Local-to-global learning approach plays an essential role in Bayesian network (BN) structure learning. Existing local-to-global learning algorithms first construct the skeleton of a DAG (directed acyclic graph) by learning the MB (Markov blanket) or PC (parents and children) of each variable in a dataset, then orient edges in the skeleton. However, existing MB or PC learning methods are often computationally expensive especially with a large-sized BN, resulting in inefficient local-to-global learning algorithms. To tackle the problem, in this article, we link feature selection with local BN structure learning and develop an efficient local-to-global learning approach using filtering feature selection. Specifically, we first analyze the rationale of the well-known Minimum-Redundancy and Maximum-Relevance (MRMR) feature selection approach for learning a PC set of a variable. Based on the analysis, we propose an efficient F2SL (feature selection-based structure learning) approach to local-to-global BN structure learning. The F2SL approach first employs the MRMR approach to learn the skeleton of a DAG, then orients edges in the skeleton. Employing independence tests or score functions for orienting edges, we instantiate the F2SL approach into two new algorithms, F2SL-c (using independence tests) and F2SL-s (using score functions). Compared to the state-of-the-art local-to-global BN learning algorithms, the experiments validated that the proposed algorithms in this article are more efficient and provide competitive structure learning quality than the compared algorithms.

CCS Concepts: • Computing methodologies → Causal reasoning and diagnostics;

Additional Key Words and Phrases: Bayesian network, feature selection, local-to-global structure learning, markov blanket

ACM Reference format:
Kui Yu, Zhaolong Ling, Lin Liu, Peipei Li, Hao Wang, and Jiuyong Li. 2023. Feature Selection for Efficient Local-to-global Bayesian Network Structure Learning. *ACM Trans. Knowl. Discov. Data.* 18, 2, Article 37 (November 2023), 27 pages. https://doi.org/10.1145/3624479


1 INTRODUCTION

Learning Bayesian network (BN) from observational data is an important problem in data mining, playing an essential part in inferring conditional independence and causal relations between variables [1]. BN learning has had many applications in various areas such as bioinformatics [11], neuroscience [3], and information retrieval [10].

The structure of a BN is represented by a DAG (directed acyclic graph) where nodes of the DAG represent the variables and edges represent dependence between variables. When there is an edge $X \rightarrow Y$, $X$ is known as a parent of $Y$ and $Y$ is a child of $X$. In general, learning a DAG over a large number of variables is computationally intractable [7, 31]. To alleviate the computational complexity, the local-to-global approach was proposed to reduce the DAG search space [12, 24, 37]. Instead of searching the entire DAG space over all the variables simultaneously, the local-to-global approach first finds the Markov blanket (MB) or parents and children (PC) of each variable (without distinguishing parents from children), then uses the learnt MB (or PC) set of each variable to construct the DAG skeleton, and finally orients edges in the skeleton. An MB of a node in a BN consists of the parents, children, and spouses (i.e., other parents of the node’s children) of the node in a BN. Figure 1 gives an example of an MB in the BN of lung cancer [16]. The MB of Lung cancer consists of Smoking and Genetics (parents), Coughing and Fatigue (children), and Allergy (spouse).

The current MB learning methods are mainly categorized into two types: simultaneous MB learning and divide-and-conquer MB learning. However, these methods are either inefficient or ineffective. Given a variable of interest, simultaneous MB learning algorithms learn the MB of the variable simultaneously without distinguishing PC from spouses and are not capable of constructing high-quality DAG skeletons especially when a dataset has a small number of data samples with high-dimensionality, but they are computationally efficient. Divide-and-conquer methods first employ a Parent-Child (PC) learning algorithm to learn the PC set of the variable, then find the spouses of the variable. They are effective for DAG skeleton construction, but they are not computationally efficient when the size of PC is large. This implies that existing local-to-global BN structure learning algorithms are either inefficient or ineffective, depending on which type of MB or PC learning methods are used.

Feature selection aims to select a subset of features with regard to a class variable of interest from the original set of features, and it is an essential preprocessing step for model building or data understanding in data analytics [4]. Existing feature selection methods can be broadly categorized into filtering, wrapper, and embedded methods [21, 38]. Filter methods are classifier-independent, and the other two types of methods are classifier-dependent.

Filtering feature selection methods have been attracting major attention, due to their fast processing speed and independence of prediction models. Studies have shown that under certain assumptions, the MB of a class variable is the optimal feature set for supervised machine learning tasks, while existing filtering feature selection methods attempt to find an approximate MB (e.g., PC) of a class variable [34]. A question is whether we can make use of efficient filtering feature selection methods to learn the MB/PC set of each variable in a dataset to significantly improve computational efficiency of BN structure learning. To investigate this question, in this article, we first establish the theoretical link of filtering feature selection methods with local BN structure learning (i.e., MB/PC learning) and then propose an efficient F2SL (Feature Selection-based Structure Learning) approach to speeding up local-to-global BN structure learning. Our contributions can be summarized as follows:

- We employ a well-known mutual information-based feature selection approach, called Min-Redundancy and Maximum-Relevance (MRMR) [4], to find parents and children for speeding up structure learning. To answer why the MRMR approach is able to learn the PC
of a variable of interest, we first analyze the relationship of the objective function of MRMR and MB learning. Then, we analyze the output of MRMR related to the MB. And finally, we discuss the instantiation of MRMR for PC learning.

Based on the analysis above, we propose the F2SL approach to local-to-global BN structure learning. F2SL employs the MRMR approach to learn the PC of each variable in a dataset, then constructs the DAG skeleton, and finally it orients edges in the skeleton. Using independence tests or score functions for orienting edges, F2SL is instantiated to two algorithms, F2SL-c (using independence tests) and F2SL-s (using score functions).

We conduct extensive experiments to validate F2SL-c and F2SL-s against the five representative local-to-global BN learning algorithms. The experiments results show that the proposed algorithms are significantly faster and also achieve better structure learning quality than the five rivals.

The article is organized as follows: Section 2 reviews the related work, and Section 3 gives notations and definitions. Section 4 presents an analysis in theory, while Section 5 proposes the new algorithms. Section 6 describes and discusses the experiments, and Section 7 concludes the article.

2 RELATED WORK

In past decades, learning BN structures has been an important task in data mining and machine learning [17, 20]. There are two main types of structure learning methods: score-based and constraint-based methods [15, 43]. Score-based algorithms use a scoring function to perform a global structure learning over a search space of possible DAGs over all the variables in a dataset [6, 9]. Constraint-based methods employ independence tests to first estimate whether there is an edge between two variables and then orient edge directions [5, 8]. Existing BN learning approaches formulated the BN structure learning problem as a traditional combinatorial optimization problem and depend on various local heuristics for enforcing the acyclicity constraint.

To avoid the combinatorial constraint, Zheng et al. [45] have formulated the BN structure learning problem as a continuous optimization problem instead of the traditional combinatorial optimization problem. Some recent studies have leveraged the idea in Reference [45] to learn BN structures using deep neural networks. Yu et al. [41] have designed a BN structure learning algorithm using graph neural networks, and Zhang et al. [44] have proposed a variational autoencoder-based method for learning BN structures.

Since the search space of DAGs is combinatorial and exponential with the number of variables, existing global BN structure learning methods are often computationally infeasible when the number of variables is large. Then, to improve efficiency of BN structure learning, local-to-global BN structure learning methods were proposed, which contain two steps: skeleton learning and edge
orientation. In the skeleton learning step, the local-to-global approach first learns the MB or PC of each variable in a dataset independently, then constructs the DAG skeleton (i.e., the undirected graph) using the learnt MB or PC sets. Through learning each variable’s MB or PC locally, the local-to-global approach significantly reduces the potential DAG search space, and thus can be scalable to thousands of variables. In the edge-orientation step, edges are oriented in the skeleton using independence tests or score functions.

How to efficiently learn the MB or PC of a variable for skeleton learning is the key to existing local-to-global BN learning algorithms. To learn skeletons, many MB and PC learning algorithms have been proposed, and they fall in two types: constraint-based methods and score-based approaches. Constraint-based MB learning algorithms employ independence tests and are mainly divided into two types: simultaneous learning approach and divide-and-conquer approach. The representative algorithms of the former type include GSMB [24], IAMB [36], and Inter-IAMB [36], while the representative algorithms of the latter type are HITON-MB [1, 2], MMMB [35], PCMB [29], STMB [13], and BAMB [23]. PC-simple [22], MMPC [35], and HITON-PC [2] are the three widely used algorithms for learning PC of a variable using independence tests. Score-based MB (or PC) learning methods use a score function to learn MB or PC and mainly include the SLL (Score-based Local Learning) [26] and $S^2$TMB (Score-based Simultaneous MB) algorithms [14].

Based on these MB or PC learning algorithms, several local-to-global structure learning methods were proposed. The GSBN [24] algorithm employs the GSMB algorithm for learning skeleton and orients edge using independence tests. SLL+C [26] uses the SLL algorithm to learn the MB of each variable and employs independence tests for orienting edges, while SLL+G [26] uses score-based methods for edge orientations. MMHC [37] uses MMPC for learning skeletons and employs a score function and hill-climbing search strategy for orienting edges. Thus, both SLL+C and MMHC belong to a hybrid local-to-global approach. Gao et al. [12] recently proposed a novel GGSL (Graph Growing Structure Learning) algorithm for local-to-global structure learning. Instead of finding the MB of each variable in advance, GGSL first randomly selects a variable and learns the local structure around the variable using the $S^2$TMB algorithm, then iteratively applies the local learning procedure to the variable’s neighbors for gradually expanding the learned local BN structure until a global BN structure is achieved.

Existing MB or PC learning algorithms have the following main drawbacks: GSMB and IAMB are efficient, but the number of samples required by them grows exponentially with the size of the MB of the target variable, since they use the entire set of variables selected currently as conditioning sets for independence tests. HITON-MB, MMMB, PCMB, STMB, and BAMB mitigate the problem of the large sample requirement by performing an exhaustive subset search within the variables selected currently, but the search is computationally expensive when the size of the currently selected variables becomes large. The computational cost of SLL and $S^2$TMB is determined by BN structure learning algorithms, since at each iteration SLL and $S^2$TMB need to use a BN structure learning algorithm to learn a local BN structure (involving all variables selected currently) around a variable. Due to the computational complexity of existing BN structure learning algorithms, they face the scalability issues when the size of the local BN structure becomes large. To improve the MB learning efficiency, Pellet et al. [28] proposed the TC (Total Conditioning) algorithm for learning BN structures from Gaussian data. The TC algorithm employs a regression-based feature-selection method to learn DAG skeletons and then uses independence tests to orient edges.

In the past decade, researchers have proposed many methods for distinguishing causes from effects purely from observational data in the two-variable case [42]. These methods are mainly divided into two types: methods based on additive noise models [18, 32] and methods based on information geometric causal inference [19]. The focus of this article is on learning a complete BN structure containing all variables in a problem of interest. For readers who are interested in the
research on distinguishing causes from effects in the two-variable case, more references can be found in the recent survey proposed by Mooij et al. [25].

In this article, we focus on address the computational problem of existing MB or PC learning algorithms and focus on learning BN structures from discrete data with multivariate random variables.

3 NOTATIONS AND DEFINITIONS

3.1 Bayesian Network and Markov Blanket

In this section, we will introduce some basic definitions. Let \( P \) be the joint probability distribution represented by a DAG \( G \) over a set of random variables \( V = \{V_1, \ldots, V_M\} \). We use \( V_i \perp V_j | S \) to denote that \( V_i \) and \( V_j \) are conditionally independent given \( S \subseteq V \setminus \{V_i, V_j\} \), and \( V_i \not\perp V_j | S \) to represent that \( V_i \) and \( V_j \) are conditionally dependent given \( S \). The definition of conditional independence (and dependence) is given as follows:

**Definition 1 (Conditional Independence).** Given two distinct variables \( V_i, V_j \in V \) are said to be conditionally independent given a subset of variables \( S \subseteq V \setminus \{V_i, V_j\} \) (i.e., \( V_i \perp V_j | S \)) if and only if \( P(V_i, V_j | S) = P(V_i | S)P(V_j | S) \). Otherwise, \( V_i \) and \( V_j \) are conditionally dependent given \( S \), i.e., \( V_i \not\perp V_j | S \).

The symbols \( pa(V_i) \), \( ch(V_i) \), and \( sp(V_i) \) denote the sets of parents, children, and spouses of \( V_i \), respectively. We call the triplet \( \langle V, G, P \rangle \) a **Bayesian network (BN)** if \( \langle V, G, P \rangle \) satisfies the Markov condition: Every variable is independent of any subset of its non-descendant variables given its parents in \( G \) [27]. In a BN \( \langle V, G, P \rangle \), by the Markov condition, the joint probability \( P \) can be decomposed into the product of conditional probabilities as

\[
P(V_1, V_2, \ldots, V_M) = \prod_{i=1}^{M} P(V_i | pa(V_i)).
\]  

(1)

**Definition 2 (D-separation [27]).** In a DAG \( G \), a path \( \pi \) is said to be d-separated (or blocked) by a set of vertices \( S \subseteq V \) if and only if (1) \( \pi \) contains a chain \( V_i \rightarrow V_k \rightarrow V_j \) or a fork \( V_i \leftarrow V_k \leftarrow V_j \) such that the middle vertex \( V_k \) is in \( S \) or (2) \( \pi \) contains an inverted fork (or collider) \( V_i \rightarrow V_k \leftarrow V_j \) such that the middle vertex \( V_k \) is not in \( S \) and such that no descendant of \( V_k \) is in \( S \).

A set \( S \) is said to d-separate \( V_i \) from \( V_j \) if and only if \( S \) blocks every path from a vertex in \( V_i \) to a vertex in \( V_j \).

**Definition 3 (Faithfulness).** [33] Given a BN \( \langle V, G, P \rangle \), \( P \) is faithful to \( G \) if \( \forall V_i, V_j \in V, \exists S \subseteq V \setminus \{V_i, V_j\} \) d-separates \( V_i \) and \( V_j \) in \( G \) if \( V_i \not\perp V_j | S \) holds in \( P \).

The faithfulness assumption establishes a relation between a probability distribution \( P \) and its underlying DAG \( G \). In a BN, the faithfulness assumption implies that two variables \( V_i, V_j \in V \) that are d-separated with each other by a subset \( S \subseteq F \setminus \{F_i, F_j\} \) in \( G \) are conditionally independent conditioning on \( S \) in \( P \). Under the assumption, we can use conditional independence tests, instead of d-separation, to find all dependencies or independencies entailed with a Bayesian network.

**Definition 4 (Markov Blanket).** [27] Under the faithfulness assumption, the MB of a variable \( V_i \) in \( G \), noted as \( MB(V_i) \), is unique and consists of parents, children, and spouses of \( V_i \).

In the following, Lemma 1 states the dependencies between a variable and its parents (or children). Lemma 2 denotes the independence/dependence relations of a variable and its spouses.

**Lemma 1.** [33] Under the faithfulness assumption, for \( V_j \in V \) and \( V_i \in V \), there is an edge between \( V_j \) and \( V_i \) if and only if \( V_j \not\perp V_i | S \), for all \( S \subseteq V \setminus \{V_j, V_i\} \).
Lemma 1 indicates that if $V_j$ is a parent or a child of $V_i$, then $V_j$ and $V_i$ are conditionally dependent given any subset $S$ of $V \setminus \{X, V_i\}$.

**Lemma 2.** [33] In a Bayesian network, assuming that $V_i$ is adjacent to $V_k$, $V_j$ is adjacent to $V_k$, and $V_i$ is not adjacent to $V_j$ (e.g., $V_i \rightarrow V_k \leftarrow V_j$), if $\forall S \subseteq V \setminus \{V_i, V_k, V_j\}$, $V_i \not\perp V_j \mid S$ and $V_i \not\perp V_j \mid S \cup \{V_k\}$ hold, then $V_j$ is a spouse of $V_i$.

Lemma 2 states that if $V_j$ is a spouse of $V_i$, then $V_j$ and $V_i$ are independent conditioning on a subset $S$ excluding their common child, but they are dependent conditioning on $S$ including their common child.

### 3.2 Mutual Information and Conditional Independence

Given variable $X$, the entropy of $X$ is defined as

$$H(X) = -\sum_x P(x) \log P(x).$$

(2)

The entropy of $X$ after observing values of another variable $Y$, denoted as $I(X; Y)$, is defined as

$$H(X|Y) = -\sum_y P(y)\sum_x P(x|y) \log P(x|y).$$

(3)

In Equations (2) and (3), $P(x)$ is the prior probability of $X = x$ (i.e., the value $x$ that $X$ takes), and $P(x|y)$ is the posterior probability of $X = x$ given $Y = y$. To calculate Equation (2), we need the estimated distributions $P(x)$ and $P(x|y)$. For a discrete dataset with $N$ independently and identically distributed samples, we estimate the probability of $P(X = x)$ using maximum likelihood, i.e., the frequency of $X = x$ occurring in the dataset divided by the total number of data samples (i.e., $N$) of the dataset. The posterior probability $P(x|y)$ is calculated by the frequency of $X = x$ occurring given that $Y = y$ holds in the dataset divided by the total number of data samples $N$.

According to Equations (2) and (3), the mutual information between $X$ and $Y$, denoted as $I(X; Y)$, is defined as

$$I(X; Y) = H(X) - H(X|Y) = \sum_{x, y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)}.$$  

(4)

From Equation (4), the conditional mutual information between $X$ and $Y$ given another variable $Z$ is defined as:

$$I(X; Y|Z) = H(X|Z) - H(X|Y, Z) = \sum_z P(z)\sum_{x, y \in Y} P(x, y|z) \log \frac{P(x, y|z)}{P(x|z)P(y|z)}.$$  

(5)

In Equations (4) and (5), $I(X; Y) = 0$ if $X$ and $Y$ are independent (i.e., $P(X, Y) = P(X)P(Y)$), $I(X; Y) = 0$ and $I(X; Y|Z) = 0$ if $X$ and $Y$ are conditionally independent given $Z$.

### 4 MRMR FEATURE SELECTION & MB LEARNING

Given a dataset $D$ consisting of a variable set $V$ and the class attribute $C \in V$, in this section, we will analyze the output of the MRMR feature selection approach with relation to the MB of $C$.

### 4.1 Objective Functions of MRMR and MB Learning

**Objective function of MB learning.** Under the faithfulness assumption, $MB(C)$ is the optimal feature set for the classification problem with $C$ as the class variable, and $MB(C)$ satisfies the following property [39]:

**Theorem 1.** $\forall S \subseteq V \setminus C, I(C; MB(C)) \geq I(C; S)$.

In Theorem 1, except for $S = MB(C)$, if $S$ is a superset of $MB(C)$, then $I(C; MB(C)) = I(C; S)$ also holds. In Reference [39], the authors do not consider this case. This case means that although
by the property of the MB, $C$ is conditionally independent of the remaining variables $S$ conditioning on $MB(C)$ (i.e., $VS \subseteq F \setminus MB(C)$, $P(C|S, MB(C)) = P(C|MB(C))$), $C$ and $S$ also conditionally independent conditioning on a superset of $MB(C)$, i.e., $VS, S' \subseteq F \setminus MB(C)$, $P(C|S, MB(C) \cup S') = P(C|MB(C) \cup S')$. Then, Theorem 1 indicates that learning $MB(C)$ is equivalent to finding a subset $S \subseteq V \setminus C$ that maximizes $I(S; C)$ and $MB(C)$ is the minimal and optimal set of $S$, since adding features to the $MB(C)$ set does not increase the mutual information to $C$. Although the issue of the superset exists in Theorem 1, it does not put any impact on existing algorithms for learning $MB(C)$ or $PC(C)$. Since the MB property $VS \subseteq F \setminus MB(C)$, $P(C|S, MB(C)) = P(C|MB(C))$) holds, these algorithms can use this property to remove redundant features in the learnt MB and are able to identify the minimal MB of $C$ (i.e., parents, children, and spouses of $C$). In addition, if we assume that all conditional independence tests are reliable and the faithfulness assumption holds, then existing MB learning algorithms achieve a correct MB of a target variable in theory.

However, in Theorem 1, it is a challenging combinatorial optimization problem for identifying $MB(C)$ from $F$. Therefore, almost all MB learning methods have adopted a greedy strategy by considering features one-by-one to find $S^*$. Specifically, at each iteration, given the currently selected set of features, $S$, they choose the feature $X$ in $V \setminus (S \cup \{C\})$ that maximizes $I((S \cup \{X\}) \setminus C)$. Since $I((S \cup X); C) = I(S; C) + I(X; C|S)$ holds and at each iteration $I(S; C)$ is the same for each variable $X$, this greedy strategy can be formulated as Equation (6) as follows:

$$X^* = \arg \max_{X \in V \setminus (S \cup \{C\})} I(X; C|S). \quad (6)$$

To solve Equation (6), there are two types of MB learning methods. One is to calculate $I(X; C|S)$ in Equation (6) using the entire set $S$ of features currently selected, i.e., all the features in $S$, such as GSMB [24] and IAMB [36], while the other is to calculate $I(X; C|S)$ using all possible subsets of $S$, such as HITON-MB [2] and MMIB [35]. When the size of $S$ increases, it will be impractical for the first type of methods to compute $I(X; C|S)$, because the computation demands a large number of training samples while it will be computationally intractable for the second type of methods, since the number of subsets of $S$ will be exponential to the number of features in $S$.

**Objective function of MRMR.** To tackle these problems, since $I(X; S; C) = I(X; S) - I(X; S|C) = I(X; C) - I(X; C|S)$ holds, MRMR decomposes Equation (6) into Equation (7) as follows:

$$X^* = \arg \max_{X \in V \setminus (S \cup \{C\})} \{I(X; C) - I(X; S) + I(X; S|C)\}. \quad (7)$$

In Equation (7), $I(X; C)$ represents the relevancy of $X$ to $C$, $I(X; S)$ denotes the redundancy of $X$ with respect to $S$, and $I(X; S|C)$ indicates the class-conditional relevance, which considers the situation where a feature $X$ provides more predictive information by jointly with other features (i.e., those in $S$) than by itself with respect to $C$. For example, in Figure 1, “Allergy” is independent of “Lung Cancer,” since “Allergy→Coughing→Lung Cancer” is a v-structure in the BN structure. Then, “Allergy” itself does not provide any information for predicting “Lung Cancer.” However, “Allergy” provides strongly predictive information by jointly with “Coughing.” For instance, if a doctor knows that a patient’s coughing is caused by his/her allergies, then he will reduce the risk of diagnosing this patient with lung cancer.

When the size of $S$ becomes large, if we direct compute $I(X; S)$ or $I(X; S|C)$ in Equation (7), on the one hand, then it will result in computationally expensive; on the other hand, the big size of $S$ will lead to the large sample requirement to guarantee the reliable results of computing $I(X; S)$ or $I(X; S|C)$. To reduce high-order mutual information computations to low-order mutual information calculations, we require highly restrictive assumptions made on the dependence/independence between features.
First, we use Assumption 1 to deal with the class-conditional relevance term $I(X; S|C)$. Assumption 1 is the assumption by a naive Bayes classifier and it assumes that the features in $V \setminus C$ are pairwise-independent conditioning on $C$. By Assumption 1, we can easily get $I(X; S|C) = 0$ and the term $\{I(X; C) - I(X; S) + I(X; S|C)\}$ in Equation (7) is reduced to $\{I(X; C) - I(X; S)\}$. This assumption makes MRMR only interested in which variables have an edge with $C$, i.e., parents and children of $C$ (the local structure of $C$). For example, in Figure 1, the MB of “Lung Cancer” consists of its parents and children, “Smoking,” “Genetics,” “Fatigue,” and its spouse, “Allergy.” By Assumption 1, we are only interested in parents and children of “Lung Cancer”: “Smoking,” “Genetics,” “Coughing,” and “Fatigue.”

**Assumption 1.** $\forall V_i, V_j \in V$ and $i \neq j$, $V_i$ and $V_j$ are assumed to be conditionally independent given the class attribute $C$, that is, $P(V_i, V_j|C) = P(V_i|C)P(V_j|C)$.

To tackle the redundancy term $I(X; S)$, Assumption 2 assumes the selected features in $S$ are conditionally independent with each other conditioning on a feature $X$ in $V \setminus S$ (i.e., conditioning on a currently unselected feature). By the chain rule, $H(S|X) = \sum_{i=1}^{[S]} H(V_i|V_{i-1}, \ldots, V_1, X)$ holds. With Assumption 2 and Equation (4), we have $I(X; S) = H(S) - \sum_{i=1}^{[S]} H(V_i) + \sum_{i=1}^{[S]} I(V_i; X)$. Since at each iteration $H(S) - \sum_{i=1}^{[S]} H(V_i)$ is the same for all unselected features in $V \setminus S$ and thus removing them will have no effect on the choice of features, the term $I(X; S)$ in Equation (7) is rewritten as $\sum_{i=1}^{[S]} I(V_i; X)$ with the pairwise mutual information between a currently unselected feature $X$ and a feature $V_i$ in $S$ without conditioning on other features.

**Assumption 2.** The selected features in $S$ are conditionally independent given an unselected feature $X \in V \setminus S$, that is, $P(S|X) = \prod_{V_i \in S} P(V_i|X)$ where $V_i \in S$.

Under Assumptions 1 and 2, we reformulate Equation (7) as Equation (8), i.e., the MRMR objective function. The MRMR approach reduces Equation (6) to a linear combination of low-order mutual information terms. To select relevant features with regard to $C$, at each iteration, Equation (8) selects the feature $X^*$ that has the maximum relevancy with $C$ and the minimum redundancy with the features in $S$.

$$X^* = \arg \max_{X \in V \setminus (S \cup \{C\})} \left\{ I(X; C) - \sum_{i=1}^{[S]} I(V_i; X) \right\}. \tag{8}$$

Assumption 2 makes an assumption on the pairwise-dependence/independence between features in $S$, then we can calculate the high-order redundancy term $I(S; X)$ with the pairwise mutual information between $X$ and a feature in $S$ (i.e., $\sum_{i=1}^{[S]} I(V_i; X)$). This pairwise-dependence/independence assumption implies that, at each iteration, we consider each feature in $V \setminus (S \cup \{C\})$ and choose a feature $X$ that has no redundancy with all variables in $S$ to keep all features in $S$ mutually independent. We use the Bayesian network structure in Figure 1 as an example to illustrate Assumption 2. Using Equation (6), assuming the currently selected feature set $S$ contains three features, {“Smoking”, “Genetics”, “Fatigue”}, without Assumption 2, when considering whether a feature $X$ can be added to $S$ or not, we need to calculate the high-order term $I(X; C|${“Smoking”, “Genetics”, “Fatigue”}) or $I({“Smoking”, “Genetics”, “Fatigue”}; X)$ or computing $I(X; C|${“Smoking”, “Genetics”, “Fatigue”}) using all possible subsets of $S$. Just as we discussed above, it is not easy to directly compute this high-order term due to both time and sample complexity, especially when the size of $S$ becomes large. With Assumption 2, we can decompose the high-order term $I({“Smoking”, “Genetics”, “Fatigue”}; X)$ into the three low-order terms without conditioning on other features: $I({“Smoking”; X})$, $I({“Genetics”; X})$, and $I({“Fatigue”; X})$. If $X$ is independent with “Smoking”, “Genetics”, and “Fatigue”, then $X$ will be added to $S$ with a high probability. That is to say, to keep the selected features in $S$ mutually independent (i.e., having no redundancy) as much as possible, if a currently unselected feature $X$ has no redundancy with
In Figure 8= is selected by MRMR or not in the current iteration), by > of or with a high and a parent or a child > and the relationship of mutual information and conditional independence discussed in C will be added to > and > in Equation (8) is a spouse of > and > may not be added to > in Equation (8) is a child of > is added to > according to Lemma > due to > is a common child of > or not (i.e., > holds in Figure > shows the three-way variable interactions in a BN. Using these interactions in prior to > is a spouse of > S > of > S > that share the same child as > and the variable > outside > Figure 2 shows the three-way variable interactions in a BN. Using these interactions in Figure 2 and the relationship of mutual information and conditional independence discussed in Section 3.2, the output of MRMR based on Equation (8) is analyzed as follows:

- **The output of MRMR contains all parents and children of C.** In Figures 2(a) and (b), X and Y are each a parent or a child of C. No matter whether X or Y enters S or not (i.e., whether X or Y is selected by MRMR or not in the current iteration), by I(C; (X, Y)) = I(C; X)+I(C; Y|X), both I(C; (X, Y)) > I(C; X) and I(C; (X, Y)) > I(C; Y) hold. In Figure 2(a), I(X; Y|C) = 0. By I(X; Y; C) = I(X; C)−I(X; C|Y) = I(Y; C)−I(Y; C|X) = I(X; Y)−I(X; Y|C), we get I(C; Y)−I(X; Y) = I(C; Y)−I(C; Y) + I(C; Y|X)−I(X; Y|C) = I(C; Y|X). Thus, I(C; Y)−I(X; Y) > 0 holds in Figure 2(a). Similarly, I(C; X)−I(X; Y) > 0. In Figure 2(b), I(C; Y)−I(X; Y) = I(C; Y) > 0 and I(C; X)−I(X; Y) > 0. Using Equation (8), MRMR selects all parents and children of C in its output.

- **The output of MRMR does not contain spouses of C.** In Figure 2(c), Y is a spouse of C with regard to X. By Lemma 1, I(C; Y) = 0 and I(C; X) > 0. Thus, I(C; X) > I(C; Y) and X will be added to S prior to Y. Meanwhile, since I(C; (X, Y)) = I(C; X)+I(C; Y|X) and C ⊥ Y|X according to Lemma 2, we get that I(C; (X, Y)) > I(C; X). This illustrates that a child of C (e.g., X) provides more predictive information of C jointly with X’s another parent, i.e., the spouse of C that shares the same child as C (e.g., Y) than by itself. Using Equation (8), assuming that currently X is added to S before Y due to X is a child of C, when adding Y to S, I(C; Y)−I(X; Y) < 0 hold. Then, using Equation (8) MRMR does not include spouses of C in its output.

- **The output of MRMR does not contain non-parent ancestors and non-child descendants of C.** As shown in Figure 2(d), for a non-parent ancestor or a non-child descendant Y of C and a parent or a child X of C on the path from Y to C, since I(C; Y|X) = 0

Fig. 2. Three-way variable interactions in a BN. (a) X and Y are each a parent or a child of C; (b) both X and Y are parents of C; (c) X is a common child of C and Y, while Y is a spouse of C; and (d) Y is a non-child descendant or a non-parent ancestor of C.

4.2 Output of the MRMR Approach and the MB of C

Here, we will link the objective function of MRMR to the learning of the MB of C. From Equation (8), we can see that Equation (8) only considers three variables, C, the variable in S, and the variable outside S. Figure 2 shows the three-way variable interactions in a BN. Using these interactions in Figure 2 and the relationship of mutual information and conditional independence discussed in Section 3.2, the output of MRMR based on Equation (8) is analyzed as follows:
K. Yu et al.

(i.e., \( C \perp Y | X \)) and \( I(C; X|Y) > 0 \) (i.e., \( X \perp C | Y \)), by \( I(C; X) - I(C; X|Y) = I(C; Y) - I(C; Y|X) \), \( I(C; X) > I(C; Y) \) holds. Then, \( X \) will be added to \( S \) first. When \( X \) is added to \( S \), does \( \{X, Y\} \) provide more prediction information than \( \{X\} \)? By \( I(C; (X, Y)) = I(C; X) + I(C; Y|X) \), we get \( I(C; (X, Y)) = I(C; X) \), thus given \( X, Y \) does not provide any information about \( C \). As \( X \) is added to \( S \), we get \( I(C; Y) = I(C; Y|X) = I(C; Y|X) + I(C; Y|X) = I(C; Y|X) \). Then, \( Y \) cannot be added to the output of MRMR. Then, for a non-parent ancestor or a non-child descendant \( Y \), the parent(s) or child(ren) of \( C \) on the path(s) from \( Y \) to \( C \) always has (have) larger mutual information with \( C \), so the parent(s) or child(ren) will be added first, and once added, it is impossible for \( Y \) to be added, because \( Y \) is independent \( C \) given the parent(s) or child(ren). However, when the sample size is small or noise data samples exist, non-parent ancestors or non-child descendants of \( C \) may be added to \( S \) prior to parents and children of \( C \). The MRMR approach has a strong capability to mitigate problems caused by noise or small-sized data samples, since it calculates pairwise mutual information between features without conditioning on other features.

In summary, from the analysis above, we can see that the output of MRMR, i.e., the feature selection based on Equation (8), prefers parents and children of \( C \). Thus, now we have established a theoretical link that bridges the gap between filtering feature selection and local BN structure learning (i.e., parent and children (PC) learning), providing a promising way to significantly improve computational efficiency of large BN structure learning by making use of efficient filtering feature selection methods to learn the PC set of each variable in a dataset.

### 4.3 Instantiation of the MRMR Approach

There are two well-known algorithms that instantiate the MRMR approach, the mRMR (max-Relevance and Min-Redundancy) [30] and FCBF (Fast Correlation-Based Filter) algorithms [40]. The mRMR algorithm needs to specify the number of PC of a target variable in advance. In fact, we do not always have the prior knowledge and thus it is not an easy task to determine a suitable value for the parameter. In contrast, FCBF does not need to specify such a parameter. So, in our proposed algorithm (details in Section 5), we will employ FCBF to learn the PC of each variable.

FCBF has two steps, the forward step (max-relevance) and backward step (min-redundancy).

- Forward step: FCBF selects a subset of features \( S \) that \( \forall X \in S, I(C; X) > 0 \), then sorts the features in \( S \) by their mutual information with \( C \) in descending order. At this step, users always use a user-defined threshold \( \delta (\delta > 0) \) to control the size of \( S \) satisfying \( I(C; X) \geq \delta(X \in S) \) instead of \( I(C; X) = 0 \).

- Backward step: beginning with the first feature \( X \in S \), if \( \exists Y \in S \setminus \{X\} \) such that \( I(X; Y) > I(X; C) \), then \( Y \) is removed from \( S \) as a redundant feature to \( X \). The FCBF algorithm is terminated until the last feature in \( S \) is checked.

At the forward step, FCBF only selects features that are relevant to \( C \), that is, the candidate PC of \( C \). If feature \( X \) is a parent or a child of \( C \) and \( \delta = 0 \), then \( I(C; X) > \delta \) should hold. Thus, at the forward step of FCBF, if using \( \delta = 0 \), then all parents and children of \( C \) enter \( S \). If using \( \delta > 0 \), then \( S \) may include a subset of parents and children of \( C \). At the backward step of FCBF, for \( X \in S \), \( Y \in S \), according to the results in Section 4.2, we have the following analysis:

First, if \( X \) and \( Y \) are each a parent or a child of \( C \) (e.g., \( X \) and \( Y \) in Figures 2(a) and (b)), then \( I(X; C) > I(X; Y) \) and \( I(Y; C) > I(X; Y) \) hold. Thus, both \( X \) and \( Y \) cannot be used to remove each other from \( S \) at the backward step of FCBF.

Second, if \( Y \) is a spouse of \( C \), i.e., both \( Y \) and \( C \) are parents of \( X \) (e.g., \( Y \) in Figure 2(c)), then \( I(C; X) > I(C; Y) \) holds and \( X \) is put before \( Y \) in \( S \). Since \( I(C; Y) < I(X; Y) \), then \( Y \) is removed from \( S \) at the backward step of FCBF.
Third, if \( Y \) is an ancestor or a descendant of \( C \) (e.g., \( Y \) in Figure 2(d)), then \( I(C; X) > I(C; Y) \) and \( I(C; Y) < I(X; Y) \) hold. Since \( X \) is put before \( Y \) in \( S \), \( Y \) is removed from \( S \).

Thus, in theory, FCBF prefers parents and children of \( C \). At the forward step of FCBF, FCBF can use \( \delta \) to control the size of \( S \), but users do not need to specify the number of selected features in advance.

5 THE PROPOSED ALGORITHMS

Based on the analysis in Section 4 of the MRMR approach and the commonly used algorithms instantiated from MRMR, we can see that the MRMR approach can speed up computing the PC of a variable by decomposing \( I(X; C|S) \) into a linear combination of low-order mutual information terms. In the section, by employing the FCBF algorithm for PC learning, we propose the F2SL approach to local-to-global BN structure learning with the following three steps:

— Step 1: Learn the PC (parents and children) set of each variable in \( V \) using FCBF;
— Step 2: Construct the DAG skeleton by combining all individual variables’ PC sets;
— Step 3: Orient edges in the skeleton obtained at Step 2.

At Step 2, F2SL constructs the global DAG skeleton (i.e., undirected graph) by combining all the PC sets of individual variables learnt at Step 1 using the symmetry constraint. The symmetry constraint in a BN means that if there exists an edge between \( V_i \) and \( V_j \), then \( V_j \) should be a parent or a child of \( V_i \). That is to say, the PC set of \( V_i \) should contain \( V_j \) and the PC set of \( V_j \) also should contain \( V_i \). For example, in Figure 1, since “Smoking” has an edge with “Lung Cancer”, “Smoking” is a parent of “Lung Cancer” while “Lung Cancer” is a child of “Smoking.” With all the PC sets of individual variables learnt at Step 1, F2SL constructs the global skeleton using the symmetry constraint that if the learnt PC set of \( V_i \) contains \( V_j \) and the learnt PC set of \( V_j \) contains \( V_i \), then F2SL adds an undirected edge between \( V_i \) and \( V_j \). At Step 3, F2SL uses either score functions or conditional independence tests to orient edges in the global skeleton obtained at Step 2.

At Step 3, using score functions or independence tests to orient edges, we instantiate the F2SL approach into two algorithms, F2SL-s and F2SL-c, respectively.

5.1 The F2SL-s Algorithm

The F2SL-s algorithm is shown in Algorithm 1. It first learns the PC set of each variable in \( V \) using FCBF, then constructs the DAG skeleton using the learnt PC sets. Finally it orients edges in the DAG skeleton using a score function and a hill climbing greedy search algorithm, which is the same as MMHC for edge orientation. Thus, the key difference between F2SL-s and MMHC is that F2SL-s uses FCBF to learn a DAG skeleton while MMHC employs the MMPC algorithm [35].

5.2 F2SL-c Algorithm

Different from F2SL-s, the F2SL-c algorithm (Algorithm 2) employs independence tests for edge orientation at Step 3. Identifying all v-structures in the DAG skeleton is the first and key step to orient edges using independence tests. Thus, for a local structure of three variables, such as

---

**ALGORITHM 1: The F2SL-s Algorithm**

| 1 | Input: \( D \): dataset |
| 2 | Output: A (partially) DAG |
| 3 | Learn PC of each variable in \( V \) using FCBF; |
| 4 | Construct the DAG skeleton based on Step 1; |
| 5 | Orient edges using the BDeu score and hill climbing greedy search; |
| 6 | Output the highest score (partially) DAG. |
Fig. 3. Independence/dependence of $A$, $C$, and $D$ and v-structure.

Fig. 4. Separation set of $A$ and $D$.

**Algorithm 2: The F2SL-c Algorithm**

1. **Input:** $D$: dataset
2. **Output:** A (partially) DAG
3. Learn PC of each variable in $V$ using FCBF;
4. Construct the DAG skeleton based on Step 1;
5. Identify all v-structures in the skeleton using FindVstructure;
6. Orient the remaining edges using the Meek rules;
7. Output a (partially) DAG.

$A - C - D$, by Lemma 2 of the independence/dependence relations of a variable (e.g., $A$ and $D$) and its spouses in Section 3, to determine whether it is a v-structure or not, we need to know the separation set that makes $A$ and $D$ conditionally independent. For example, Figure 3 gives all possible DAGs corresponding to the skeleton $A - C - D$. In Figure 3(a), all three DAGs have the same independence/dependence of $A$, $C$, and $D$, i.e., $A \perp D$ and $A \perp D | C$. The DAG in Figure 3(b) satisfies $A \perp D$ and $A \not\perp D | C$, thus it is a v-structure and the separation set of $A$ and $D$ is an empty set.

The separation sets are learned simultaneously with the DAG skeleton using existing MB and PC learning algorithms, however, they have to be learned separately using the MRMR approach based on Equation (8). Since MRMR calculates pairwise mutual information between features without conditioning on other features, it cannot output the separation set of each pair of features. This is a caveat of the fast MRMR approach. To tackle this problem, we propose the FindVstructure algorithm (Algorithm 3) to learn separation sets to identify v-structures in a skeleton.

Then, based on the FindVstructure algorithm, the F2SL-c algorithm is described in Algorithm 2. Before introducing the FindVstructure algorithm, let us have a look at the different situations/types of separation sets of two variables in a DAG skeleton.

**Situation 1: the separation set is an empty set.** In Figure 3(a), $A$ and $D$ are dependent but become independent given $C$, and $A - C - D$ is not a v-structure. However, in Figure 3(b), $A$...
**ALGORITHM 3: The FindVstructure Algorithm**

| Line | Code |
|------|------|
| 1    | Input: Skeleton-DAG |
| 2    | Output: Vstructure-DAG |
| 3    | for any local structure $A - C - D$ in Skeleton-DAG do |
| 4    | if $A \perp D|C$ then |
| 5    | continue; |
| 6    | if $A \perp D \land A \perp D|C$ then |
| 7    | Orient $A - C - D$ as $A \rightarrow C \leftarrow D$; |
| 8    | continue; |
| 9    | end |
| 10   | if $\exists S \subseteq PC_A \setminus \{C\}$ such that $A \perp D|S$ and $A \perp D|S \cup \{C\}$ then |
| 11   | Orient $A - C - D$ as $A \rightarrow C \leftarrow D$; |
| 12   | end |
| 13   | end |

and $D$ are independent but dependent given $C$, and $A - C - D$ is a v-structure. In this case, the separation set of $A$ and $D$ is an empty set.

- **Situation 2: the separation set is a subset of the PC set of $A$ or $D$.** In Figure 4(a), $A$, $C$, and $D$ do not form a v-structure. Thus, $A \not\perp D|C$ and $A \perp D|C$, $E$ hold. In Figure 4(b), $A$, $C$, and $D$ form a v-structure, then $A \perp D|E$ and $A \not\perp D|C$, $E$. In this case, both $E$ and $C$ are the common PC of $A$ and $B$. In Figure 4(c), $A \perp D|E$ and $A \perp D|G$. In this case, $E \in PC(D)$ and $G \in PC(A)$. In Figure 4(c), the separation set of $A$ and $D$ is a subset of the PC set of $A$ or $D$.

From the above discussions, we see that for a local structure $A - C - D$, a separation set of $A$ and $D$ in a DAG skeleton will be a subset of the PC set of $A$ or $D$. Thus, we do not need to search for the separation set for $A$ and $D$ from the union of the PC sets of $A$ and $D$. With these observations, we propose the FindVstructure algorithm (Algorithm 3) to find separation sets and to determine v-structures. For a skeleton $A - C - D$ in a DAG skeleton, instead of searching for the separation set for $A$ and $D$ from the union of the PC sets of $A$ and $D$, FindVstructure divides this search into the following three steps:

- Step 1 (Lines 3–5). If $A \perp D|C$ holds, then $A - C - D$ is not a v-structure. Otherwise, go to Step 2.
- Step 2 (Lines 6–9). If $A \perp D$ and $A \not\perp D|C$ hold, then $A - C - D$ is oriented as a v-structure. Otherwise, go to Step 3.
- Step 3 (Lines 10–12). Assume $PC(A)$ is the PC set of $A$. Step 3 tests all subsets of $PC(A)$ (or $PC(D)$) excluding $C$. Once finding a subset $S$ that makes both $A \perp D|S$ and $A \not\perp D|S \cup \{C\}$ hold, $A - C - D$ is oriented as a v-structure. Otherwise, Step 3 continues until all subsets within $PC(A)$ are checked.

### 5.3 Analysis of the Computational Complexity

In this section, we use $|V|$ to denote the size of the entire set of variables and $|PC|$ to denote the maximum size of the PC set among all variables. The core computation parts of F2SL lie in Step 1 (learning the PC set of each variable) and Step 3 (edge orientations). Since only MMHC and F2SL achieve the tradeoff between time efficiency and structure correctness and both algorithms adopt the local-to-global BN structure learning strategy, we analyze the computational complexity of MMHC and F2SL mainly using the number of mutual information computations. At the step of learning the PC set of each variable, using Equation (6), the computational complexity of MMHC is $O(|V||PC|^2|PC|)$ while using Equation (8), the computational complexity of F2SL is reduced to $O(|V|^2|PC|)$. For Step 3 of orienting edges, F2SL is instantiated by F2SL-s using the score-based strategy and F2SL-c using the constraint-based strategy, respectively. F2SL-s uses the same
scoring search strategy as MMHC for edge orientations, thus they have the similar computational complexity at Step 3. Thus, F2SL-s is always more efficient than MMHC. At Step 3, F2SL-c first needs to identify the unshielded triple \(<X_i, X_k, X_j>\), then finds the separation set for determining whether this unshielded triple is a \(v\)-structure. The computational complexity of F2SL-c at Step 3 is \(O(|M|^2|PC|)\). In fact, the number of unshielded triples (denoted as \(|M|\)) in the skeleton of a DAG is always much smaller than the number of features in the skeleton. F2SL-c is often more efficient than MMHC, especially when the number of variables is large, which is verified in the experiments.

6 EXPERIMENTS

In this section, we will systematically evaluate the F2SL-s and F2SL-c algorithms. In Section 6.1, we describe the datasets, comparison methods, and evaluation metrics in the experiments. In Section 6.2, we report the results of F2SL-s and F2SL-c with the five representative local-to-global BN structure learning algorithms, and in Section 6.3, we analyze why F2SL-s and F2SL-c are better than the five rivals.

6.1 Experiment Setting

To evaluate the F2SL-s and F2SL-c algorithms, we use two groups of data generated from the six benchmark BNs as shown in Table 1. One group includes 10 datasets each with 500 data instances, and the other group also contains 10 datasets each with 1,000 data instances. All experiments were conducted on a computer with Intel(R) i7-8700, 3.2 GHz CPU, and 16 GB memory. The significance level for independence tests is set to 0.01. In all tables in Section 6, the symbol “-” denotes that an algorithm does not produce results when the running time of the algorithm exceeded 48 hours. We compare the F2SL-s and F2SL-c algorithms against the following five representative local-to-global BN structure learning algorithms:

- GSBN [24]. The GSBN algorithm first uses the GSMB algorithm to learning the MB of each variable for constructing the skeleton of a BN, then employs conditional independence tests to orient edges.
- MMHC [37]. The MMHC algorithm first employs the MMPC algorithm to learning the PC of each variable for constructing the skeleton of a BN, then uses a score function and a hill climbing greedy search method to orient edges.3
- MMHC-c. The MMHC-c algorithm is proposed by us. Compared to MMHC, MMHC-c uses MMPC to learn a DAG skeleton, then it uses conditional independence tests to orient edges.
- SLL+C/G [26]. The SLL+C and SLL+G algorithms learn the MB of each variable using a score-based MB learning algorithm, then SLL+C uses conditional independence tests to orient edges while SLL+G employs score functions to orient edges.4
- GGSL [12]. The GGSL algorithm starts to learn a local BN structure of a randomly selected variable using a score-based MB algorithm, then gradually expands the learnt structure until the entire structure is learnt.
- NOTEARS [45]. The NOTEARS algorithm learns a global BN structure using continuous optimization and score functions.
- DAG-GNN [41]. The DAG-GNN algorithm learns a global BN structure using graph neural networks.

1 The source codes of F2SL-s/c are available at http://bigdata.ahu.edu.cn/causal-learner
2 The datasets are available at http://pages.mtu.edu/~lebrown/supplements/mmhc_paper/mmhc_index.html
3 The source codes of MMHC are available at http://bigdata.ahu.edu.cn/causal-learner
4 The source codes of SLL+C/G are available at https://www.cs.helsinki.fi/u/tzniinim/uai2012
Table 1. Summary of Benchmark BNs

| Network | # Vars | # Edges | Max In/Out-Degree | Min/Max | Variable Range |
|---------|--------|---------|-------------------|---------|----------------|
| Mildew  | 35     | 46      | 3/3               | 1/5     | 3–100          |
| Hailfinder | 56     | 66      | 4/16              | 1/17    | 2–11           |
| Munin   | 189    | 282     | 3/15              | 1/15    | 1–21           |
| Pigs    | 441    | 592     | 2/39              | 1/41    | 3–3            |
| Link    | 724    | 1,125   | 3/14              | 0/17    | 2–4            |
| Gene    | 801    | 972     | 4/1               | 0/11    | 3–5            |

Using the two groups of datasets, in addition to running time (in seconds), we evaluate our algorithms in structure learning quality against the eight rivals from two aspects, structure errors and structure correctness. The metrics of structural errors (i.e., number of extraneous edges) are described below.

— **SHD (Structural Hamming Distance)**: the sum of the values of Miss, Extra, and Reverse as follows:
  — Miss: the number of missing edges in the network structure learnt by the algorithm against the true network structure.
  — Extra: the number of extra edges in the learnt BN.
  — Reverse: the number of edges with wrong directions according to the true structure.

The metrics of structural correctness (i.e., number of correctly oriented edges) are Ar_Precision, Ar_Recall, and F1 described as follows:

— **Ar_Recall** = \( \frac{\text{correctly predicted arrowheads}}{\text{predicted arrowheads}} \).
  The number of correctly predicted arrowheads in the output divided by the number of edges in the output of an algorithm.

— **Ar_Precision** = \( \frac{\text{correctly predicted arrowheads}}{\text{true arrowheads}} \).
  The number of correctly predicted arrowheads in the output divided by the number of true arrowheads in a test DAG.

— **Ar_F1** = \( F_1 = 2 \times (\text{Ar_Precision} \times \text{Ar_Recall}) / (\text{Ar_Precision} + \text{Ar_Recall}) \).
  Compared to SHD, **Ar_F1** not only considers extraneous edges, but also correct edges.

In the experiments, for an algorithm, we report the average results of these metrics over the 10 datasets in each group.

### 6.2 Results of Bayesian Network Structure Learning

In this section, we will report the experimental results of F2SL-c and F2SL-s vs. the eight rivals in terms of time efficiency and quality of learned structures, respectively.

1. **Time efficiency**. The last columns in Tables 2 and 3 show that F2SL-c and F2SL-s are significantly faster than GSBN, MMHC, SLL+C, SLL+G, MMHC-C, GGSL, NOTEARS, and DAG-GNN in both groups of datasets. GGSL, SLL+C, NOTEARS, DAG-GNN, and SLL+G are not scalable to a large size of a BN network. Based on the normalized running time results shown in Table 5, F2SL-s is up to 15, 15, 50,800, 32,408, and 58,065 times faster than GSBN, MMHC, SLL+C, SLL+G, and GGSL, respectively. Figures 5 and 6 show that the running time of GSBN and MMHC increases with the size of BNs networks while F2SL-c and F2SL-s are much more scalable.

2. **Structure errors**. Tables 2 to 3 show the BN quality by SHD of F2SL-c and F2SL-s against the eight rivals.
Table 2. Summaries of Wrongly Learnt Edges and Time Efficiency for the Datasets of Group 1 (size = 500)

| Network   | Algorithm | SHD  | Reverse | Miss  | Extra | Time  |
|-----------|-----------|------|---------|-------|-------|-------|
| Mildew    | GSNB      | 256.20 ± 3.22 | 3.90 ± 1.52 | 246.80 ± 3.01 | 5.30 ± 1.63 | 5.43 ± 0.05 |
|           | MMHC      | 322.90 ± 7.23 | 59.70 ± 3.68 | 168.10 ± 4.23 | 95.10 ± 5.02 | 31.42 ± 2.35 |
|           | MMHC-c    | 381.10 ± 12.78 | 45.40 ± 3.93 | 178.80 ± 5.18 | 156.90 ± 13.08 | 8.11 ± 0.92 |
|           | SLL-C     | –     | –       | –     | –     | –     |
|           | SLL+G     | –     | –       | –     | –     | –     |
|           | GSBN      | 297.70 ± 12.51 | 70.30 ± 7.54 | 216.60 ± 6.22 | 10.80 ± 2.62 | 33.55 ± 0.80 |
|           | MMHC      | 19.90 ± 8.43 | 8.10 ± 4.95 | 0.00 ± 0.00 | 11.70 ± 3.80 | 67.49 ± 0.84 |
|           | MMHC-c    | 243.50 ± 7.11 | 105.90 ± 8.46 | 0.00 ± 0.00 | 13.60 ± 4.88 | 36.96 ± 0.60 |
|           | SLL-C     | –     | –       | –     | –     | –     |
|           | SLL+G     | –     | –       | –     | –     | –     |
|           | GSBN      | 1,154.50 ± 13.87 | 158.10 ± 8.16 | 911.10 ± 9.70 | 85.30 ± 9.75 | 174.41 ± 8.49 |
|           | MMHC      | 1,198.60 ± 26.68 | 130.40 ± 23.04 | 722.40 ± 15.90 | 345.80 ± 22.65 | 187.42 ± 5.67 |
|           | MMHC-c    | 1,801.00 ± 28.03 | 266.70 ± 18.57 | 591.90 ± 14.58 | 942.40 ± 25.22 | 117.52 ± 11.01 |
|           | SLL-C     | –     | –       | –     | –     | –     |
|           | SLL+G     | –     | –       | –     | –     | –     |
|           | GSBN      | 1,314.20 ± 72.19 | 233.30 ± 9.76 | 695.50 ± 9.59 | 385.40 ± 70.82 | 67.79 ± 13.82 |
|           | MMHC      | 1,159.90 ± 21.57 | 79.50 ± 9.95 | 72.40 ± 13.02 | 356.00 ± 22.90 | 24.22 ± 1.02 |
|           | MMHC-c    | 153.10 ± 11.62 | 62.40 ± 9.09 | 46.50 ± 3.31 | 44.20 ± 3.55 | 183.39 ± 1.31 |
|           | SLL-C     | –     | –       | –     | –     | –     |
|           | SLL+G     | –     | –       | –     | –     | –     |
|           | GSBN      | 59.30 ± 7.96 | 16.20 ± 3.91 | 34.90 ± 4.53 | 8.20 ± 1.32 | 29.61 ± 5.37 |
|           | MMHC      | 112.80 ± 9.76 | 54.60 ± 7.76 | 49.40 ± 6.02 | 8.80 ± 3.12 | 23.70 ± 0.64 |

(1) F2SL-c and F2SL-s against GSBN. We can see that for both groups of datasets (with 500 and 1,000 samples, respectively), F2SL-c and F2SL-s are inferior to GSBN on the Hailfinder, Munin, and Link networks, since the values of both Extra and Reverse of GSBN are much smaller than those of F2SL-c and F2SL-s. GSBN employs the GSMB algorithm to find the MB of each feature for each network.
Table 3. Summaries of Wrongly Learnt Edges and Time Efficiency for the Datasets of Group 2 (Size = 1,000)

| Network | Algorithm | SHD | Reverse | Miss | Extra | Time |
|---------|-----------|-----|---------|------|-------|------|
| Mildew | GSBN      | 41.00 ± 1.05 | 4.00 ± 0.47 | 36.90 ± 0.88 | 0.10 ± 0.32 | 0.22 ± 0.00 |
|         | MMHC      | 39.40 ± 2.99 | 8.60 ± 1.26 | 25.50 ± 1.18 | 5.30 ± 1.34 | 0.49 ± 0.03 |
|         | MMHC-c    | 33.60 ± 1.58 | 7.20 ± 0.79 | 21.40 ± 0.84 | 5.00 ± 1.25 | 0.28 ± 0.01 |
|         | SLL+c     | 33.80 ± 0.84 | 7.20 ± 0.45 | 22.60 ± 0.55 | 4.00 ± 0.00 | 6,867.33 ± 1,970.90 |
|         | SLL+c+G   | 39.00 ± 1.73 | 12.20 ± 1.92 | 22.80 ± 0.45 | 4.00 ± 0.00 | 5,533.74 ± 1,640.63 |
|         | GGSL      | -         | -         | -         | -         | -         |
|         | NOTEARS   | 53.10 ± 1.45 | 11.30 ± 0.67 | 10.40 ± 0.97 | 31.40 ± 2.07 | 185.71 ± 9.99 |
|         | DAG-GNN   | 53.90 ± 8.10 | 10.60 ± 0.70 | 11.30 ± 2.16 | 32.00 ± 7.27 | 947.51 ± 91.00 |
|         | F2SL-c    | 24.80 ± 2.66 | 5.60 ± 1.43 | 15.00 ± 0.82 | 4.20 ± 1.14 | 0.13 ± 0.01 |
|         | F2SL-s    | 52.40 ± 2.12 | 10.60 ± 1.71 | 20.60 ± 0.52 | 1.20 ± 0.79 | 0.15 ± 0.01 |
|         | SLLBN     | 80.80 ± 0.79 | 0.10 ± 0.32 | 60.60 ± 0.00 | 20.70 ± 0.67 | 8.85 ± 0.02 |
|         | GSBN      | 246.90 ± 3.21 | 6.50 ± 1.27 | 232.50 ± 2.42 | 7.90 ± 1.85 | 6.57 ± 0.06 |
|         | MMHC      | 302.50 ± 4.62 | 66.30 ± 3.89 | 151.30 ± 2.45 | 8.40 ± 3.60 | 45.35 ± 3.51 |
|         | MMHC-c    | 479.60 ± 14.21 | 67.40 ± 2.95 | 141.30 ± 3.27 | 279.60 ± 13.69 | 14.39 ± 0.87 |
|         | SLL+c     | -         | -         | -         | -         | -         |
|         | SLL+c+G   | -         | -         | -         | -         | -         |
|         | GSBN      | 2,882.40 ± 272.50 | 53.90 ± 5.57 | 145.20 ± 6.49 | 2,283.30 ± 272.13 | 11,362.21 ± 576.29 |
|         | MMHC      | 291.20 ± 8.08 | 12.90 ± 2.23 | 215.30 ± 8.15 | 63.00 ± 3.24 | 12,552.36 ± 685.19 |
|         | SLL+c     | 252.70 ± 3.40 | 24.30 ± 4.55 | 160.40 ± 1.71 | 68.00 ± 3.13 | 2.98 ± 0.09 |
|         | SLL+c+G   | 309.10 ± 9.85 | 65.30 ± 5.98 | 150.30 ± 3.77 | 93.50 ± 3.54 | 3.05 ± 0.10 |
| Pigs    | GSBN      | 280.70 ± 8.54 | 65.30 ± 8.92 | 298.80 ± 7.74 | 6.60 ± 0.97 | 37.94 ± 0.53 |
|         | MMHC      | 9.00 ± 4.50  | 5.00 ± 3.62 | 0.00 ± 0.00 | 4.00 ± 2.05 | 74.09 ± 1.00 |
|         | MMHC-c    | 207.70 ± 11.56 | 65.30 ± 6.79 | 0.00 ± 0.00 | 148.30 ± 7.10 | 44.70 ± 1.04 |
|         | SLL+c     | -         | -         | -         | -         | -         |
|         | SLL+c+G   | -         | -         | -         | -         | -         |
|         | GSBN      | 29.40 ± 8.24 | 5.20 ± 2.74 | 17.70 ± 5.19 | 6.50 ± 2.51 | 13.90 ± 0.25 |
|         | MMHC      | 23.90 ± 7.09 | 5.40 ± 2.95 | 11.20 ± 4.05 | 7.30 ± 2.50 | 8.93 ± 0.03 |
| Link    | GSBN      | 1,093.50 ± 11.40 | 163.80 ± 5.83 | 828.80 ± 9.58 | 102.30 ± 7.30 | 178.85 ± 4.69 |
|         | MMHC      | 1,120.70 ± 32.66 | 138.20 ± 23.43 | 650.00 ± 6.45 | 332.50 ± 14.81 | 200.64 ± 3.76 |
|         | MMHC-c    | 1,766.90 ± 29.21 | 222.70 ± 15.90 | 546.20 ± 5.39 | 998.00 ± 16.72 | 129.53 ± 5.22 |
|         | SLL+c     | -         | -         | -         | -         | -         |
|         | SLL+c+G   | -         | -         | -         | -         | -         |
|         | GSBN      | 1,237.10 ± 21.00 | 196.20 ± 15.50 | 666.60 ± 6.40 | 374.30 ± 12.54 | 65.66 ± 2.99 |
|         | MMHC      | 1,149.20 ± 16.59 | 95.50 ± 11.61 | 663.70 ± 6.53 | 390.00 ± 12.44 | 32.77 ± 1.68 |
|         | MMHC-c    | 113.00 ± 12.02 | 58.80 ± 9.96 | 28.60 ± 3.34 | 25.60 ± 4.17 | 190.79 ± 0.96 |
|         | SLL+c     | -         | -         | -         | -         | -         |
|         | SLL+c+G   | -         | -         | -         | -         | -         |
|         | MMHC      | 586.20 ± 21.19 | 226.70 ± 15.11 | 3.00 ± 0.82 | 356.50 ± 11.94 | 91.08 ± 13.82 |
|         | SLL+c     | -         | -         | -         | -         | -         |
|         | SLL+c+G   | -         | -         | -         | -         | -         |
|         | GSBN      | 50.70 ± 5.23 | 11.50 ± 2.84 | 33.40 ± 2.95 | 5.80 ± 0.92 | 45.51 ± 4.76 |
|         | MMHC      | 94.70 ± 10.07 | 52.50 ± 10.38 | 35.00 ± 2.58 | 7.20 ± 1.62 | 29.40 ± 0.27 |

constructing the DAG skeleton. Since GSMB uses the entire feature subset currently selected for computing Equation (6), the number of data samples required by GSMB is exponential to the size of the MB. In Hallfinder, Munin, and Link networks, for both groups of datasets, the size of the MB of each feature selected by GSMB is much small. As a result, GSBN misses many true edges (i.e.,
Using both groups of datasets, we have the following observations: F2SL algorithms learned a DAG skeleton better than MMHC. Both overall SHD lower than F2SL-c and F2SL-s.

### Table 4. Comparison of Correctly Learnt Edge Directions

| Network | Algorithm | Size = 500 | Size = 1,000 |
|---------|-----------|------------|--------------|
|         |           | F1 Precision | Recall | F1 Precision | Recall |
| GSBN    | 0.16 ± 0.02 | 0.46 ± 0.05 | 0.10 ± 0.01 | 0.18 ± 0.03 | 0.55 ± 0.07 | 0.11 ± 0.02 |
| MMHC    | 0.20 ± 0.03 | 0.30 ± 0.03 | 0.15 ± 0.02 | 0.33 ± 0.05 | 0.46 ± 0.07 | 0.26 ± 0.04 |
| MMHC-c  | 0.40 ± 0.05 | 0.60 ± 0.06 | 0.30 ± 0.04 | 0.46 ± 0.03 | 0.53 ± 0.03 | 0.38 ± 0.03 |
| SLL+C   | 0.26 ± 0.05 | 0.38 ± 0.07 | 0.19 ± 0.03 | 0.44 ± 0.02 | 0.59 ± 0.02 | 0.35 ± 0.02 |
| SLL+G   | 0.21 ± 0.08 | 0.32 ± 0.12 | 0.16 ± 0.06 | 0.30 ± 0.05 | 0.40 ± 0.07 | 0.24 ± 0.04 |
| GGSL    | 0.14 ± 0.05 | 0.28 ± 0.09 | 0.09 ± 0.04 | - | - | - |
| NOTEARS | 0.42 ± 0.02 | 0.34 ± 0.02 | 0.52 ± 0.03 | 0.43 ± 0.01 | 0.36 ± 0.01 | 0.53 ± 0.02 |
| DGG-NMN | 0.48 ± 0.03 | 0.44 ± 0.02 | 0.53 ± 0.04 | 0.43 ± 0.05 | 0.36 ± 0.05 | 0.52 ± 0.05 |
| F2SL-c  | 0.34 ± 0.07 | 0.38 ± 0.08 | 0.31 ± 0.07 | 0.63 ± 0.05 | 0.72 ± 0.06 | 0.55 ± 0.04 |
| F2SL-s  | 0.27 ± 0.05 | 0.41 ± 0.07 | 0.20 ± 0.04 | 0.41 ± 0.03 | 0.56 ± 0.07 | 0.32 ± 0.04 |

- **Mildew**
- **Hallfinder**
- **Munin**
- **Pigs**
- **Link**
- **Gene**

has higher Miss values), but smaller Extra and Reverse of GSBN than F2SL-c and F2SL-s make it overall SHD lower than F2SL-c and F2SL-s.

(2) **F2SL-c and F2SL-s against MMHC and MMHC-c.** Using both groups of datasets, we have the following observations: F2SL algorithms learned a DAG skeleton better than MMHC. Both
Fig. 5. Running time with varying variables using sample size 500 (each point denotes running time of an algorithm on a BN in Table 1).

Fig. 6. Running time with varying variables using sample size 1,000 (each point denotes running time of an algorithm on a BN in Table 1).

F2SL-c and MMHC-c have the same orientation process, and the difference in their performance lies in learning skeletons. The BN structures learnt by MMHC-c have much more extra edges than those learnt by F2SL-c, but have slightly less missing edges than the BN structures learnt by F2SL-c. This indicates that F2SL finds more accurate skeletons than MMHC. Edge orientation in F2SL-c (by a conditional independence test) is better than edge orientation in MMHC (by a score function). The edge orientation by a scoring function greedily deletes edges during edge orientation, and the deletion reduces the performance of BN learning, since this edge deletion procedure not only deletes wrong edges, but may also remove correct edges. MMHC and F2SL-s employ the same orientation strategy. The skeleton learnt by F2SL is better than that learnt by MMHC, as we discussed above, but after edge orientations, the performance of MMHC and F2SL-s are similar and the reduced performance of F2SL-s is due to edge deletion. To confirm this, we compare F2SL-c, which does not delete edges in the orientation process, with MMHC. On the Mildew, Hailfinder, Munin, and Gene networks, F2SL-c achieves better SHD than MMHC, and this confirms that deleting edges hurts the overall performance. Pigs is a special dataset for MMHC that achieves 0 missing edges (100% recall rate, as shown in Table 6 in Section 6.3). Such an exceptional performance does not repeat in other datasets.
(3) **F2SL-c and F2SL-s against SLL+C/G.** Both SLL+C and SLL+G only produce the results on the Mildew and Hailfinder networks due to expensive computational costs. We can see that the SHD values of both SLL+C and SLL+G are very close to those of F2SL-c and F2SL-s, but SLL+C and SLL+G are very computationally expensive.

(4) **F2SL-c and F2SL-s against NOTEARS and DAG-GNN.** From Tables 2 to 3, we can see that NOTEARS and DAG-GNN are computationally expensive and are not scalable to large and/or high dimensional datasets, thus they do not produce the results on the Pigs, Link, and Gene networks. NOTEARS and DAG-GNN often achieve a smaller number of missing edges but a much larger number of extra edges than F2SL-c and F2SL-s. In small BN networks, the SHD values of DAG-GNN are competitive with those of F2SL-c and F2SL-s.

3. **Structural correctness.** Table 4 reports the BN structure quality by Ar_ReCALL, Ar_Precision, and Ar_F1 of F2SL-c and F2SL-s and their rivals. Although on the Hailfinder, Munin, and Link networks, GSBN achieves less structure errors than F2SL-c and F2SL-s in terms of SHD, it is inferior to F2SL-c/s on Ar_F1 with both groups of data, as shown in Table 4. As discussed above, GSMB employed by GSBN finds much smaller sized MBs than FCBF used by F2SL-c and F2SL-s, and thus GSMB has missed more true edges than F2SL-c and F2SL-s, and GSBN achieves much lower value of Ar_ReCALL than F2SL-c and F2SL-s.

F2SL-c and F2SL-s are better than its eight rivals on all networks except for the Pigs network. F2SL-c and F2SL-s learn more correct edges than its eight rivals. From Tables 2 and 3, F2SL-s and MMHC are very competitive in terms of Miss, thus they are comparable on Ar_F1, as shown in Table 4. The Pigs network is a very special for MMHC, which achieves 100% recall (as shown in Table 6).

From Table 4, we can see that F2SL-c and F2SL-s achieve larger Recall than NOTEARS and DAG-GNN (except for the Mildew network using 500 samples), although NOTEARS and DAG-GNN get smaller Miss than F2SL-c and F2SL-s (as shown in Tables 2 and 3). This is because NOTEARS and DAG-GNN have large orientation errors. Overall, F2SL-s and F2SL-s achieve better Ar_F1 than NOTEARS and DAG-GNN.

4. **Simultaneous comparison of time efficiency and structure quality.** In BN structure learning, an algorithm may be chosen to sacrifice structure learning quality for computational efficiency, while another algorithm may be chosen to sacrifice time efficiency for structure learning quality. Therefore, it is interesting to compare two algorithms in terms of both time efficiency and quality of structure learning at the same time.

For this comparison, we normalize SHD, Ar_F1, and running time reported in Tables 2 to 4. Normalized SHD (or Ar_F1 or running time) is the value of SHD (or Ar_F1 or running time) of an algorithm for a particular sample size and network divided by the SHD (or Ar_F1 or running time) of F2SL-s on the same sample size and network. The normalized results are shown in Table 5. A normalized SHD or running time greater than one implies that the algorithm is worse or slower than F2SL-s on the same learning task, while a normalized Ar_F1 smaller than one implies that the algorithm is worse than F2SL-s on the same learning task.

Using the normalized results in Table 5, we explore the tradeoff between structure quality (i.e., SHD or Ar_F1) and time efficiency of each algorithm. Figure 7 shows the logarithm of the normalized time versus the logarithm of the normalized SHD for the two groups of datasets with sample sizes 500 and 1,000, respectively, and Figure 8 illustrates the logarithm of the normalized time vs. the normalized Ar_F1 for the two groups of datasets with sample sizes 500 and 1,000, respectively. Each point in Figures 7 and 8 denotes the performance in terms of the two metrics of a given algorithm on learning one of the six BN networks. Since we are using normalized measures with respect to the measures of F2SL-s, the performance measures of F2SL-s always fall on point (1, 1) and thus are not indicated in the figures.
We can see that in Figure 7, there are no algorithms (except for F2SL-c) that fall in the grey area, indicating that no algorithms outperform F2SL-s in terms of both running time and SHD. In terms of SHD, F2SL-s and F2SL-c are very competitive with the other algorithms in most cases, while no algorithms are faster than F2SL-s and F2SL-c.

Figure 8 illustrates that no algorithms (except for F2SL-c) fall in the grey area, indicating that no algorithms are better than F2SL-s in terms of both running time and Ar_F1, except for F2SL-c. GGSL is better than F2SL-s only in one case in the left figure of Figure 8, and SLL+C is superior to F2SL-s in one case in the right figure of Figure 8. Except for the two cases, no other algorithms outperform F2SL-s and F2SL-c.

In summary, in terms of time efficiency, F2SL-c and F2SL-s are significantly faster than GSMB, MMHC, GGSL, SLL+C, SLL+G, NOTEARS, and DAG-GNN. Both of them achieve competitive performance against their eight rivals in terms of the metrics of structural errors. In terms of the metrics of structural correctness, F2SL-c and F2SL-s are better than their five rivals. In the following, we will analyze why F2SL-c and F2SL-s are better than these rivals.

6.3 Why Are the Proposed Algorithms Better?

The efficiency and quality of MB or PC learning for each feature are the key to local-to-global BN learning algorithms. Among F2SL-c, F2SL-s, and their five rivals, GSBN employs the GSMB algorithm for learning the MB (not the PC) of each feature to constructing skeletons, whereas MMHC, SLL+C, SLL+G, GGSL, F2SL-c, and F2SL-s learn the PC set of each feature for constructing the skeletons. GSBN employs a different method to learn the skeleton of a BN from MMHC, SLL+C, SLL+G, GGSL, F2SL-c, and F2SL-s. In this section, we evaluate the performance of the PC learning algorithms used by F2SL-c, F2SL-s, MMHC, SLL+C, SLL+G, and GGSL to find out why F2SL-c and
Table 5. Comparison of Structure Learning Methods on Different Groups of Datasets in Terms of Normalized Metrics

| Network       | Size = 500 | Size = 1,000 |
|---------------|------------|--------------|
|               | Mildew     | Hailfinder   | Munin | Pigs | Link | Gene |
| F2SL-s        | 1.00       | 1.00         | 1.00  | 1.00 | 1.00 | 1.00 |
| F2SL-c        | 1.64       | 0.73         | 0.90  | 1.45 | 2.80 | 1.25 |
| GSBN          | 1.07       | 2.42         | 2.09  | 4.62 | 7.20 | 14.58 |
| MMHC          | 2.86       | 6.08         | 12.08 | 9.30 | 7.74 | 7.74 |
| MMHC-c        | 1.71       | 2.54         | 3.12  | 5.09 | 4.85 | 3.20 |
| SLL+C         | 50,880.71  | 189.92       |       |      |      |      |
| SLL+G         | 32,408.79  | 123.04       |       |      |      |      |
| GGSL          | 58,065.21  | 1,420.50     |       |      |      |      |
| NOTEARS       | 1,308.36   | 618.27       | 4,274.26 | 3,770.66 |      |
| DAG-GNN       | 3,568.29   | 4,685.96     | 3,936.27 |      |

| Network       | Size = 500 | Size = 1,000 |
|---------------|------------|--------------|
|               | Mildew     | Hailfinder   | Munin | Pigs | Link | Gene |
| Normalized Time ↓ |
| F2SL-s        | 1.00       | 1.00         | 1.00  | 1.00 | 1.00 | 1.00 |
| F2SL-c        | 1.26       | 1.00         | 1.42  | 1.00 | 0.59 | 1.05 |
| GSBN          | 0.59       | 0.82         | 0.79  | 0.66 | 0.24 | 0.68 |
| MMHC          | 0.74       | 0.91         | 0.92  | 1.04 | 0.85 | 0.98 |
| MMHC-c        | 1.48       | 1.00         | 0.88  | 0.79 | 0.59 | 0.67 |
| SLL+C         | 0.96       | 0.92         |       |      |      |      |
| SLL+G         | 0.78       | 0.64         |       |      |      |      |
| GGSGL         | 0.52       | 0.64         |       |      |      |      |
| NOTEARS       | 1.56       | 0.36         | 0.21  | 0.15 |      |      |
| DAG-GNN       | 1.78       | 0.09         | 0.50  |      |      |      |

| Network       | Size = 500 | Size = 1,000 |
|---------------|------------|--------------|
|               | Mildew     | Hailfinder   | Munin | Pigs | Link | Gene |
| Normalized SHD ↓ |
| F2SL-s        | 1.00       | 1.00         | 1.00  | 1.00 | 1.00 | 1.00 |
| F2SL-c        | 1.26       | 1.00         | 1.42  | 1.00 | 0.59 | 1.05 |
| GSBN          | 0.59       | 0.82         | 0.79  | 0.66 | 0.24 | 0.68 |
| MMHC          | 0.74       | 0.91         | 0.92  | 1.04 | 0.85 | 0.98 |
| MMHC-c        | 1.48       | 1.00         | 0.88  | 0.79 | 0.59 | 0.67 |
| SLL+C         | 0.96       | 0.92         |       |      |      |      |
| SLL+G         | 0.78       | 0.64         |       |      |      |      |
| GGSGL         | 0.52       | 0.64         |       |      |      |      |
| NOTEARS       | 1.56       | 0.36         | 0.21  | 0.15 |      |      |
| DAG-GNN       | 1.78       | 0.09         | 0.50  |      |      |      |

| Network       | Size = 500 | Size = 1,000 |
|---------------|------------|--------------|
|               | Mildew     | Hailfinder   | Munin | Pigs | Link | Gene |
| Normalized Ar_F1 ↑ |
| F2SL-s        | 1.00       | 1.00         | 1.00  | 1.00 | 1.00 | 1.00 |
| F2SL-c        | 1.26       | 1.00         | 1.42  | 1.00 | 0.59 | 1.05 |
| GSBN          | 0.59       | 0.82         | 0.79  | 0.66 | 0.24 | 0.68 |
| MMHC          | 0.74       | 0.91         | 0.92  | 1.04 | 0.85 | 0.98 |
| MMHC-c        | 1.48       | 1.00         | 0.88  | 0.79 | 0.59 | 0.67 |
| SLL+C         | 0.96       | 0.92         |       |      |      |      |
| SLL+G         | 0.78       | 0.64         |       |      |      |      |
| GGSGL         | 0.52       | 0.64         |       |      |      |      |
| NOTEARS       | 1.56       | 0.36         | 0.21  | 0.15 |      |      |
| DAG-GNN       | 1.78       | 0.09         | 0.50  |      |      |      |

(↓ Means the smaller, the better while ↑ denotes that the bigger, the better).

F2SL-s are better. For the evaluation of the performance of the PC learning algorithms, we use precision and recall as described below (and for time-efficiency evaluation, we use running time).

- **Precision.** The number of true positives in the output of a PC learning algorithm (i.e., the number of variables in the output belonging to the true PC of a target variable) divided by the total number of variables in the output.
- **Recall.** The number of true positives in the output divided by the size of the true PC set of a variable.
- **F1.**

\[
F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.
\]

### 6.3.1 FCBF vs. Standard PC Learning Algorithm

As described in Section 5, F2SL-c and F2SL-s employ the FCBF algorithm to learn the PC of each variable for skeleton construction, and MMHC uses the MMPC algorithm [35] to find PC of each variable. SLL+C/G and GGSGL employ SLL-PC and $S^2$TMB-PC to learn the PC of each variable in a dataset for skeleton construction. In this section, in addition to MMPC, SLL-PC, and $S^2$TMB-PC, we do a comprehensive comparison of FCBF with another two widely used PC learning algorithms, HITON-PC [2] and PC-simple [22], to investigate why F2SL-c and F2SL-s are better than MMHC, SLL+C, SLL+G, and GGSGL.

Table 6 shows the PC learning quality and running time of the PC learning algorithms. Table 7 gives the normalized running time and F1. Normalized F1 or running time of an algorithm for a particular sample size and network divided by the F1 or running time of F2SL-s on the same sample size and network. In Table 7, a normalized running time greater...
than one indicates that an algorithm is slower than FCBF, while a normalized F1 smaller than one implies that an algorithm is worse than FCBF.

From Tables 6 and 7, we can see that FCBF is significantly faster than the other PC methods on both groups of datasets. This explains why F2SL-c and F2SL-s are much more efficient than the other five BN structure learning algorithms. SLL-PC and S2TMB-PC are so computationally expensive, and this explains why SLL+C, SLL+G, and GGSGL are not scalable to large-sized networks.

In PC learning quality, in terms of F1, FCBF is better than MMPC, HITON-PC, SLL-PC, and S2TMB-PC on almost all networks. FCBF achieves almost the same F1 value as PC-simple on the Hilfinder, Pigs, and Gene networks, but it is significantly better than PC-simple on the remaining networks. In terms of recall, FCBF is very competitive with its rivals while it achieves better precision than these rivals, except for SLL-PC and S2TMB-PC on the Mildew network. However, SLL-PC and S2TMB-PC are significantly worse than FCBF in terms of recall. Thus, on average, F2SL-c and F2SL-s are better than SLL+C, SLL+G, and GGSGL. On the Hilfinder network, FCBF, SLL-PC, and S2TMB-PC are very comparable, thus F2SL-c and F2SL-s are competitive with SLL+C, SLL+G, and GGSGL.

For FCBF and MMPC, MMPC calculates the high-order mutual information between X and C conditioning on a subset of the already selected features S. In the worst case, MMPC needs to explore all possible subsets of S. When the size of PC sets becomes large, MMPC will be very computationally expensive and require large data samples for reliable independence tests. A large
Table 7. Comparison of PC Learning Methods with Different Groups of Datasets in Terms of Normalized Metrics

| Network     | Mildew | Hailfinder | Munin | Pigs | Link | Gene | Size = 500 | Size = 1,000 |
|-------------|--------|------------|-------|------|------|------|-------------|--------------|
| Normalized Time ↓ |        |            |       |      |      |      | 1.00       | 1.00         |
| FCBF        | 1.00   | 1.00       | 1.00  | 1.00 | 1.00 | 1.00 | 1.00       | 1.00         |
| MMPC        | 5.14   | 13.48      | 66.12 | 10.26| 14.67| 4.35 | 10.06      | 16.26        |
| HITONPC     | 5.91   | 20.29      | 999.69| 12.67| 5.30 | 3.00 | 550.14     | 19.02        |
| PC-simple   | 294.17 | 16.93      | –     | 18.32| 6.38 | –    | 180.58     | 16.76        |
| SLL-PC      | 141,780.00 | 3,894.44  | –     | –    | –    | –    | –          | –            |
| S²TMB-PC    | 120,580.00 | 3,866.67  | –     | –    | –    | –    | –          | –            |
| Normalized F1 ↑ |        |            |       |      |      |      | 1.00       | 1.00         |
| FCBF        | 1.00   | 1.00       | 1.00  | 1.00 | 1.00 | 1.00 | 1.00       | 1.00         |
| MMPC        | 0.90   | 0.92       | 0.95  | 0.79 | 0.86 | 0.83 | 0.83       | 0.82         |
| HITONPC     | 0.86   | 1.00       | 0.36  | 0.79 | 0.86 | 0.79 | 0.83       | 0.82         |
| PC-simple   | 0.46   | 1.00       | –     | 1.03 | –    | 1.00 | 0.43       | 1.00         |
| SLL-PC      | 0.90   | 0.92       | –     | –    | –    | –    | 1.08       | –            |
| S²TMB-PC    | 0.92   | 0.92       | –     | –    | –    | –    | 1.08       | –            |

(↓ Means the smaller, the better while ↑ denotes that the bigger, the better).

size of PC sets and/or small-sized data samples will make MMPC impractical in many real-world applications. Just as we discussed in Section 4, FCBF can deal with small and/or high-dimensional datasets, since FCBF computes the pairwise mutual information between X and C without conditioning on other features.

Table 6 has shown that FCBF achieves higher precision rate than and almost the same recall rate as MMPC although FCBF only uses pairwise mutual information to learn an approximate PC set of a variable. When the size of a PC set is large (e.g., in Munin) or a BN network has a large number of variables (e.g., in Gene), FCBF is much better than MMPC on both recall and precision measures. In Tables 2 and 3 in Section 6.2, we observed that a higher recall achieved by MMPC does not transfer to smaller missing edges in BN structures learnt by MMHC in comparison with F2SL-c. FCBF achieves higher precision than MMPC. This means that the output of MMPC contains more false PCs than FCBF and hence the BN structures learnt by MMHC and MMHC-c should have more extra edges than those learnt by F2SL-c. But MMHC achieves a competitive number of extra edges with F2SL-c (except for the Gene network) due to edge deletion at the edge-orientation step, while the BN structures learnt by MMHC-c have more extra edges and less missing edges than those learnt by F2SL-c on all networks except for the Mildew network due to non-edge deletion. In summary, for local-to-global structure learning, regardless of which method—score-based or constraint-based—is used for edge orientations, it is crucial to find a correct DAG skeleton.

6.3.2 Experimental Analysis of the Threshold $\delta$ for FCBF. Although FCBF does not require users to specify the number of selected features before learning, it needs a user-specified parameter $\delta$ to control the number of candidate parents and children of a variable of interest at the forward step of FCBF as described in Section 4.3. A large $\delta$ makes FCBF remove false positives as well as true positives. A small $\delta$ leads FCBF to find more true positives and more false positives as well.

To choose an optimal value of $\delta$ for FCBF, we have investigated the F1, recall, and precision of FCBF using different $\delta$ value from 0 to 0.1 with the six BNs in Table 1. In Figure 9, we can see that when $\delta$ is small, FCBF gets a low value of F1. As $\delta$ increases, the value of F1 gradually becomes high and stable. The explanation is that when $\delta$ is small, as shown in Figure 10, many false positives will enter the final output of FCBF; this reduces the precision of FCBF. However, Figure 11 illustrates that the value of $\delta$ does not have much impact on the true PC entering the output of FCBF, since parents and children of a variable should have a strong dependency relationship with the variable and they are less likely be discarded even when $\delta$ becomes big. As $\delta$ increases, some true positives will be discarded. Using the six BNs, we can see that when $\delta$ is up to 0.05, FCBF is stable and almost...
Fig. 9. F1 of FCBF on the six BNs when varying the value of \( \delta \) from 0 to 0.1 with different groups of datasets. The red line denotes \( \delta = 0.05 \).

Fig. 10. Precision of FCBF on the six BNs when varying the value of \( \delta \) from 0 to 0.1 with different groups of datasets. The red line denotes \( \delta = 0.05 \).

Fig. 11. Recall of FCBF on the six BNs when varying the value of \( \delta \) from 0 to 0.1 with different groups of datasets. The red line denotes \( \delta = 0.05 \).

gets the highest F1, and thus in all the experiments in Section 6, we choose 0.05 as the value of \( \delta \) for FCBF.

7 CONCLUSION

In this article, we link feature selection methods to BN structure learning to improve computational efficiency and propose the F2SL framework for BN structure learning by using the FCBF algorithm. By instantiating the F2SL framework, we propose two efficient local-to-global structure learning
algorithms, F2SL-c and F2SL-s. Using six benchmark BNs, the experimental results have shown that F2SL-c and F2SL-s significantly improve computational efficiency of BN structure learning compared to the eight state-of-the-art BN structure learning algorithms and both also achieve competitive structure learning quality with the eight rivals.

REFERENCES
[1] Constantin F. Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D. Koutsoukos. 2010. Local causal and Markov blanket induction for causal discovery and feature selection for classification part I: Algorithms and empirical evaluation. J. Mach. Learn. Res. 11, Jan. (2010), 171–234.
[2] Constantin F. Aliferis, Ioannis Tsamardinos, and Alexander Statnikov. 2003. HITON: A novel Markov blanket algorithm for optimal variable selection. In AMIA Annual Symposium Proceedings, Vol. 2003. American Medical Informatics Association, 21.
[3] Concha Bielza and Pedro Larrañaga. 2014. Bayesian networks in neuroscience: A survey. Front. Comput. Neurosci. 8 (2014), 131.
[4] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. 2012. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. J. Mach. Learn. Res. 13 (2012), 27–66.
[5] Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, and Weiru Liu. 2002. Learning Bayesian networks from data: An information-theory based approach. Artif. Intell. 157, 1-2 (2002), 43–90.
[6] David Maxwell Chickering. 2002. Optimal structure identification with greedy search. J. Mach. Learn. Res. 3, Nov. (2002), 507–554.
[7] David Maxwell Chickering, David Heckerman, and Christopher Meek. 2004. Large-sample learning of Bayesian networks is NP-hard. J. Mach. Learn. Res. 5, Oct. (2004), 1287–1330.
[8] Diego Colombo and Marloes H. Maathuis. 2014. Order-independent constraint-based causal structure learning. J. Mach. Learn. Res. 15, 1 (2014), 3741–3782.
[9] Cassio P. de Campos, Mauro Scanagatta, Giorgio Corani, and Marco Zaffalon. 2018. Entropy-based pruning for learning Bayesian networks using BIC. Artif. Intell. 260 (2018), 42–50.
[10] Luis M. de Campos, Juan M. Fernández-Luna, and Juan F. Huete. 2004. Bayesian networks and information retrieval: An introduction to the special issue. Inf. Process. Manag. 40, 5 (2004), 727–733.
[11] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pé’er. 2000. Using Bayesian networks to analyze expression data. J. Comput. Biol. 7, 3–4 (2000), 601–620.
[12] Tian Gao, Kshitij Fadnis, and Murray Campbell. 2017. Local-to-global Bayesian network structure learning. In International Conference on Machine Learning (ICML’17). 1193–1202.
[13] Tian Gao and Qiang Ji. 2016. Efficient Markov blanket discovery and its application. IEEE Trans. Cybern. 47, 5 (2016), 1169–1179.
[14] Tian Gao and Qiang Ji. 2017. Efficient score-based Markov blanket discovery. Int. J. Approx. Reason. 80 (2017), 277–293.
[15] Clark Glymour, Kun Zhang, and Peter Spirtes. 2019. Review of causal discovery methods based on graphical models. Front. Genet. 10 (2019), 524.
[16] Isabelle Guyon, Constantin Aliferis, and Elisseeff Andre. 2007. Causal feature selection. In Computational Methods of Feature Selection. Chapman and Hall/CRC, 79–102.
[17] David Heckerman. 2008. A tutorial on learning with Bayesian networks. In Innovations in Bayesian Networks. Springer, 33–82.
[18] Patrik O. Hoyer, Dominik Janzing, Joris M. Mooij, Jonas Peters, and Bernhard Schölkopf. 2009. Nonlinear causal discovery with additive noise models. In International Conference on Advances in Neural Information Processing Systems. 689–696.
[19] Dominik Janzing, Joris Mooij, Kun Zhang, Jan Lemeire, Jakob Zscheischler, Povilas Daniušis, Bastian Steudel, and Bernhard Schölkopf. 2012. Information-geometric approach to inferring causal directions. Artif. Intell. 182 (2012), 1–31.
[20] Daphne Koller and Nir Friedman. 2009. Probabilistic Graphical Models: Principles and Techniques. MIT Press.
[21] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. 2017. Feature selection: A data perspective. ACM Comput. Surv. 50, 6 (2017), 1–45.
[22] Jiuyong Li, Lin Liu, and Thuc Duy Le. 2015. Practical Approaches to Causal Relationship Exploration. Springer.
[23] Zhaolong Ling, Kui Yu, Hao Wang, Lin Liu, Wei Ding, and Xindong Wu. 2019. BAMB: A balanced Markov blanket discovery approach to feature selection. ACM Transactions on Intelligent Systems and Technology (TIST) 10, 5 (2019), 1–25.
[24] Dimitris Margaritis and Sebastian Thrun. 2000. Bayesian network induction via local neighborhoods. In International Conference on Advances in Neural Information Processing Systems. 505–511.

ACM Transactions on Knowledge Discovery from Data, Vol. 18, No. 2, Article 37. Publication date: November 2023.
[25] Joris M. Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. 2016. Distinguishing cause from effect using observational data: Methods and benchmarks. J. Mach. Learn. Res. 17, 1 (2016), 1103–1204.

[26] T. Niinimäki and Pekka Parviainen. 2012. Local structure discovery in Bayesian networks. In UAI’12, Workshop on Causal Structure Learning, 2012. 634–643.

[27] Judea Pearl. 2014. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann.

[28] Jean-Philippe Pellet and André Elisseeff. 2008. Using Markov blankets for causal structure learning. J. Mach. Learn. Res. 9 (2008), 1295–1342.

[29] Jose M. Peña, Roland Nilsson, Johan Björkegren, and Jesper Tegnér. 2007. Towards scalable and data efficient learning of Markov boundaries. Int. J. Approx. Reason. 45, 2 (2007), 211–232.

[30] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. 27, 8 (2005), 1226–1238.

[31] Marco Scutari, Claudia Vitolo, and Allan Tucker. 2019. Learning bayesian networks from big data with greedy search: computational complexity and efficient implementation. Statistics and Computing 29 (2019), 1095–1108.

[32] Shohei Shimizu, Patrik O. Hoyer, Aapo Hyvärinen, and Antti Kerminen. 2006. A linear non-Gaussian acyclic model for causal discovery. J. Mach. Learn. Res. 7, Oct. (2006), 2003–2030.

[33] Peter Spirtes, Clark N. Glymour, and Richard Scheines. 2000. Causation, Prediction, and Search. MIT Press.

[34] Ioannis Tsamardinos and Constantin F. Aliferis. 2003. Towards principled feature selection: Relevancy, filters and wrappers. In International Conference on Artificial Intelligence and Statistics (AISTATS’03).

[35] Ioannis Tsamardinos, Constantin F. Aliferis, and Alexander Statnikov. 2003. Time and sample efficient discovery of Markov blankets and direct causal relations. In 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 673–678.

[36] Ioannis Tsamardinos, Constantin F. Aliferis, Alexander R. Statnikov, and Er Statnikov. 2003. Algorithms for large scale Markov blanket discovery. In FLAIRS Conference.

[37] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. Mach. Learn. 65, 1 (2006), 31–78.

[38] Kui Yu, Xianjie Guo, Lin Liu, Jiuyong Li, Hao Wang, Zhaolong Ling, and Xindong Wu. 2020. Causality-based feature selection: Methods and evaluations. ACM Comput. Surv. 53, 5 (2020), 1–36.

[39] Kui Yu, Lin Liu, Jiuyong Li, Wei Ding, and Thuc Le. 2019. Multi-source causal feature selection. IEEE Trans. Pattern Anal. Mach. Intell. DOI: 10.1109/TPAMI.2019.2908373

[40] Lei Yu and Huan Liu. 2004. Efficient feature selection via analysis of relevance and redundancy. J. Mach. Learn. Res. 5 (2004), 1205–1224.

[41] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. 2019. DAG-GNN: DAG structure learning with graph neural networks. In International Conference on Machine Learning. 7154–7163.

[42] Kun Zhang and Aapo Hyvärinen. 2012. On the identifiability of the post-nonlinear causal model. arXiv preprint arXiv:1205.2599 (2012).

[43] Kun Zhang, Bernhard Schölkopf, Peter Spirtes, and Clark Glymour. 2018. Learning causality and causality-related learning: Some recent progress. Nat. Sci. Rev. 5, 1 (2018), 26–29.

[44] Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. 2019. D-VAE: A variational autoencoder for directed acyclic graphs. In International Conference on Advances in Neural Information Processing Systems. 1588–1600.

[45] Xun Zheng, Bryon Aragam, Pradeep K. Ravikumar, and Eric P. Xing. 2018. DAGs with NO TEARS: Continuous optimization for structure learning. In International Conference on Advances in Neural Information Processing Systems. 9472–9483.

Received 6 May 2022; revised 26 April 2023; accepted 6 September 2023.