node order is one of the most important factors in learning the structure of a Bayesian network (BN) for probabilistic reasoning. To improve the BN structure learning, we propose a node order learning algorithm based on the frequently used Bayesian information criterion (BIC) score function. The algorithm dramatically reduces the space of node order and makes the results of BN learning more stable and effective. Specifically, we first find the most dependent node for each individual node, prove analytically that the dependencies are undirected, and then construct undirected subgraphs $\mathrm{U_G}$. Secondly, the $\mathrm{U_G}$ is examined and connected into a single undirected graph $\mathrm{U_{GC}}$. The relation between the subgraph number and the node number is analyzed. Thirdly, we provide the rules of orienting directions for all edges in $\mathrm{U_{GC}}$, which converts it into a directed acyclic graph (DAG). Further, we rank the DAG's topology order and describe the BIC-based node order learning algorithm. Its complexity analysis shows that the algorithm can be conducted in linear time with respect to the number of samples, and in polynomial time with respect to the number of variables. Finally, experimental results demonstrate significant performance improvement by comparing with other methods.

Keywords probabilistic reasoning, Bayesian networks, node order learning, structure learning, BIC scores, V-structure

1 Introduction

Bayesian networks (BNs) [1] are well-known probabilistic graphical models that can be used to represent and deal with uncertainty in terms of probabilistic relationships. Their graphical and probabilistic nature makes such networks rather popular in uncertainty reasoning. Consequently, BNs have been applied in a wide range of fields, including bioinformatics [2, 3], a visual analysis of geospatial data [4], camera detection of pedestrians [5], complex activity recognition [6], and crime analysis [7].

Received June 12, 2020; accepted October 15, 2020

E-mail: jinchengqyh@126.com

To enable probabilistic reasoning based on a BN, a critical and necessary task is to learn the BN structure, which is composed of two sequential problems. First, the directed acyclic graph (DAG) of a BN that best captures the relationships between variables should be found. Second, the parameters quantifying the probabilities of the relationships should be learned. Because the parameters can be derived in a straightforward manner (e.g., using a maximum likelihood estimation) when the DAG structure is available, most existing studies have focused on the first problem of learning the best DAG structure of a BN.

However, learning the DAG structure of a BN from the given data is NP-hard [8, 9]. As derived by Robinson [10], the recursive formula for DAGs' number of a BN with n variables is as follows:

$$f(n) = \sum_{i=1}^{n}(-1)^{i+1}\binom{n}{i}2^{i(n-i)}f(n-i) = n^{2^{O(n)}}. \tag{1}$$

That is, the search space required to find the best DAG structure is exponential to the variables' number. Therefore, it is a challenging problem to learn the BN structure for a large dataset with numerous variables.

In general, most studies use score-&-search learning methods to learn a BN structure. These methods use heuristic strategies to search the DAG space of possible candidates and evaluate the quality of the candidates with respect to predefined scoring functions. The DAG structure with an optimal score will then be returned. A number of scoring functions are used, such as Bayesian Dirichlet likelihood-equivalent (BDe) [11] and Minimum Description Length (MDL) [12]. In this study, we follow most existing methods to adopt the Bayesian information criterion (BIC) [13, 14] as the metric, and search the DAG with the optimal BIC score. Many heuristic constraint strategies are devised to improve the search efficiency, such as limiting the number of parent nodes for each node in a DAG [15], and restricting the treewidth of a DAG [16–19]. Among such strategies, *node ordering* is one of the most potential schemes used to reduce the DAG search space. Basically, node ordering generates an order among variables such that the preceding variables can only be parent nodes for the succeeding variables.

Several node ordering methods are developed, including domain knowledge-based ordering [15] and sampling for node ordering [13, 20]. Cooper and Herskovits [15] proposed the famous K2 learning algorithm, which exploits some constraints during the learning process, such as limiting the number of parent nodes and giving node order in advance. Although the given node order greatly reduces the search spaces, it is rather difficult to be derived from domain knowledge [15], especially when the number of nodes is large. Thus, we need to learn the node order from data.

In addition, given a particular order, [21] studied a simple and effective algorithm called Ordering-Based Search (OBS) for learning a BN structure, which greedily explores neighboring orders by swapping their position, thereby optimizing the BN's score. Subsequently, [22] proposed an Acyclic Selection OBS (ASOBS) that allows reverse edges if they do not generate a directed cycle. Moreover, it has been proved that ASOBS has a better performance than OBS in the analyzed dataset. However, ASOBS was conducted by uniformly sampling the space of the orders. Hence, [13] introduced a novel entropy-based sample approach that more effectively sampled from the order spaces. It only considers sampling the node order based on the entropy of each node. But according to the BIC metric, the node order depends on not only its entropy but also the joint entropy with the other node and its domain value. Moreover, the node order spaces in [13, 22] are $O(n!)$. So the algorithms are more random and unstable because of the applied sample methods.

Besides, a new BN structure learning algorithm was studied by finding the node order based on conditional independence test [23]. To reflect the cause and effect between variables, an efficient node order learning method was devised using a novel scoring function based on the conditional frequency [24]. By minimizing the inferential loss using a genetic algorithm, [25] designed a wrapper node order learning method. [26] proposed a mixed integer programming model based on the topological order to optimize the structure of the BN, the nodes of which have continuous values. However, these methods do not consider the node order from a viewpoint consistent with the frequently used BIC measurement, which not only reflects the knowledge of the data by fitting the degree of data and also simplifies the model, for learning the BN model.

Therefore, to avoid the aforementioned disadvantages of existing methods, we use the frequently adopted BIC score metric to learn the node order in $O(|V_1|! \cdot |V_2|! \cdot \cdots \cdot |V_q|!)$ order space, where $|V_1| + |V_2| + \cdots + |V_q| = n$; n is the node number in a BN, $|V_q|$ is the node number in node set V_q in which the node order belongs to q-th rank. Obviously, $O(|V_1|! \cdot |V_2|! \cdot \cdots \cdot |V_q|!) \ll O(n!)$, which makes the performance of the learning methods more stable and effective. Moreover, our node ranking technique can be efficiently computed. Therefore, it is easy to be integrated into the existing BN learning methods.

The rest of this paper is organized as follows. Section 2 briefly reviews Bayesian networks and the BIC score function. Section 3 describes our proposed node order learning algorithm based on the BIC score function. Section 4 provides the experimental results and their analysis. Section 5 gives the related work. Finally, some concluding remarks regarding this research are provided in Section 6.

2 Preliminaries

In this section, we review some preliminaries including Bayesian networks and a BIC score function for measuring a Bayesian network model.

2.1 Bayesian networks

Probabilistic reasoning is an important issue in the field of artificial intelligence. For a set of n random variables, the joint probability distribution requires the specification of $2^n - 1$ numbers. That is, the number of independent parameters is $2^n - 1$. To reduce the computational complexity, a Bayesian network (BN) [1] was proposed, the goal of which is to represent a joint distribution Pr over a set of random variables through the conditional independence among the variables. In general, the number of specifications or independent parameters is far smaller than $2^n - 1$.

A BN model is the combination of graph theory and probability theory. It is a probabilistic graphical model and can be expressed as $\mathcal{M} = (\mathcal{G}, \theta)$, where $\mathcal{G} = (V, E)$ denotes the DAG structure. The triple is as follows.

- First, $\mathrm{V} = \{X_1, X_2, \ldots, X_n\}$, it is a set of nodes or random variables.
- Second, $\mathrm{E} = \{(X_i, X_j) | 1 \leqslant i \neq j \leqslant n\}$, it is a directed edge set, where (X_i, X_j) denotes $X_i \rightarrow X_j$ in the DAG. Note that $< X_i, X_j >$ indicates an undirected edge between X_i and X_j.
- Finally, $\theta = \{\Pr(X_i | \Pi_{X_i})\}$ is the conditional probabilistic distribution set of each node conditioned by its parent nodes, where Π_{X_i} is the set of parent nodes of node X_i.

Thus, a BN can be formally expressed as follows:

$$\Pr(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} \Pr(X_i | \Pi_{X_i}), \tag{2}$$

which indicates that, given the parent set Π_{X_i}, each variable X_i is independent of its non-descendants in the BN model.

2.2 BIC score function

The structure of a BN is a prerequisite of probabilistic reasoning using a BN model. Therefore, it is necessary to learn the BN structure from the known dataset $\mathcal{D} = \{\mathrm{D}_1, \mathrm{D}_2, \ldots, \mathrm{D}_m\}$, where m denotes the sample number in $\mathcal{D}$. Assume that the data are complete and each variable X_i is a categorically random variable. Ω_{X_i} denotes the state space of a random variable X_i, $\Omega_{\Pi_{X_i}}$ denotes the state space of all parents of X_i, and we set $\Omega_{\emptyset} = 1$. The goal of structure learning is to find the best BN model $\mathcal{M} = (\mathcal{G}, \theta)$ that fits well with the data $\mathcal{D}$.

In this study, we take the frequently used BIC score function as a metric of a BN model, because it reflects the data knowledge and simplifies the model. Also the BIC score is decomposable, and it can be expressed as the score sum of each variable and its parent set. It can be written as follows:

$$\mathrm{BIC}(\mathcal{M}|\mathcal{D}) = \log \Pr(\mathcal{D}|\mathcal{M}) - \frac{d}{2} \log m$$

$$\Leftrightarrow \sum_{i=1}^{n} \mathrm{BIC}(X_i|\Pi_{X_i}) = \sum_{i=1}^{n}(\mathrm{L}(X_i|\Pi_{X_i}) - \lambda(X_i|\Pi_{X_i})). \quad (3)$$

where

- d is the number of independent parameters in the BN model $\mathcal{M}$, and
- $\mathrm{L}(X_i|\Pi_{X_i})$ represents the log-likelihood of X_i and its parent set Π_{X_i}, that is,

$$\mathrm{L}(X_i|\Pi_{X_i}) = \sum_{\pi\in\Omega_{\Pi_{X_i}}} \sum_{x\in\Omega_{X_i}} m_{x,\pi} \log \theta^{*}_{x|\pi}. \quad (4)$$

  Here, $\theta^{*}_{x|\pi}$ denotes the maximum likelihood estimate (MLE) of the conditional probability $\Pr(X_i = x|\Pi_{X_i} = \pi)$, which is written as $\frac{m_{x,\pi}}{m_\pi}$. m_π and $m_{x,\pi}$ represent the number of samples ($\Pi_{X_i} = \pi$) and ($X_i = x$ & $\Pi_{X_i} = \pi$) appear in $\mathcal{D}$, respectively. If π is null, then $m_\pi = m$ and $m_{x,\pi} = m_x$. If node X_i has no parents, then $\mathrm{L}(X_i) = \mathrm{L}(X_i|\emptyset)$.
- $\lambda(X_i|\Pi_{X_i})$ is the penalization of complexity for X_i and its parent set Π_{X_i}.

$$\lambda(X_i|\Pi_{X_i}) = \frac{\log m}{2}(|\Omega_{X_i}| - 1)|\Omega_{\Pi_{X_i}}|. \quad (5)$$

  Similarly, if $\Pi_{X_i} = \emptyset$, then $\lambda(X_i) = \lambda(X_i|\emptyset)$.

The higher the BIC score is, the better the BN model is, which means that it fits the data well. Thus, to maximize the score of each node in the learned model $\mathcal{M}$, the ultimate goal is to find the following:

$$\mathcal{M}^{*} = \arg\max_{\mathcal{M}} \mathrm{BIC}(\mathcal{M}|\mathcal{D}) = \arg\max_{\mathcal{M}} \sum_{i=1}^{n} \mathrm{BIC}(X_i|\Pi_{X_i}). \quad (6)$$

3 BIC-based node order learning

In this section, we address the BIC-based node order learning method, which is divided into four steps. First, we learn some undirected subgraphs based on the BIC measure. Second, all undirected subgraphs can be connected into one undirected subgraph according to the BIC scores. Third, we orient all edges in the entire undirected graph based on the BIC measure by orienting small graph with three nodes. Fourth, we rank the topology order of the DAG. Finally, we provide the pseudo-code of the BIC-based node order learning algorithm and analyse its time complexity.

3.1 Undirected subgraph learning

Usually, each node in a DAG has at least a directed edge that points to or points from the other node, unless the DAG is an unconnected graph. In this paper, we assume that the final learned DAG with n nodes should be a connected DAG. So in the first step, for each node $X_i (1 \leqslant i \leqslant n)$, we learn the single node that is most closely dependent on X_i from $\mathrm{V}\backslash X_i$. Here for each node X_i, we will try to find its best single parent $X_j (1 \leqslant j \neq i \leqslant n)$ using the BIC function, and thus we need to compute the following:

$$\Pi_{X_i} \leftarrow \arg \max_{X_j\in\Pi_{X_i} \text{ in } \mathcal{M}_{ij}} \mathrm{BIC}(\mathcal{M}_{ij}|\mathcal{D}), \quad (7)$$

where $\mathcal{M}_{ij}$ indicates that the DAG $\mathcal{G}_{ij}$ in $\mathcal{M}_{ij}$ has only one directed edge $X_j \to X_i$ for n variables. This means that for X_i, we search its best parent X_j from $n-1$ DAGs, in which each DAG has only one edge $X_j \to X_i$. The $n-1$ DAGs are as follows:

$$\mathcal{G}_{i1},\ \mathcal{G}_{i2},\ \ldots,\ \mathcal{G}_{ij},\ \ldots,\ \mathcal{G}_{in}\ (1 \leqslant j \neq i \leqslant n).$$

Theorem 1 For each variable X_i, finding its Π_{X_i} that meets Eq. (7) from $\mathcal{G}_{i1},\ \mathcal{G}_{i2},\ \ldots,\ \mathcal{G}_{ij},\ \ldots,\ \mathcal{G}_{in} (1 \leqslant j \neq i \leqslant n)$ is equivalent to the search of

$$\Pi_{X_i} \leftarrow \arg \max_{X_j\in\Pi_{X_i}} \mathrm{BIC}(X_i|\Pi_{X_i}),$$

between node X_i and any other one parent node X_j.

Proof For each X_i and any one node $X_j \in \Pi_{X_i} (1 \leqslant j \neq i \leqslant n)$, we have

$$\begin{aligned}\mathrm{BIC}(\mathcal{M}_{i1}|\mathcal{D}) = {} & \mathrm{BIC}(\mathcal{G}_{i1}|\mathcal{D}) \\ & = \mathrm{BIC}(X_i|X_1) + \mathrm{BIC}(X_1) + \mathrm{BIC}(X_2) \\ & \quad + \cdots + \mathrm{BIC}(X_{n'-1}) + \mathrm{BIC}(X_{n'}) \\ & = \mathrm{BIC}(X_i|X_1) + \text{constant1},\end{aligned} \quad (8)$$

where constant1 $= \mathrm{BIC}(\mathrm{X}_1) + \mathrm{BIC}(\mathrm{X}_2) + \cdots + \mathrm{BIC}(\mathrm{X}_{n'})$ and is a constant number. In addition, $X_{n'} \in V\backslash X_i$ and $n' = n - 1$.

$$\begin{aligned}\mathrm{BIC}(\mathcal{M}_{i2}|\mathcal{D}) = {} & \mathrm{BIC}(\mathcal{G}_{i2}|\mathcal{D}) \\ & = \mathrm{BIC}(X_1) + \mathrm{BIC}(X_i|X_2) + \mathrm{BIC}(X_2) \\ & \quad + \cdots + \mathrm{BIC}(X_{n'-1}) + \mathrm{BIC}(X_{n'}) \\ & = \mathrm{BIC}(X_i|X_2) + \text{constant1}.\end{aligned} \quad (9)$$

Similarly,

$$\begin{aligned}\mathrm{BIC}(\mathcal{M}_{ij}|\mathcal{D}) = {} & \mathrm{BIC}(\mathcal{G}_{ij}|\mathcal{D}) \\ & = \mathrm{BIC}(X_1) + \mathrm{BIC}(X_2) + \cdots \\ & \quad + \mathrm{BIC}(X_i|X_j) + \mathrm{BIC}(X_j) + \cdots \\ & \quad + \mathrm{BIC}(X_{n'-1}) + \mathrm{BIC}(X_{n'}) \\ & = \mathrm{BIC}(X_i|X_j) + \text{constant1},\end{aligned} \quad (10)$$

Further, we have the following:

$$\mathrm{BIC}(\mathcal{M}_{ij}|\mathcal{D}) \propto \mathrm{BIC}(X_i|X_j) \propto \mathrm{BIC}(X_i|\Pi_{X_i}).$$

Thus, we conclude the following:

$$\begin{aligned} & \Pi_{X_i} \leftarrow \operatorname*{argmax}_{X_j\in\Pi_{X_i} \text{ in } \mathcal{M}_{ij}} \mathrm{BIC}(\mathcal{M}_{\mathrm{ij}}|\mathcal{D}) \\ \Longleftrightarrow \quad & \Pi_{X_i} \leftarrow \operatorname*{argmax}_{X_j\in\Pi_{X_i}} \mathrm{BIC}(X_i|\Pi_{X_i}).\end{aligned} \quad (11)$$

□

According to the BIC score function shown in Eq. (3), we also have the following theorem.

Theorem 2 Let $X_i, X_j \in \mathrm{V}$, $\mathcal{G}_{ij}$ and $\mathcal{G}_{ji}$ be a DAG with only one directed edge $X_j \to X_i$ and $X_i \to X_j$ for n variables in $\mathcal{M}_{ij}$ and $\mathcal{M}_{ji}$, respectively. Then, for the same dataset $\mathcal{D}$, we have the following:

$$\mathrm{BIC}(\mathcal{M}_{ij}|\mathcal{D}) = \mathrm{BIC}(\mathcal{M}_{ji}|\mathcal{D}).$$

Proof Using the definition of information entropy, we have the following formula:

$$\begin{aligned}\mathrm{H}(X_i) &= -\log \sum_{x\in\Omega_{X_i}} \Pr(x)\log\Pr(x)\\ &= -\log \sum_{x\in\Omega_{X_i}} \frac{m_x}{m}\log\frac{m_x}{m},\end{aligned} \tag{12}$$

where x is the configuration of X_i. Because $\theta^*_x = m_x/m$, for any disjointed subsets $X_i, X_j \in \mathrm{V}$, using Eqs. (4) and (12), it is obvious that

$$m\mathrm{H}(X_i|X_j) = -\mathrm{L}(X_i|X_j) \tag{13}$$

as described in [14]. Moreover, we know that the information entropy is also defined as follows:

$$\mathrm{H}(X_i|X_j) = \mathrm{H}(X_i, X_j) - \mathrm{H}(X_j). \tag{14}$$

Further, based on Eqs. (3)–(5), and (10), we have

$$\begin{aligned}&\mathrm{BIC}(\mathcal{M}_{ij}|\mathcal{D}) = \mathrm{BIC}(\mathcal{G}_{ij}|\mathcal{D})\\ =\ &\mathrm{BIC}(X_1) + \cdots + \mathrm{BIC}(X_i|X_j) + \mathrm{BIC}(X_j) + \cdots + \mathrm{BIC}(X_{n''})\\ =\ &\mathrm{BIC}(X_i|X_j) + \mathrm{BIC}(X_j) + \text{constant2}\\ =\ &\mathrm{L}(X_i|X_j) - \lambda(X_i|X_j) + \mathrm{L}(X_j|\emptyset) - \lambda(X_j|\emptyset) + \text{constant2}\\ =\ &-m\mathrm{H}(X_i|X_j) - \frac{\log m}{2}|\Omega_{X_i} - 1||\Omega_{X_j}|\\ &-m\mathrm{H}(X_j) - \frac{\log m}{2}|\Omega_{X_j} - 1| + \text{constant2}\\ =\ &-m\mathrm{H}(X_i, X_j) + m\mathrm{H}(X_j) - m\mathrm{H}(X_j) - \frac{\log m}{2}|\Omega_{X_i}||\Omega_{X_j}|\\ &+\frac{\log m}{2}|\Omega_{X_j}| - \frac{\log m}{2}|\Omega_{X_j}| + \frac{\log m}{2} + \text{constant2}\\ =\ &-m\mathrm{H}(X_i, X_j) - \frac{\log m}{2}|\Omega_{X_i}||\Omega_{X_j}| + \frac{\log m}{2} + \text{constant2}\\ =\ &-m\mathrm{H}(X_i, X_j) - \frac{\log m}{2}|\Omega_{X_i}||\Omega_{X_j}| + \text{constant3},\end{aligned} \tag{15}$$

where constant2 $= \mathrm{BIC}(X_1) + \cdots + \mathrm{BIC}(X_{n''})$, $X_{n''} \in V\backslash\{X_i, X_j\}$, constant3 $= \frac{\log m}{2}$ + constant2.

Similarly,

$$\begin{aligned}&\mathrm{BIC}(\mathcal{M}_{ji}|\mathcal{D}) = \mathrm{BIC}(\mathcal{G}_{ji}|\mathcal{D})\\ =\ &-m\mathrm{H}(X_i, X_j) - \frac{\log m}{2}|\Omega_{X_i}||\Omega_{X_j}| + \text{constant3},\end{aligned} \tag{16}$$

Therefore, according to Eqs. (15) and (16), we have that

$$\mathrm{BIC}(\mathcal{M}_{ij}|\mathcal{D}) = \mathrm{BIC}(\mathcal{M}_{ji}|\mathcal{D}). \tag{17}$$

□

From Theorems 1 and 2, we can see that the best single parent node with the highest BIC score can be regarded as the closest dependent node. Thus, we can connect X_i and X_j but not distinguish the direction between them, meaning we will have a learned undirected graph $\mathrm{U_G}$ that reflects the undirected dependent relation between variables during this process.

In addition, to improve readability and understanding, we take a simple benchmark BN that can be found at http://www.bnlearn.com/bnrepository/ as a toy example to illustrate the process of the algorithm in this paper. The learning process includes four steps and the learned edges in each step are described in red.

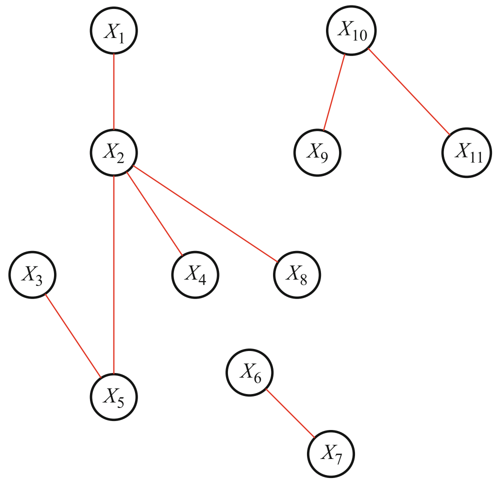

Fig. 1 The undirected subgraphs $\mathrm{U_G}$ that can be learned by the first step

For example, we select a benchmark BN including 11 nodes $(X_1, X_2, \ldots, X_{11})$ and we sample 10000 instances to learn the node order. In this step, we learn the undirected subgraphs $\mathrm{U_G}$ shown in Fig. 1.

3.2 Multiple undirected subgraph connecting

From Subsection 3.1, we can obtain n undirected edges by finding the best single parent node given each node X_i $(1 \leqslant i \leqslant n)$. However, there may be some overlapping in n undirected edges, which means a unconnected undirected graph may result in after the first step. Therefore, during the second step we check and connect $\mathrm{U_G}$ into a connected undirected graph $\mathrm{U_{GC}}$.

Let $\widetilde{n}$ be the number of different or non-overlapping undirected edges and k be the number of unconnected undirected subgraphs in $\mathrm{U_G}$. If $\widetilde{n} < n - 1$, then $\mathrm{U_G}$ is an unconnected undirected graph consisting of some undirected subgraphs. The number of variables n may be even or odd, and the relation between $\widetilde{n}$ and k is shown as Table 1 in detail.

From Table 1, when n is even, the range of $\widetilde{n}$ is $[\frac{n}{2}, n]$ and the range of k is $[1, \frac{n}{2}]$; when n is an odd, the range of $\widetilde{n}$ is $[\lceil\frac{n}{2}\rceil, n]$ and the range of k is $[1, \lfloor\frac{n}{2}\rfloor]$. To summarise, n is even or odd, and we have $\widetilde{n} \in [\lceil\frac{n}{2}\rceil, n]$ and $k \in [1, \lfloor\frac{n}{2}\rfloor]$. In general, $\widetilde{n} + k = n$, only there are no overlapping undirected edges in $\mathrm{U_G}$, $\widetilde{n} + k = n + 1$.

Table 1 The relation between $\widetilde{n}$ and k

	n is an even		n is an odd	
	$\widetilde{n}$	k	$\widetilde{n}$	k
1	$\frac{n}{2}$	$\frac{n}{2}$	$\lceil\frac{n}{2}\rceil$	$\lfloor\frac{n}{2}\rfloor$
2	$\frac{n}{2}+1$	$\frac{n}{2}-1$	$\lceil\frac{n}{2}\rceil+1$	$\lfloor\frac{n}{2}\rfloor-1$
3	$\frac{n}{2}+2$	$\frac{n}{2}-2$	$\lceil\frac{n}{2}\rceil+2$	$\lfloor\frac{n}{2}\rfloor-2$
4	…	…	…	…
5	$\frac{n}{2}+(\frac{n}{2}-1)$	1	$\lceil\frac{n}{2}\rceil+(\lfloor\frac{n}{2}\rfloor-1)$	1
6	$\frac{n}{2}+\frac{n}{2}$	1	$\lceil\frac{n}{2}\rceil+\lfloor\frac{n}{2}\rfloor$	1
n is an even or odd	$\widetilde{n} \in [\lceil\frac{n}{2}\rceil, n]$		$k \in [1, \lfloor\frac{n}{2}\rfloor]$	

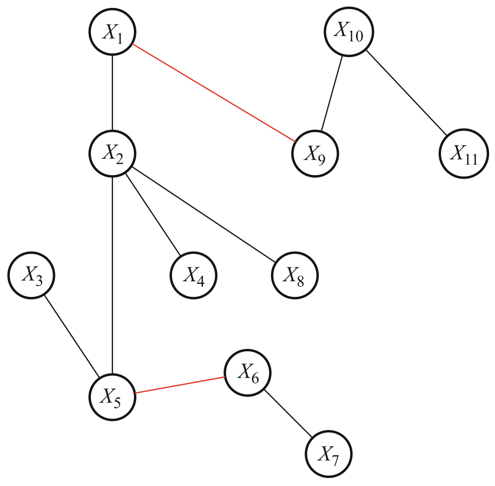

Fig. 2 The single undirected graph U_{GC} that can be connected by the second step

Further, we address how to connect some unconnected undirected subgraphs into a connected undirected graph when $\widetilde{n} < n-1$. Let V_k be the variable set of the kth undirected subgraph. When $k \geqslant 2$, we search the highest BIC score of $\mathcal{G}_{ij}$ in U_G, where $i \in V_k$ and $j \in V \setminus V_k$. We then connect X_i and X_j using an undirected edge, meanwhile, let $k = k-1$. We can loop this process until $k = 1$. Thus, after the second step we obtain U_{GC}, which is a connected undirected graph. Continuing with the previous example in Fig. 1, the connected undirected graph U_{GC} is shown in Fig. 2.

3.3 Orienting a connected undirected graph

In this step, we need to consider orienting the direction of each edge in U_{GC}. A large graph is composed of many small graphs of three nodes, written as $\mathcal{G}_s$, which can accurately represent a serial-link (e.g., $X_1 \to X_2 \to X_3$), divergent-link (e.g., $X_2 \to X_1$ & $X_2 \to X_3$), and collision-link (e.g., $X_1 \to X_2 \leftarrow X_3$, which is also called a V-structure) relationships among the variables in a large graph. Hence, we discuss the direction in U_{GC} by orienting an undirected graph in any small graph of three nodes $\mathcal{G}_s$.

For any $\mathcal{G}_s$, we know they can generate 25 DAGs. There are four different cases in these DAGs $\mathcal{G}_s$ by the number of edges, which may be zero, one, two, or three. Assume that $\mathcal{G}_s$ have three nodes: X_1, X_2, and X_3. The DAG $\mathcal{G}_s$ with zero edges is as shown in Fig. 3; the DAGs $\mathcal{G}_s$ with one edge are as shown in Fig. 4; the DAGs $\mathcal{G}_s$ with two edges are as shown in Fig. 5; and the DAGs $\mathcal{G}_s$ with three edges are as shown in Fig. 6.

When considering the orientation of the directions of undirected edges in U_{GC} by $\mathcal{G}_s$, we have the following theorem.

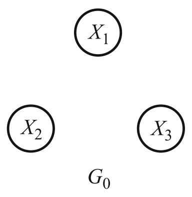

Fig. 3 DAG $\mathcal{G}_0$, which has zero edges

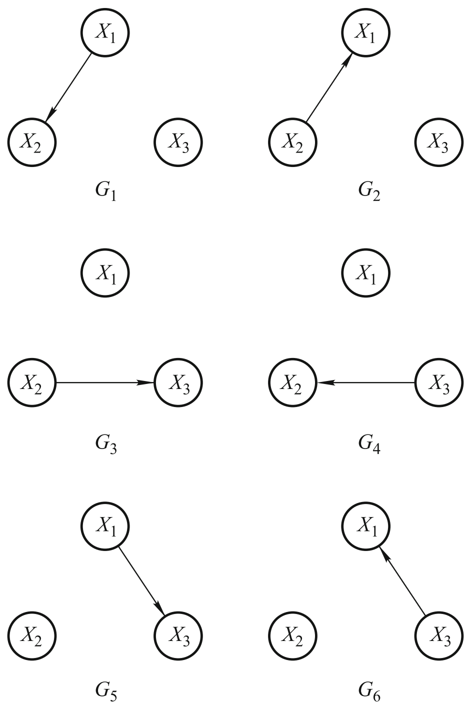

Fig. 4 DAGs $\mathcal{G}_1 \sim \mathcal{G}_6$, all of which have one edge

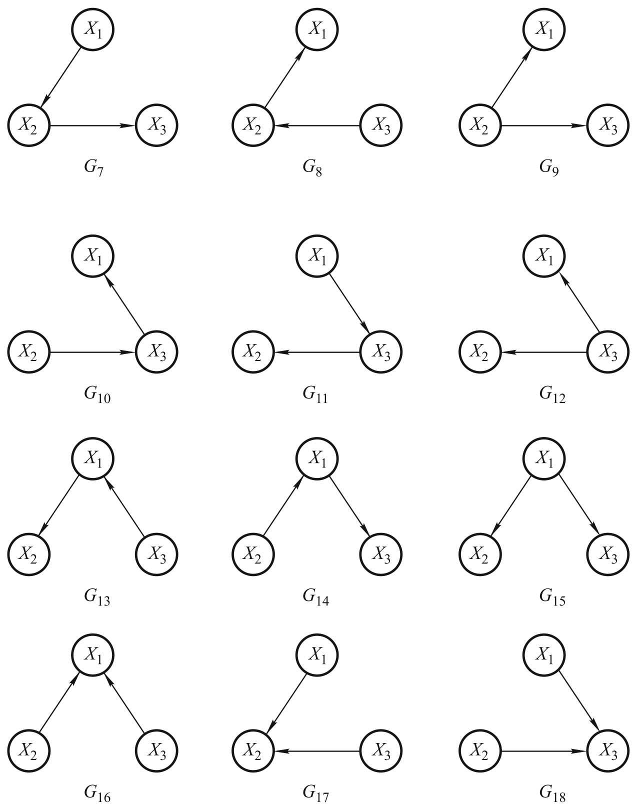

Fig. 5 DAGs $\mathcal{G}_7 \sim \mathcal{G}_{18}$, all of which have two edges

Theorem 3 For 25 DAGs consisting of three nodes, we only need to consider orienting the DAGs in these cases that have two undirected edges in U_{GC}; that is, only the 12 cases that are from $\mathcal{G}_7$ to $\mathcal{G}_{18}$ in Fig. 5 need to be considered.

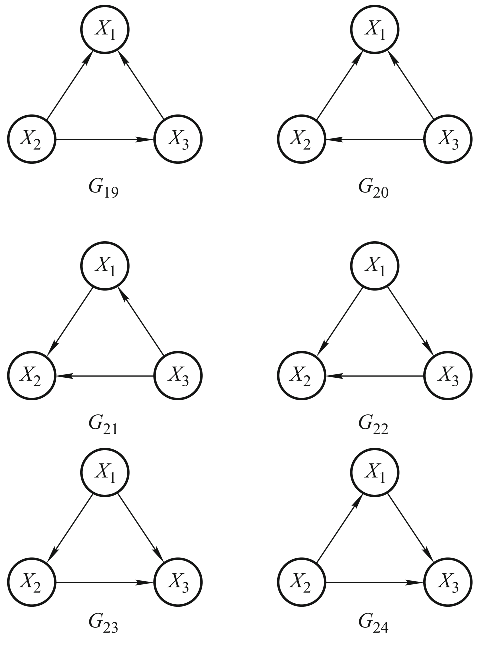

Fig. 6 DAGs $\mathcal{G}_{19} \sim \mathcal{G}_{24}$, all of which have three edges

Table 2 Six different BIC scores for 12 DAGs $\mathcal{G}_s$

Cases	DAGs with equal BIC
1	$\text{BIC}(\mathcal{G}_7) = \text{BIC}(\mathcal{G}_8) = \text{BIC}(\mathcal{G}_9)$
2	$\text{BIC}(\mathcal{G}_{10}) = \text{BIC}(\mathcal{G}_{11}) = \text{BIC}(\mathcal{G}_{12})$
3	$\text{BIC}(\mathcal{G}_{13}) = \text{BIC}(\mathcal{G}_{14}) = \text{BIC}(\mathcal{G}_{15})$
4	$\text{BIC}(\mathcal{G}_{16})$
5	$\text{BIC}(\mathcal{G}_{17})$
6	$\text{BIC}(\mathcal{G}_{18})$

Proof First, because there are $n-1$ connected undirected edges in U_{GC} after the second step, it is impossible to incur a cycle for any $\mathcal{G}_s$ that consists of three nodes, which means there are no cases in which $\mathcal{G}_s$ has three edges. Second, because U_{GC} is a connected undirected graph, for any one edge, it is always connected by at least one edge. In addition, we can always find another adjacent edge that gives the two edges a common node. This means there are no cases in which $\mathcal{G}_s$ has zero or one edge. Therefore we do not need to consider $\mathcal{G}_s$ with zero edges, one edge, or three edges. We only need to consider orienting the undirected edges in U_{GC} using $\mathcal{G}_s$ with two edges, meaning we only need to consider 12 DAGs from $\mathcal{G}_7$ to $\mathcal{G}_{18}$. □

Theorem 4 There are 6 different BIC score cases shown in Table 2 for the 12 DAGs $\mathcal{G}_s$ with two edges from $\mathcal{G}_7$ to $\mathcal{G}_{18}$, as shown in Fig. 5. Here, some DAGs are equivalent according to the BIC metric.

Proof These cases shown in Table 2 can be directly deduced from the BIC score function. □

As shown in Fig. 5, according to Theorem 4 and Table 2, we have the following conclusions.

- For two edges having a common node X_2, there are four cases, namely $\mathcal{G}_7$, $\mathcal{G}_8$, $\mathcal{G}_9$, and $\mathcal{G}_{17}$, which denote exactly a serial-link relationship ($X_1 \to X_2 \to X_3$ or $X_3 \to X_2 \to X_1$), a divergent-link relationship ($X_2 \to X_1$ & $X_2 \to X_3$), and a collision-link relationship ($X_1 \to X_2 \leftarrow X_3$, which is also called a V-structure). Similarly, for two edges having common nodes X_3 and X_1, the same cases occur.
- Moreover, $\text{BIC}(\mathcal{G}_7) = \text{BIC}(\mathcal{G}_8) = \text{BIC}(\mathcal{G}_9)$, $\text{BIC}(\mathcal{G}_{10}) = \text{BIC}(\mathcal{G}_{11}) = \text{BIC}(\mathcal{G}_{12})$ and $\text{BIC}(\mathcal{G}_{13}) = \text{BIC}(\mathcal{G}_{14}) = \text{BIC}(\mathcal{G}_{15})$, which shows that, for any small graph with three nodes $\mathcal{G}_s$ having two edges with a common node, its serial-link and divergent-link graphs always have equivalent BIC scores and only its collision-link graph (V-structure) may have different BIC scores.

Therefore, for any three nodes X_i, X_j, X_k, if X_i is a common node connecting the other two nodes, there are two undirected edges $< X_i, X_j >$ and $< X_i, X_k >$, which is denoted as a undirected small graph $\overline{\mathcal{G}_s^{ijk}}$. Let $\mathcal{G}_s^{ijk}$ be a DAG oriented for $\overline{\mathcal{G}_s^{ijk}}$, we can also indicate that $\mathcal{G}_s^{ijk}$ has four cases, as shown in Fig. 7. With serial-link DAGs ($\mathcal{G}_s^{ijk}-1$ and $\mathcal{G}_s^{ijk}-2$) and a divergent-link DAG ($\mathcal{G}_s^{ijk}-3$) having equivalent BIC scores, it is easy to conclude that

$$\text{BIC}(\mathcal{G}_s^{ijk}-1) = \text{BIC}(\mathcal{G}_s^{ijk}-2) = \text{BIC}(\mathcal{G}_s^{ijk}-3). \tag{18}$$

Further, for any $\overline{\mathcal{G}_s^{ijk}}$, we can see which case $\mathcal{G}_s^{ijk}$ belongs to in Fig. 7. In fact, by simply calculating the corresponding BIC scores of $\text{BIC}(\mathcal{G}_s^{ijk}-1)$ and $\text{BIC}(\mathcal{G}_s^{ijk}-4)$, we have the following orienting rules:

- **Rule 1:** If $\text{BIC}(\mathcal{G}_s^{ijk}-4) > \text{BIC}(\mathcal{G}_s^{ijk}-1)$, it means that there is a collision-link relationship or V-structure in $\mathcal{G}_s^{ijk}$ and it is easy to orient the two edges, so we orient their directions as $X_j \to X_i$ and $X_k \to X_i$.
- **Rule 2:** If $\text{BIC}(\mathcal{G}_s^{ijk}-4) \leqslant \text{BIC}(\mathcal{G}_s^{ijk}-1)$, it means that there is a serial-link or divergent-link relationship in $\mathcal{G}_s^{ijk}$, so it isn't easy to orient their directions and we list the two undirected edges $< X_i, X_j >$ and $< X_i, X_k >$ into a list $\mathcal{L}$.

When all undirected edges in U_{GC} are visited, some edges are oriented and the others are not oriented. The graph is a partial undirected graph, which we denote as PU_{GC}. Continuing with Fig. 2, the partial undirected graph PU_{GC} is shown in Fig. 8.

Next, according to the principle of not raising V-structure, we orient each undirected edge $< X_i, X_j >$ in list $\mathcal{L}$ by its

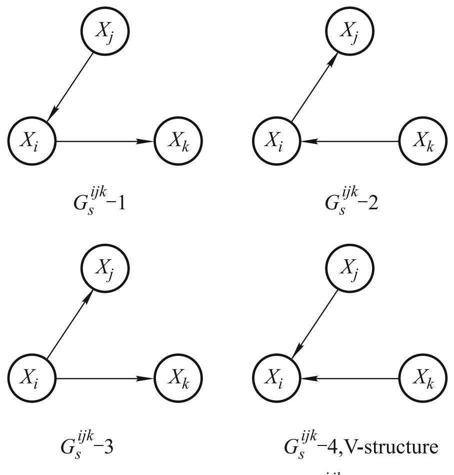

Fig. 7 Four DAGs in $\mathcal{G}_s^{ijk}$

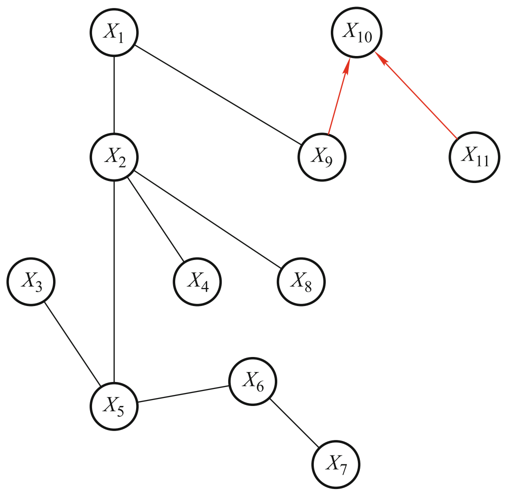

Fig. 8 The partial undirected graph $\mathrm{PU_{GC}}$ that can be oriented by Rule 1 and Rule 2 in the third step

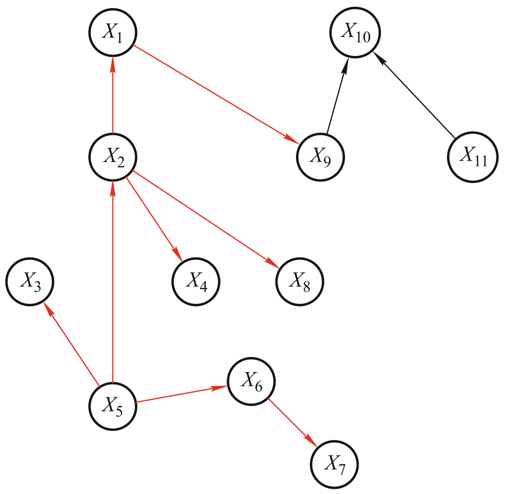

Fig. 9 The learned DAG that can be oriented by Rule 2.1, Rule 2.2 and Rule 2.3 in the third step

neighbour edges oriented in $\mathrm{PU_{GC}}$. This process is repeated using the following rules until $\mathcal{L} = \emptyset$ or all edges are oriented in $\mathrm{PU_{GC}}$.

- **Rule 2.1:** If a directed edge points to X_i (e.g., $\to X_i$) in $\mathrm{PU_{GC}}$, then we can set a serial-link relationship (e.g., $\to X_i \to X_j$) that doesn't change the BIC score value, thus we have $X_i \to X_j$ and $\mathcal{L} = \mathcal{L}\backslash\{< X_i, X_j >\}$.
- **Rule 2.2:** If a directed edge points from X_i and another points to X_i (e.g., $\to X_i \to$) in $\mathrm{PU_{GC}}$, then we can also set a serial-link relationship (e.g., $\to X_i \to X_j$), thus, we also have $X_i \to X_j$ and $\mathcal{L} = \mathcal{L}\backslash\{< X_i, X_j >\}$.
- **Rule 2.3:** If a directed edge points from X_i and another points from X_i (e.g., $\leftarrow X_i \to$) in $\mathrm{PU_{GC}}$, then we can set a serial-link or divergent-link relationship that doesn't change the BIC score value. To not increase the probability of generating V-structure, we set $X_j \to X_i$ and $\mathcal{L} = \mathcal{L}\backslash\{< X_i, X_j >\}$.
- **Rule 2.4:** If no directed edge connects to node X_i, then we consider its neighbor node X_j using the same Rules (Rule 2.1–Rule 2.3) as considering node X_i.

Finally, we obtain a DAG $\mathcal{G}$ after the third step. Continuing with Fig. 8, the learned DAG by Rule 2.1, Rule 2.2 and Rule 2.3 is shown in Fig. 9.

3.4 Ranking node order from DAG

For a DAG, to rank the order of nodes, we first search the nodes whose in-degree is zero, and sort them in the first ranking order. If several nodes have the same in-degree, then they have the same ranking order. Next, we delete these nodes and the corresponding edges and then search for those nodes whose in-degree is zero and sort them in the second ranking order. This process is looped until all nodes and edges are deleted. The loop number is the number of maximum rank. After the fourth step, we can obtain the node order or topological order.

To summarize, there are $|V_1|! * |V_2|! \cdots * |V_q|!$ node orders, where q is the number of maximum rank, $|V_q|$ is the node number in node set V_q. Clearly, the search space of the node order is dramatically reduced by comparing with the number of random node orders. That is, $|V_1|! * |V_2|! \cdots * |V_q|! \ll |V|!$, where n is the node number in a BN, $|V_1| + |V_2| + \cdots + |V_q| = n$, $|V| = n$.

Continuing with Fig. 9, the node order is $[\underline{X_5, X_{11}}, \underline{X_2, X_3, X_6}, \underline{X_1, X_4, X_7, X_8}, \underline{X_9}, \underline{X_{10}}]$, where $\underline{V_q}$ denotes the node set of q-th rank, and the nodes in node set $\overline{V_q}$ have the same rank q. Here, $q = 5$. In this example, X_5 and X_{11} have the same rank (i.e., the first rank V_1); X_2, X_3 and X_6 belong to the same rank V_2, and so on. In the same rank, we can randomly sample their order when learning the BN structure.

Thus, there are 288 ($|V_1|! * |V_2|! \cdots * |V_5|! = 2! \cdot 3! \cdot 4! \cdot 1! \cdot 1!$) node orders, it can easily be concluded that $288 \ll 11!$ in this example.

3.5 Node-order learning algorithm

From the above four steps, The pseudocode of the node order learning algorithm from the complete data $\mathcal{D}$ is given in Algorithm 1.

3.6 Time complexity analysis

In Algorithm 1, there are four steps used to learn the node order ρ from the data; hence, we analyse the time complexity in a step-by-step manner. Let n be the node number, m be the sample number, k be the number of unconnected undirected subgraphs in $\mathrm{U_G}$, and $|V_k|$ be the node number of the k-th undirected subgraph in $\mathrm{U_G}$. In addition, $|V\backslash V_k|$ is the number of variables that are not in V_k but are in V.

In the first step (from lines 2 through 6), for each node, we need to search its most dependent single node by computing $n-1$ $\mathrm{BIC}(\mathcal{M}_{ij})$. Because there are n nodes, we need to compute $n(n-1)$ $\mathrm{BIC}(\mathcal{M}_{ij})$. Moreover, according to the BIC score function,

$$\mathrm{BIC}(\mathcal{M}_{ij}) = \sum_i^n (-\mathrm{mH}(X_i|X_j) - \frac{\log m}{2}(|\Omega_{X_i}| - 1)|\Omega_{X_j}|). \quad (19)$$

the time complexity is related to m and $(|\Omega_{X_i}|-1)|\Omega_{X_j}|$; however, $(|\Omega_{X_i}|-1)|\Omega_{X_j}| \ll m$, and thus the time complexity is $O(n^2m)$ in this step.

In the second step (from lines 8 through 14), there are k unconnected undirected subgraphs, where k is at most $\lfloor \frac{n}{2} \rfloor$. For each k, we need to search the maximum BIC from $|V_k| * |V\backslash V_k|$

Algorithm 1 Node-order learning algorithm

```
Input: Data $\mathcal{D}$
Output: Node order $\rho$
 1: The first step: obtain $\mathrm{U_G}$
 2: Undirected edge set $\overline{E} = \emptyset$; Directed edge set $E = \emptyset$;
 3: for $i = 0$ to $n$ do
 4:     $\Pi_{X_i} \leftarrow \arg\max_{X_j \in \Pi_{X_i} \text{ in } \mathcal{M}_{ij}} \mathrm{BIC}(\mathcal{M}_{ij}|\mathcal{D})$;
 5:     $\overline{E} = \overline{E} \cup \{< X_i, X_j >\}$;
 6: end for
 7: The second step: obtain $\mathrm{U_{CG}}$
 8: $\widetilde{n} = |\overline{E}|$;
 9: if $\widetilde{n} < n - 1$ then
10:     for $k = n - \widetilde{n}$ to 2 do
11:         $< X_i, X_j > \leftarrow \arg\max_{<X_i,X_j>} \mathrm{BIC}(\mathcal{M}_{ij})$ in $\mathrm{U_G}$, where $X_i \in V_k$ and
            $X_j \in V \backslash V_k$;
12:         $\overline{E} = \overline{E} \cup \{< X_i, X_j >\}$; $k = k - 1$;
13:     end for
14: end if
15: The third step: obtain DAG $\mathcal{G}$
16: $\overline{\mathcal{G}_s} \leftarrow$ all small graphs of three nodes with two edges in $\mathrm{U_{GC}}$;
17: Let any three nodes $X_i, X_j, X_k$, $\overline{\mathcal{G}_s^{ijk}} \in \overline{\mathcal{G}_s}$;
18: $\mathcal{G}_s^{ijk} \leftarrow$ the DAG in which the edges in $\overline{\mathcal{G}_s^{ijk}}$ have been oriented;
19: Undirected edge List $\mathcal{L} = \emptyset$;
20: for each $\overline{\mathcal{G}_s^{ijk}}$ in $\overline{\mathcal{G}_s}$ do
21:     compute $\mathrm{BIC}(\mathcal{G}_s^{ijk} - 1)$ and $\mathrm{BIC}(\mathcal{G}_s^{ijk} - 4)$;
22:     if $\mathrm{BIC}(\mathcal{G}_s^{ijk} - 1) < \mathrm{BIC}(\mathcal{G}_s^{ijk} - 4)$ then
23:         $E = E \cup \{(X_j, X_i), (X_k, X_i)\}$;
24:     else
25:         $\mathcal{L} = \mathcal{L} \cup \{< X_i, X_j >, < X_i, X_k >\}$;
26:     end if
27: end for
28: while $\mathcal{L} \neq \emptyset$ do
29:     for each $< X_i, X_j >$ in $\mathcal{L}$ do
30:         if (a directed edge points to $X_i$) $\vee$ (a directed edge points from $X_i$
            and another directed edge points to $X_i$) then
31:             $X_i \rightarrow X_j$; $\mathcal{L} = \mathcal{L} \backslash \{< X_i, X_j >\}$; $E = E \cup \{(X_i, X_j)\}$;
32:         end if
33:         if (a directed edge points from $X_i$) $\wedge$ (another directed edge points
            from $X_i$) then
34:             $X_j \rightarrow X_i$; $\mathcal{L} = \mathcal{L} \backslash \{< X_i, X_j >\}$; $E = E \cup \{(X_j, X_i)\}$;
35:         end if
36:         if no directed edge connects to $X_i$ then
37:             considering $X_j$ in the same way as node $X_i$;
38:         end if
39:     end for
40: end while
41: The fourth step: obtain node order $\rho$
42: $\mathcal{G} \leftarrow (E, V)$; $V' \leftarrow V$; $i = 0$; $\rho = \emptyset$;
43: Building the in-degree of each node using directed edges in $\mathcal{G}$;
44: while $V' \neq \emptyset$ do
45:     $i = i + 1$;
46:     Find the node set $V'_i$ in which all nodes have a zero in-degree;
47:     $\rho = \rho \cup V'_i$; $V' \leftarrow V' \backslash V'_i$
48: end while
49: Return $\rho$;
```

$\mathrm{BIC}(\mathcal{M}_{ij})$, where $|V_k| + |V \backslash V_k| = n$, and thus we need to search $k * |V_k| * |V \backslash V_k|$ $\mathrm{BIC}(\mathcal{M}_{ij})$. Assume that the largest number is n for $|V_k|$ and $|V \backslash V_k|$, although the number is actually less than n. Thus, the search time complexity is $O(n^3 m)$ in this step.

For the third step (from lines 16 through 40), the two loops need a long computational time and the other lines need a constant time. In terms of the **for** loop from lines 20 through 27, because $\overline{\mathcal{G}_s}$ is a set of all small graphs of three nodes with two edges in $\mathrm{U_{GC}}$, let $n - 1$ be the edge number in $\mathrm{U_{GC}}$, and there are $\binom{n-1}{2}$ small graphs in $\overline{\mathcal{G}_s}$ that need to compute their BIC scores. Thus, the time complexity is $O(n^2 m)$ in this loop. For the **while** loop from lines 28 through 40, there are at most $n - 1$ undirected edges in $\mathcal{L}$, for each undirected edge $< X_i, X_j >$, and when we search the adjacent directed edges of node X_i or X_j, $n - 1$ edges at most are required. Thus, the time complexity is $O(n^2)$ in this loop; therefore, the time complexity is $O(n^2 m)$ during the third step.

In the fourth step (from lines 42 through 48), building the in-degree of each node in line 43 requires $n - 1$ iterations, because $n - 1$ directed edges are in $\mathcal{G}$. For the **while** loop from lines 44 through 48, because there are n nodes, n iterations are required. Thus, the time complexity during the fourth step is $O(n)$.

Therefore, the entire time complexity is $O(n^3 m)$ for our proposed learning Algorithm 1, that is, the worst time complexity of Algorithm 1 is linear with respect to the sample number and polynomial with respect to the node number.

4 Experiments

In this section, we assess the performance of the proposed node order learning method, and compare it with the other existing methods for ranking the node order by taking the node order into the popular K2 algorithm to learn the BN structure. Clearly, our proposed method can be used into most score-&-search-based structure learning methods. The evaluation metrics include BIC score, KL divergence, precision, recall, F1 and run time for the learned BN, and the mean absolute error (mae) [13] for the inference results based on the learned BN.

4.1 Experiment setting and datasets

In this experiment, all experiments were implemented in Python 3. We first learn the node order from the data, and then take them into K2 algorithm to learn the structure of BN from the same data. Here, we set the maximum number of parents to 3 when $n \leqslant 10$; to 4 when $10 < n \leqslant 30$; and to 5 when $n \geqslant 30$.

To conduct the experiments, we use nine synthetic datasets generated from nine benchmark BNs, which can be found at http://www.bnlearn.com/bnrepository/, respectively. The basic properties of each BN include the BN names(abbreviated as BNs), the node number(abbreviated as nodes), the edge number(abbreviated as edges), the parameter number(abbreviated as parameters), the instance number and the sample times on each BN (abbreviated as instances(times)). For the different BNs, we generate 10,000 instances for 5 times, 3 times, or once, using logic sample methods, which are denoted as 10000(5), 10000(3), and 10000(1), respectively. They all are described in Table 3.

4.2 Evaluation metrics

To verify the performance of our proposed node order algorithm, six evaluation metrics are used for comparison and analysis of the experimental results between our algorithm and the other algorithms. The evaluation index are described as follows.

Table 3 Description of nine synthetic benchmark BN generated datasets

No.	BNs	Nodes	Edges	Parameters	Instances(times)
1	Cancer	5	4	10	10000(5)
2	Earthquake	5	4	10	10000(5)
3	Survey	6	6	21	10000(5)
4	Asia	8	8	18	10000(5)
5	Sachs	11	17	178	10000(5)
6	Alarm	37	46	509	10000(5)
7	Hailfinder	56	66	2656	10000(5)
8	Andes	223	338	1157	10000(3)
9	Pigs	441	592	5618	10000(1)

(1) **BIC score** This is described in Section 2.

(2) **KL divergence** KL divergence is also called relative entropy and is a measure between different probability distributions of learned BNs and benchmark BNs. It can be formalized into the following formula.

$$\mathrm{D_{KL}(S\|T)} = \sum_{\mathrm{x}\in\Omega_{\mathrm{V}}} \mathrm{S(x)\log(\frac{S(x)}{T(x)})}. \tag{20}$$

Here, S and T are two different discrete probability distributions, S is the joint probability distribution of the learned BN, and T is the joint probability distribution of the benchmark BN.

(3) **Precision** We can take the presence or absence of each edge in a learned BN as a binary classification problem herein, and thus the precision is the number of correct edges present divided by the total number of edges in the learned BN.

$$\text{precision} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}. \tag{21}$$

Here, TP is the number of edges correctly present in the learned BN, and FP is the number of edges incorrectly present in the learned BN.

(4) **Recall** The recall is the number of correct edges present in the learned BN divided by the total number of edges present in the benchmark BN.

$$\text{recall} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}. \tag{22}$$

Here, TP is described similarly to Eq. (21), and FN is the number of incorrectly absent edges in the learned BN.

(5) **F1** F1 is the harmonic mean of the recall and precision.

$$\mathrm{F1} = 2\cdot\frac{\text{recall} * \text{precision}}{\text{recall} + \text{precision}}. \tag{23}$$

(6) **Mean Absolute Error (mae)** This can be used to measure the inference results between those based on the learned BN and those based on the benchmark BN.

$$\text{mae} = \frac{1}{c}\sum_i |\Pr_i(q) - \widehat{\Pr}_i(q)|. \tag{24}$$

Here, c denotes the total number of queries based on the BN inference, $\Pr_i(q)$ is the computed posterior probability of the ith query based on the benchmark BNs, and $\widehat{\Pr}_i(q)$ is the computed posterior probability of the ith query based on the learned BNs.

(7) **Run time(s)** Here the run time(s) denotes that how many seconds the k2 algorithm is conducted based on different node orders.

4.3 Learning results based on synthetic data

In this subsection, we first verify the performance of our proposed method based on the BN learning results by taking the learned node order into the K2 algorithm, and by comparing our proposed method (abbreviated as BIC-based) with the three existing methods. They are the entropy-based node order sampling method [13] (abbreviated as Entropy-based), random sample node order method (abbreviated as Random), and topological order of benchmark BN (abbreviated as Benchmark).

Note that we can obtain the node order space of DAG structure of a BN by the fourth step described in Subsection 3.4. Benchmark algorithm denotes that we use the fourth step of our algorithm to obtain the node order space of benchmark DAG of a BN, thus we sample one node order from the node order space of benchmark DAG as the relative optimal order. Our BIC-based algorithm denotes that we learn the DAG of a BN by the first three steps of our algorithm and then obtain one node order by the fourth step described in Subsection 3.4, so our learned DAG may not be the same good as the corresponding benchmark DAG, which makes that its node order may not be relative optimal order as that of benchmark DAG.

The comparison results using the K2 algorithm with the four different node order methods are shown in Table 4. Specifically, the precision, recall, F1 and run time are used to measure the learned DAG structures $\mathcal{G}$; the BIC score and KL divergence are used to measure the learned DAG structures and parameters $(\mathcal{G},\theta)$, that is, they are used to measure the BN model $\mathcal{M}$.

The BIC score is the trade-off between the likelihood function for data and the complexity of the network structure. The KL divergence is used to measure the joint probability distribution. The smaller the KL divergence is, the better the learned BN. Note that "mean±sd" denotes the mean and standard deviation of the learned results of three times or five times from the 10,000 different instances for each evaluation metric. For each dataset, the best results in each metric are shown in bold in Table 4.

From Table 4, we can conclude the following.

(1) According to the mean of the learned DAG structure of a BN based on the precision, recall, and F1 metrics, we obtained the best mean results except for the Hailfinder BN when applying the benchmark node order method. The second best results are from our proposed BIC-based node order, with the exception of the Sachs dataset.
The reason for this is that the benchmark Sachs BN is an unconnected DAG and includes two subDAGs, but we connected the unconnected graph using the second step of our method during the learning from the Sachs dataset.
In particular, our method has the same best results for the Canner BN and Earthquake BN as those of the benchmark method, meaning the DAGs are completely learned in the Canner and Earthquake datasets using our method.
Moreover, for the Hailfinder BN, our proposed method is

Table 4 Comparison results of the learned BNs using K2 algorithm with four different node order methods

Dataset	Methods	BIC (mean±sd)	KL divergence (mean±sd)	Precision (mean±sd)	Recall (mean±sd)	F1 (mean±sd)	run time(s) (mean±sd)
	Entropy-based	**–3152.694**±3036.858	0.022±0.007	0.350±0.082	0.300±0.100	0.161±0.090	**0.000±0.000**
Cancer	Random	–12418.956±3087.023	0.023±0.020	0.510±0.258	0.450±0.187	0.236±0.213	**0.000±0.000**
10000	Benchmark	–11949.743±**44.769**	**0.018±0.000**	**1.000±0.000**	**1.000±0.000**	**0.500±0.000**	**0.000±0.000**
	BIC-based	–11949.743±**44.769**	**0.018±0.000**	**1.000±0.000**	**1.000±0.000**	**0.500±0.000**	**0.000±0.000**
	Entropy-based	**–3715.421**±623.309	0.046±0.002	0.347±0.187	0.550±0.245	0.212±0.216	**0.000±0.000**
Earthquake	Random	–4313.796±748.221	0.032±0.025	0.211±0.079	0.400±0.123	0.137±0.093	**0.000±0.000**
10000	Benchmark	–4230.709±**114.87**	**0.027±0.000**	**1.000±0.000**	**1.000±0.000**	**0.500±0.000**	**0.000±0.000**
	BIC-based	–4230.709±**114.87**	**0.027±0.000**	**1.000±0.000**	**1.000±0.000**	**0.500±0.000**	**0.000±0.000**
	Entropy-based	**–19944.035**±6427.984	0.009±**0.005**	0.493±0.168	0.400±0.133	0.220±0.148	0.800±0.400
Survey	Random	–21896.582±5782.218	0.010±0.006	0.500±0.217	0.367±0.125	0.211±0.160	0.200±0.400
10000	Benchmark	–22883.773±**110.284**	0.008±0.007	**1.000±0.000**	**0.967±0.067**	**0.491±0.036**	**0.000±0.000**
	BIC-based	–20918.311±3917.798	**0.006**±0.005	0.813±0.016	0.733±0.082	0.385±0.052	0.400±0.490
	Entropy-based	–24375.558±2215.956	0.028±0.020	0.261±0.163	0.350±0.166	0.148±0.166	**1.400±0.490**
Asia	Random	–24317.325±2955.781	0.022±0.014	0.290±0.163	0.375±0.177	0.163±0.170	1.600±0.490
10000	Benchmark	**–21991.124±275.301**	**0.001±0.000**	**1.000±0.000**	**0.925±0.061**	**0.480±0.033**	**1.400±0.490**
	BIC-based	–22981.3366±1974.582	0.019±0.036	0.858±0.226	0.825±0.100	0.418±0.169	**1.400±0.490**
	Entropy-based	–79707.263±1075.057	0.119±0.040	0.333±0.134	0.471±0.162	0.186±0.118	13.400±6.344
Sachs	Random	–75285.587±1882.341	0.154±0.059	0.476±0.160	0.518±0.155	0.248±0.157	13.000±6.841
10000	Benchmark	–78212.950±**191.800**	**0.075±0.010**	**1.000±0.000**	**1.000±0.000**	**0.500±0.000**	**10.000**±5.060
	BIC-based	**–74844.417**±2156.969	0.120±0.015	0.422±0.027	0.447±0.029	0.217±0.028	14.000±**1.414**
	Entropy-based	–185510.511±2074.264	×	0.216±0.042	0.313±0.045	0.128±0.044	389.200±37.664
Alarm	Random	–184993.350±5853.005	×	0.278±0.090	0.409±0.130	0.166±0.106	379.000±30.183
10000	Benchmark	**–174219.558±602.242**	×	**0.957±0.000**	**0.957±0.000**	**0.478±0.000**	**238.200±10.323**
	BIC-based	–174913.070±2803.941	×	0.690±0.055	0.791±0.038	0.368±0.048	286.800±28.684
	Entropy-based	–593490.104±15283.241	×	0.363±0.058	0.424±0.059	0.391±0.059	1348.000±77.087
Hailfinder	Random	–625115.637±10248.728	×	0.245±0.093	0.294±0.112	0.267±0.102	1384.400±190.713
10000	Benchmark	**–489725.072±211.998**	×	0.770±**0.012**	0.730±**0.011**	0.750±0.011	**829.400±73.796**
	BIC-based	–519608.918±11238.885	×	**0.809**±0.014	**0.782**±0.007	**0.795±0.004**	1251.400±118.417
	Entropy-based	–1477041.770±49783.398	×	0.282±0.020	0.448±0.029	0.173±0.012	14196.000±1638.252
Andes	Random	–1522018.851±32834.350	×	0.243±0.025	0.390±0.025	0.150±0.013	16115.333±2078.353
10000	Benchmark	–1191163.696±20040.096	×	**0.938±0.003**	**0.920**±0.022	**0.464±0.005**	**6067.000**±1232.552
	BIC-based	**–1164586.578±12672.698**	×	0.660±0.017	0.727±**0.005**	0.346±0.006	9424.333±**1200.276**
	Entropy-based	–5964326.913	×	0.321	0.530	0.200	×
Pigs	Random	–6024467.904	×	0.291	0.490	0.183	×
10000	Benchmark	**–4448620.160**	×	**1.000**	**1.000**	**0.500**	×
	BIC-based	–4899597.984	×	0.676	0.830	0.373	×

better than the benchmark method, because it uses a random sample for nodes that are in the same rank of node order. Further, it is occasionally possible for our method to outperform the benchmark method when the variable number is large.

(2) For the learned BNs in the BIC score metric, when the number of nodes is extremely small, such as $n = 4$ or $n = 5$, the BIC scores of the learned BN are the best when using the Entropy-based method, whereas the results for the corresponding precision, recall, and F1 are the worst. It is clear that these results are completely inconsistent with the BIC scores, particularly with the Canner BN. But for $n \geqslant 8$, although the BIC score is consistent with the corresponding precision, recall, and F1 metrics, the results of these metrics are all poor when using the Entropy-based method. That is, the BIC score is unsuitable for measuring a much smaller BN when using the Entropy-based method.

However, the BIC scores of our methods are consistent with the corresponding precision, recall, and F1 metrics for all datasets except for the Sachs benchmark dataset, which is an unconnected DAG and includes two sub-DAGs. In general cases, the BIC score based on our proposed method is higher than that based on the Entropy-based method and the Random method. It is occasionally higher than that based on the benchmark method, such as for the Andes BN.

(3) For the learned BNs in the KL divergence metric, it achieves the best result based on the benchmark method for most datasets, our method achieves the second. The experiment results in Table 4 show that the KL divergence is consistent with that of the other metrics. In addition, because the KL divergence needs to compute the joint probability distribution of the BNs, it has a rather high computational complexity. Thus, we no longer compute the KL divergence in large BNs, and use "×" to express the results for BNs of $n > 20$.

(4) For run time, four methods have the same run times in

small BNs ($n < 10$) except for Survey BN, because the different node orders are ones that are sampled from different node order space based on four methods, respectively, with a certain probability, it is possible to sample the same good node order, and even other methods sample the better node order than our method and benchmark method. In addition, our method is suitable for single connected DAG and the benchmark Sachs BN includes two subDAGs, thus there is not good result on Sachs BN. In general, for big or large BNs, the bigger or larger n is, the higher the probabilities of our method sampling good node order are. In this experiment, the benchmark method achieves the best results and our method have the second ones on the big or large BNs, such as Alarm, Haifinder and Andes BNs, which is consistent with our intuition.

(5) In terms of the stability or robustness of the four methods, the rank of the standard deviation (sd) of the benchmark method is the best for the six metrics in most cases. There are 86.67% cases in which the sd achieves the best results for the benchmark method, that is,

$$\frac{7(\text{BIC}) + 4(\text{KL}) + 8(\text{precision}) + 7(\text{recall}) + 7(\text{F1}) + 6(\text{run time})}{8+5+8+8+8+8}.$$

Further, except for the benchmark method, compared with the other two methods, the stability or robustness of our BIC-based method is better in most cases. That is, 35 out of 45 (77.78%) cases are better than the other methods, except for the benchmark method.

To summarize, our proposed node order method outperforms the Entropy-based method and Random sample method in most cases. In addition, our method is more suitable for a connected graph.

4.4 Inference results on synthetic data

Because one of the main aims at learning a BN is to infer or compute certain queries, in this subsection we further verify the performance of our method based on the inference results. Here, we consider the inference tasks over domains where the true BN is known, and thus we can obtain the true inference probability. The nine true BNs are shown in Table 3. We can compare the inference performance of the learned BNs by the four different methods mentioned above with that of the corresponding true BNs.

Numerous inference algorithms have been developed, such as clique tree propagation algorithm [27, 28], iterative join graph propagation [29], and variational inference algorithm. In this experiment, we use the famous clique tree propagation algorithm to compute $\Pr_i(q)$ and $\widehat{\Pr}_i(q)$ by conducting an exact inference on each true BN and each learned BN, respectively. Note that for each BN, the inference results are the average of the mae from several different datasets except for the Pigs BN.

For each small BN ($n \leqslant 20$), such as the Cancer, Earthquake, Survey, Asia, and Sachs BNs, based on the mae determined through Eq. (24), we first take each variable as evidence to infer the posterior probabilities of other variables using the four learned BNs and the known true BN, and thus $c = n(n-1)$ in these cases. For each medium BN ($20 < n \leqslant 50$), each large BN ($50 < n \leqslant 100$), or each extremely large BN ($100 < n \leqslant 1000$), such as Alarm, Hailfinder, Andes, and Pigs BNs, because they have a rather high inference complexity, we consider taking each variable as a query variable and the other variables as evidence to compute the mae, and thus $c = n$ in these cases.

Note that in the BNs based on the four node order methods, the "mean±sd" in the inference results shown in Table 5 are obtained by conducting five times for small BNs ($n \leqslant 20$), three times for Alarm, Hailfinder, and Andes, and once for the Pigs BN, respectively. The comparison details of the inference results are shown in Table 5. The smallest mae of each BN based on four different node order methods is shown in bold in Table 5.

From Table 5, the benchmark method has the smallest mae or the best inference results in most cases. Our method has the second smallest mae and it outperforms the other two methods. Occasionally, our method has the equivalent best inference results, such as in the Cancer and Earthquake BNs, and even achieves much better inference results than those based on the benchmark method, such as with the Sachs BN.

Moreover, in terms of the stability or robustness of the four methods regarding the inference results of the learned BNs (ex-

Table 5 Comparison of the mae in the inference results

Dataset	c	Methods	Mae (mean ± sd)
Cancer BNs	$n(n-1)$	Entropy-based	0.02185±0.00513
		Random	0.01842±0.00737
		Benchmark	**0.00967±0.00326**
		BIC-based	**0.00967±0.00326**
Earthquake BNs	$n(n-1)$	Entropy-based	0.04220±**0.00395**
		Random	0.03460±0.00732
		Benchmark	**0.01721**±0.00549
		BIC-based	**0.01721**±0.00549
Survey BNs	$n(n-1)$	Entropy-based	0.00507±0.00094
		Random	0.00527±0.00094
		Benchmark	**0.00365±0.00084**
		BIC-based	0.00413±0.00085
Asia BNs	$n(n-1)$	Entropy-based	0.01024±0.00227
		Random	0.01181±0.00276
		Benchmark	**0.00674±0.00151**
		BIC-based	0.00999±0.00530
Sachs BNs	$n(n-1)$	Entropy-based	0.00519±0.00169
		Random	0.00682±0.00274
		Benchmark	0.00443±0.00072
		BIC-based	**0.00440±0.00058**
Alarm BNs	n	Entropy-based	0.08897±0.02752
		Random	0.08265±0.02519
		Benchmark	**0.03187±0.00962**
		BIC-based	0.06371±0.01702
Hailfinder BNs	n	Entropy-based	0.04288±0.01644
		Random	0.04106±0.01345
		Benchmark	**0.01685**±0.00611
		BIC-based	0.02128±**0.00484**
Andes BNs	n	Entropy-based	0.13245±0.00589
		Random	0.15127±0.00561
		Benchmark	**0.12952±0.00162**
		BIC-based	0.14578±0.00456
Pigs BNs	n	Entropy-based	0.50383
		Random	0.49961
		Benchmark	**0.56184**
		BIC-based	0.54067

cept for the benchmark method), six out of eight (75%) cases based on our method are better than those based on the Entropy-based and Random methods shown in Table 5. Note that the sd of the Pigs BN does not need to be considered because there is only one dataset. In conclusion, our BIC-based method is more stable or robust than the other methods except for the benchmark method.

5 Related work

Learning the BN structure is a critical and necessary task in probabilistic reasoning based on a Bayesian network (BN) model. BN learning includes structure learning [13, 14, 16, 20, 26, 30–39] and parameter learning [40–43]. When the DAG of the BN is known, we only need to conduct parameter learning. The methods of parameter learning can be summarised as the maximum likelihood estimation (MLE) and Bayesian estimation. When the DAG is unknown, we not only need to learn the DAG, but also need to learn its parameters, which is called the structure learning of the BN.

Existing methods of DAG structure learning can be generally classified into three categories: 1) score-&-search-based methods, 2) constraint-based algorithms, and 3) hybrid approaches. 1) Score-&-search-based methods, as mentioned previously, employ a scoring function to measure the quality of a candidate DAG. Heuristic strategies are used to search the DAG space of the possible candidates and evaluate the quality of the candidates with respect to predefined scoring functions. 2) Constraint-based algorithms exploit the dependent (and independent) relationships between variables based on information theory to learn the BN structure. Although constraint-based methods are computationally efficient (e.g., their running time is linear with respect to the number of data records and polynomial with respect to the number of attributes), the effectiveness may be handicapped through the engagement of too many conditional independence (CI) tests. Each test is built upon the results of the other tests, which leads to escalated errors. 3) Hybrid approaches combine the techniques of the first two categories. In general, a constraint-based approach is adopted as the initial step to restrict the search space of the possible structures, which is followed by a score-&-search-based method to search for the optimal DAG. To avoid the accumulated errors caused by CI tests involved in constraint-based methods and hybrid approaches, in this study we consider score-&-search-based methods, focusing on improving the computational efficiency incurred by the exponential DAG search space.

For a score-&-search-based BN structure method, a BN model selection and a BN model optimisation are included. We can select a BN model based on the score functions, such as CH [15], MDL [12], BIC, Akaike information criterion (AIC), and BDe [11]. Here, the BIC score function is a frequently used metric for the learned BN model, and we can optimise the BN model by considering some influential factors, such as limiting the number of parent nodes [15], giving the node order through the domain knowledge [15], sampling partial orders [20], applying the entropy-based sampling node order [13], swapping the order between two consecutive nodes [44], using an ordering-based search for learning the BN structure [21], applying a bounding tree width of the corresponding BN, conducting a parallel search [45], and providing the iteration conditions or limiting the iteration times. Here, the node order is one of the most important factors influencing the optimisation of the BN model.

Moreover, the other goal of learning the BN structure is to apply probabilistic reasoning other than a knowledge representation. A good indicator of the inference complexity of a BN model with DAG $\mathcal{G}$ is the treewidth of $\mathcal{G}$, which is denoted as $tw(\mathcal{G})$. However, determining the tree width of a graph is NP-hard [46], no efficient methods are available to solve this problem. Thus, to learn the BN structure with a low inference complexity, numerous researchers have addressed the bounded treewidth BN learning methods and have recently achieved some advanced results [13, 16, 19, 31, 32, 47]. In most of these methods, the node order is also rather helpful for improving the learning of the structure of a BN with a low inference complexity. Therefore, learning the node order from the data is an important and necessary issue in a knowledge representation and probabilistic reasoning based on the BN model.

However, existing node order learning methods have certain disadvantages: 1) The node order is rather difficult to be given by domain knowledge, as in [15], and thus we need to learn the node order from the data, particularly when the node number is large. 2) According to the frequently used BIC metric, the node order not only depends on its entropy but also depends on the joint entropy with the other node and its domain value. Moreover, the learned BN results are unstable because of the sample methods adopted [13,21,22]. 3) Most existing methods [23–26] have not been studied from a perspective consistent with the frequently used BIC measurement of the learning BN model. Therefore, in this study, we address the node order learning algorithm using data from the viewpoint of the frequently applied BIC measurement and avoid these disadvantages as much as possible.

6 Conclusion

To improve the BN structure learning, we proposed a node order learning method based on a frequently used BIC score function. The main contributions are summarized as follows:

- We provided the learning and connecting method for connecting undirected subgraphs based on BIC measure and proved the relative theorems of the nondirectional properties.
- We presented the orienting edges rules for an undirected graph based on the BIC.
- We proposed a learning algorithm of the node order for improving the BN structure learning and analysed its time complexity.
- Based on experiments conducted on synthetic databases, we illustrated the significant performance of our proposed learning method.

To sum up, the theoretical analysis and experimental results demonstrate that our algorithm outperforms other state-of-the-art methods in terms of BIC score, KL divergence, precision, recall, F1, run time and mae in most cases.

In the future, it will be interesting to show some theoretical analysis over the correctness of the learned graph structure. Also, although there are numerous existing methods for leaning a BN structure and obtaining better results from incomplete data [16, 20, 32, 48–50], there are few methods for learning the node order from incomplete data, and thus we have focused on learning the node order from incomplete data for improving the BN structure. In addition, we aim to address the relation between the DAGs' density and the number of node order spaces, this would clarify the situations in which the stability of learning results. Which remains as our further research.

Acknowledgements The work partially supported by the National Natural Science Foundation of China (Grant Nos. 61432011, U1435212, 61322211 and 61672332), the Postdoctoral Science Foundation of China (2016M591409), the Natural Science Foundation of Shanxi Province, China (201801D121115 and 2013011016-4) and Research Project Supported by Shanxi Scholarship Council of China (2020-095).

References

1. Judea P. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers, San Mateo, California, 1988
2. Friedman N. Inferring cellular networks using probabilistic graphical models. Science, 2004, 303(5659): 799–805
3. Raval A, Ghahramani Z, Wild D L. A Bayesian network model for protein fold and remote homologue recognition. Bioinformatics, 2002, 18(6): 788–801
4. Chen W, Zhu B, Zhang H. BN-mapping: visual analysis of geospatial data with Bayesian network. Chinese Journal of Computers, 2016, 39(7): 1281–1293
5. Peng P, Tian Y, Wang Y, Li J, Huang T. Robust multiple cameras pedestrian detection with multi-view Bayesian network. Pattern Recognition, 2015, 48(5): 1760–1772
6. Liu L, Wang S, Su G, Huang Z, Liu M. Towards complex activity recognition using a Bayesian network-based probabilistic generative framework. Pattern Recognition, 2017, 68: 295–309
7. Oatley G, Ewart B. Crimes analysis software: 'pins in maps', clustering and Bayes net prediction. Expert Systems with Applications, 2003, 25(4): 569–588
8. Chickering D M, Heckerman D, Meek C. Learning Bayesian networks is NP-hard. Technical Report, MSR-TR-94-17, Microsoft Research, Microsoft Corporation, 1994
9. Chickering D M, Heckerman D, Meek C. Large-sample learning of Bayesian networks is NP-hard. Journal of Machine Learning Research, 2004, 5: 1287–1330
10. Bouhamed H, Masmoudi A, Lecroq T, Rebai A. Structure space of Bayesian networks is dramatically reduced by subdividing it in subnetworks. Journal of Computational and Applied Mathematics, 2015, 287: 48–62
11. Heckerman D, Geiger D, Chickering D M. Learning Bayesian networks: the combination of knowledge and statistical data. Machine Learning, 1995, 20: 197–243
12. Lam W, Bacchus F. Learning Bayesian belief networks: an approach based on the MDL principle. Computational Intelligence, 1994, 10(3): 269–293
13. Scanagatta M, Corani G, De Campos C P, Zaffalon M. Approximate structure learning for large Bayesian networks. Machine Learning, 2018, 107: 1209–1227
14. De Campos C P, Scanagatta M, Corani G, Zaffalon M. Entropy-based pruning for learning Bayesian networks using BIC. Artificial Intelligence, 2018, 260: 42–50
15. Cooper G F, Herskovits E H. A Bayesian method for the induction of probabilistic networks from data. Machine Learning, 1992, 9: 309–347
16. Scanagatta M, Corani G, Zaffalon M, Yoo J, Kang U. Efficient learning of bounded-treewidth Bayesian networks from complete and incomplete data sets. International Journal of Approximate Reasoning, 2018, 95: 152–166
17. Nie S, De Campos C P, Ji Q. Learning Bayesian networks with bounded treewidth via guided search. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence. 2016, 3294–3300
18. Parviainen P, Farahani H S, Lagergren J. Learning bounded treewidth Bayesian networks using integer linear programming. In: Proceedings of the 17th International Conference on Artificial Intelligence and Statistics. 2014, 751–759
19. Elidan G, Gould S. Learning bounded treewidth Bayesian networks. Journal of Machine Learning Research, 2008, 9: 2699–2731
20. Niinimaki T, Parviainen P, Koivisto M. Structure discovery in Bayesian networks by sampling partial orders. Journal of Machine Learning Research, 2016, 17(1): 2002–2048
21. Teyssier M, Koller D. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence. 2005, 584–590
22. Scanagatta M, De Campos C P, Corani G, Zaffalon M. Learning Bayesian networks with thousands of variables. Neural Information Processing Systems, 2015, 28: 1855–1863
23. Chen X, Anantha G, Lin X. Improving Bayesian network structure learning with mutual information-based node ordering in the K2 algorithm. IEEE Transactions on Knowledge and Data Engineering, 2008, 20(5): 1–13
24. Ko S, Kim D. An efficient node ordering method using the conditional frequency for the K2 algorithm. Pattern Recognition Letters, 2014, 40: 80–87
25. Hsu W H, Guo H, Perry B B, Stilson J A. A permutation genetic algorithm for variable ordering in learning Bayesian networks from data. In: Proceedings of the Genetic and Evolutionary Computation Conference. 2002, 383–390
26. Park Y W, Klabjan D. Bayesian network learning via topological order. Journal of Machine Learning Research, 2017, 18: 1–32
27. Zhang L, Guo H. Introduction to Bayesian Networks. Science Press, 2006
28. Zhang N L, Yan L. Independence of causal influence and clique tree propagation. International Journal of Approximate Reasoning, 1998, 19(3–4): 335–349
29. Mateescu R, Kask K, Gogate V, Dechter R. Join-graph propagation algorithms. Journal of Artificial Intelligence Research, 2010, 37: 279–328
30. Goudie R J, Mukherjee S. A Gibbs sampler for learning DAGs. Journal of Machine Learning Research, 2016, 17: 1–39
31. Benjumeda M, Bielza C, Larranaga P. Learning tractable Bayesian networks in the space of elimination orders. Artificial Intelligence, 2019, 274: 66–90
32. Benjumeda M, Luengosanchez S, Larranaga P, Bielza C. Tractable learning of Bayesian networks from partially observed data. Pattern Recognition, 2019, 91: 190–199
33. Tsamardinos I, Brown L E, Aliferis C F. The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning, 2006, 65: 31–78
34. Lv Y, Wu J, Liang J, Qian Y. Random search learning algorithm of BN based on super-structure. Journal of Computer Research and Development, 2017, 54(11): 2558–2566
35. Qi X, Fan X, Gao Y, Liu Y. Learning Bayesian network structures using weakest mutual-information-first strategy. International Journal of Approximate Reasoning, 2019, 114: 84-98
36. Talvitie T, Eggeling R, Koivisto M. Learning Bayesian networks with local structure, mixed variables, and exact algorithms. International Journal of Approximate Reasoning, 2019, 115: 69–95
37. Scutari M, Graafland C E, Gutiérrez J M. Who learns better Bayesian

network structures: accuracy and speed of structure learning algorithms. International Journal of Approximate Reasoning, 2019, 115: 235–253
38. Ye Q L, Amini A A, Zhou Q. Optimizing regularized cholesky score for order-based learning of Bayesian networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, DOI: 10.1109/TPAMI.2020.2990820
39. Lee S, Kim S B. Parallel simulated annealing with a greedy algorithm for Bayesian network structure learning. IEEE Transactions on Knowledge and Data Engineering, 2020, 32(6): 1157–1166
40. Yao T, Choi A, Darwiche A. Learning Bayesian network parameters under equivalence constraints. Artificial Intelligence, 2017, 244: 239–257
41. Riggelsen C. Learning parameters of Bayesian networks from incomplete data via importance sampling. International Journal of Approximate Reasoning, 2006, 42: 69–83
42. Niculescu R S, Mitchell T M, Rao R B. Bayesian network learning with parameter constraints. Journal of Machine Learning Research, 2006, 7: 1357–1383
43. Yang Y, Gao X, Guo Z, Chen D. Learning Bayesian networks using the constrained maximum a posteriori probability method. Pattern Recognition, 2019, 91: 123–134
44. Benjumeda M, Bielza C, Larranaga P. Tractability of most probable explanations in multidimensional Bayesian network classifiers. International Journal of Approximate Reasoning, 2018, 93: 74–87
45. Madsen A L, Jensen F, Salmeron A, Langseth H, Nielsen T D. A parallel algorithm for Bayesian network structure learning from large data sets. Knowledge Based Systems, 2017, 117: 46–55
46. Arnborg S, Corneil D G, Proskurowski A. Complexity of finding embeddings in a k-tree. SIAM Journal on Algebraic Discrete Methods, 1987, 8(2): 277–284
47. Nie S, Maua D D, De Campos C P, Ji Q. Advances in learning Bayesian networks of bounded treewidth. Advances in Neural Information Processing Systems, 2014, 27: 2285–2293
48. Liao W, Ji Q. Learning Bayesian network parameters under incomplete data with domain knowledge. Pattern Recognition, 2009, 42: 3046–3056
49. Lv Y, Wu J, Jing T. Pqisem: BN's structure learning based on partial qualitative influences and SEM algorithm from missing data. International Journal of Wireless and Mobile Computing, 2018, 14(4): 348–357
50. Masegosa A R, Feelders A, Der Gaag L C. Learning from incomplete data in Bayesian networks with qualitative influences. International Journal of Approximate Reasoning, 2016, 69: 18–34

Yali Lv received the PhD degree in Computer Application Technology from Tianjin University, China. She is a professor at Shanxi University of Finance & Economics of China, and also a member of Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Shanxi University, China. From Sep. 2018 to Oct. 2019, she visited the University of Technology Sydney (UTS), Australia, as a visiting scholar. Her research interests include probabilistic reasoning, data mining and machine learning.

Junzhong Miao received the Bachelor degree in Computer Science and Technology from Shanxi University of Finance & Economics, China. He is currently a master candidate at Shanxi University of Finance & Economics of China. His research interests include Bayesian machine learning and data mining.

Jiye Liang received the PhD degree in Applied Mathematics from Xi'an Jiaotong University, China. He is currently a professor and PhD supervisor in the Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Shanxi University, China. His research interests include artificial intelligence, granular computing, data mining and machine learning. He has published more than 100 articles and his papers appear in major international journals including AI, PAMI, TKDE, TNNLS, and so on.

Ling Chen received the PhD degree in Computer Engineering from Nanyang Technological University, Singapore. She is currently an associate professor and PhD supervisor in the Priority Research Center for Artificial Intelligence (CAI), University of Technology Sydney (UTS), Australia. Before joining UTS, She was a Postdoctoral Research Fellow with L3S Research Center, Leibiniz University of Hannover, Germany. Her current research interests include machine learning, data mining, social network analysis, and recommendation system. Her papers appear in major conferences and journals including SIGKDD, ICDM, SDM, ACM TOIS, TKDE, and TNNLS.

Yuhua Qian received the PhD degrees in Computer Application Technology from Shanxi University, China. He is currently a professor and PhD supervisor in the Institute of Big Data Science and Industry, and also a member of the Key Laboratory of Computational Intelligence and Chinese Information Processing at the Ministry of Education, Shanxi University, China. His research interests include data mining and machine learning, granular computing, and artificial intelligence. He has published more than 80 articles and his papers appear in major international journals including AI, KDM, TKDE and so on. He is a member of the IEEE.