Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2012, Article ID 974063, 17 pages
doi:10.1155/2012/974063

Research Article

A Decomposition Algorithm for Learning Bayesian Networks Based on Scoring Function

Mingmin Zhu and Sanyang Liu

Department of Mathematics, Xidian University, Xi'an 710071, China

Correspondence should be addressed to Mingmin Zhu, zmmzhu2010@126.com

Received 14 May 2012; Revised 12 August 2012; Accepted 28 August 2012

Academic Editor: B. V. Rathish Kumar

Learning Bayesian network (BN) structure from data is a typical NP-hard problem. But almost existing algorithms have the very high complexity when the number of variables is large. In order to solve this problem(s), we present an algorithm that integrates with a decomposition-based approach and a scoring-function-based approach for learning BN structures. Firstly, the proposed algorithm decomposes the moral graph of BN into its maximal prime subgraphs. Then it orientates the local edges in each subgraph by the K2-scoring greedy searching. The last step is combining directed subgraphs to obtain final BN structure. The theoretical and experimental results show that our algorithm can efficiently and accurately identify complex network structures from small data set.

1. Introduction

Bayesian networks (BNs), also known as belief networks, are becoming a popular tool for representing uncertainty in artificial intelligence. They have been applied to a wide range of tasks such as natural spoken dialog systems, vision recognition, expert systems, medical diagnosis, and genetic regulatory network inference [1–5]. A BN consists of two important components: a directed acyclic graph (DAG) representing the dependency structure among the variables and a conditional probability table (CPT) for each variable given its parent set. There has been a lot of work in the last ten years on the learning of BNs for both graph structure and probability parameters. However, learning the structure is harder and, arguably, more critical [6–8]. Most of these algorithms can be grouped into two different categories: constraint-based methods and search-and-score methods. The algorithms based on constraint generate a list of conditional independence (CI) relationships among the variables in the domain and attempt to find a network that represents these relationships as far as possible [9–11]. The number, complexity, and reliability of the required independence

tests are the main concerns regarding this type of algorithms. The algorithms based on the score-and-search paradigm see the learning task as a combinatorial optimization problem, where a search method operates on a search space associated with BNs. They evaluate the degree of fitness between each element and the available data by a scoring function [12–14]. Among the search-and-score algorithms, the K2 algorithm [15] is commonly used which learns the network structure from data by requiring a prior ordering of nodes as input. However, searching for the best structure is difficult because the search space increases exponentially with the number of variables [16].

In this paper, we propose a BN structure learning algorithm, which combined with the merits of the constraint-based method and the K2 algorithm. The proposed algorithm not only employs constraint knowledge to decompose the search space but also uses the K2 score as heuristic knowledge to induce the process of local greedy search. It uses the property that the local information of variables in each maximal prime sub-graph cannot be destroyed by decomposing the undirected independence graph. At the same time, the K2 algorithm can be used to learn the local structure of each undirected subgraph and obtain a group of directed subgraphs. Thus, by combining these directed subgraphs, we obtain the final structure. Theoretical results and large number of experiments show that the new algorithm is effective and efficient, especially in small data set. Moreover, the proposed algorithm uses maximal prime decomposition to identify the whole graph structure, which reduces the search space significantly and greatly enhances learning speed.

The remainder of this paper is organized as follows. Section 2 introduces notation and definitions. We describe our algorithm and its theoretical proofs in Section 3. Section 4 discusses how to construct the moral graph from observed data. Simulation studies are conducted to demonstrate the performance of our algorithm and existing algorithms in Section 5. Finally, in Section 6, we conclude and outline our future work. All proofs will be presented in the Appendix section.

2. Notation and Definitions

In this section, we provide some basic technical terminologies that are sufficient for understanding this paper.

A BN is a tuple (G, P), where $G = (V, E)$ is a directed acyclic graph (DAG) with nodes representing the random variables in V and P a joint probability distribution on V. A node V_i is called a parent of V_j if the directed edge $V_i \rightarrow V_j \in E$. The set of all parents of V_j is denoted as $pa(V_j)$. A path from V_i to V_j is a sequence $[V_i = V_{i0}, V_{i1}, \ldots, V_{in} = V_j]$ of distinct nodes such that $V_{i(k-1)} \rightarrow V_{ik}$ or $V_{ik} \rightarrow V_{i(k-1)}$ for $k = 1, 2, \ldots, n$. We say that V_j is an ancestor of V_i and V_i is a descendant of V_j if there is a directed path from V_j to V_i in G. In addition, G and P must satisfy the Markov condition: every variable $V_i \in V$ is independent of any subset of its nondescendant variables conditioned on the set of its parents, denoted by $pa(V_i)$. We denote the conditional independence of the variable sets X and Y given Z in some distribution P with $\mathrm{Ind}_P(X; Y \mid Z)$. A path ρ is said to be d-separated by a set Z in a DAG G, if and only if (1) ρ contains a "head-to-tail meeting": $V_i \rightarrow V_j \rightarrow V_k$ or a "tail-to-tail meeting": $V_i \leftarrow V_j \rightarrow V_k$ such that the middle node V_j is in Z, or (2) ρ contains a "head-to-head meeting": $V_i \rightarrow V_j \leftarrow V_k$ such that the middle node V_j is not in Z and no descendant of V_j is in Z. Two distinct sets X and Y of nodes are said to be d-separated by a set Z in G if Z d-separates every path from any node in X to any node in Y. We use $D\mathrm{sep}_G(X; Y \mid Z)$ to denote the assertion that X is d-separated from Y given Z in G.

For a DAG $G = (V, E)$, its moral graph $G^m = (V, E^m)$ is an undirected graph obtained by connecting every pair of nodes with a common child which are not already connected in G and then dropping the directionality of all directed edges. An undirected path ρ is said to be separated by a set Z in a moral graph G^m, if and only if ρ passes through Z. A BN (G, P) satisfies the faithfulness condition if the d-separations in G identify all and only the conditional independencies in P, that is, $\mathrm{Ind}_P(X; Y \mid Z)$ if and only if $D\mathrm{sep}_G(X; Y \mid Z)$. We will drop the subscript G or P in the notation of conditional independence when the faithfulness condition holds. Let $G(V)$ denote a graph consisting of a finite set V of nodes.

Definition 2.1 (see [17]). Suppose the triplet (V', S, V'') denote a partition of V where $V = V' \cup S \cup V''$; if every path in $G^m(V)$ between $V_i \in V'$ and $V_j \in V''$ contains a node in S and $G^m(S)$ is complete, then G^m is decomposable and (V', S, V'') is a decomposition of G^m into subgraphs $G^m(V' \cup S)$ and $G^m(V'' \cup S)$; otherwise G^m is prime. Moreover, S is called a complete separator of G^m with respect to (V', S, V'').

Furthermore, an undirected graph G^m is said to be decomposed maximally if G^m and all its sub-graphs can be decomposed recursively until all sub-graphs are not decomposable. These sub-graphs are the maximal prime sub-graphs of G^m. Transformation of a BN into its maximal prime sub-graphs is equivalent to recursively decomposing the moral graph of DAG. The most well-known algorithm is the MPD-JT algorithm [17] which first triangulates the moral graph G^m by adding a fill-in edge to every cycle whose length is greater than three and then identifies all the cliques of triangulation graph and arranges them as a junction tree (JT), and finally, recursively aggregates cliques connected by incomplete separators (in the moral graph). Figure 2 schematically illustrates the process of the MPD-JT algorithm.

Example 2.2. Consider the Asia network [18] in Figure 1(a). Its moral graph G^m is shown in Figure 1(b). By our definition, G^m is decomposable. Figure 1(c) provides one of its triangulation graphs of G^m and Figure 2(a) gives its corresponding junction tree. Since $G^m(BL)$ is an incomplete separator in the moral graph G^m, we aggregate cliques *BLS* and *BEL* according to the MPD-JT algorithm. The resulting graph is shown in Figure 2(b). From Figure 2(b), we can see that all the separators are complete in G^m, so, there are no cliques that need to be aggregated. We obtain five maximal prime sub-graphs (Figure 2(c)).

The K2 algorithm is a greedy search algorithm that learns the network structure from data. It attempts to select the network structure which maximizes the network's posterior probability, given the observed data. Its scoring function can be expressed as follows:

$$f_{K2}(G, D) = \log p(G) + \sum_{i=1}^{n} \left[\sum_{j=1}^{q_i} \left[\log \left(\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \right) + \sum_{k=1}^{r_i} \log(N_{ijk}!) \right] \right]. \tag{2.1}$$

N_{ijk} is the number of cases in D where V_i is in its kth state and its parents are in their jth state. $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ denotes the number of cases in D where $pa(V_i)$ is in its jth state, r_i denotes the number of states of variable V_i, and q_i is the number of parent configurations of V_i. It is obvious that the search space is the main element influencing the performance of the K2 algorithm. In theory, the more constraint conditions, the smaller the search space of BNs. And then the search efficiency will be much higher. Hence, it is very necessary to

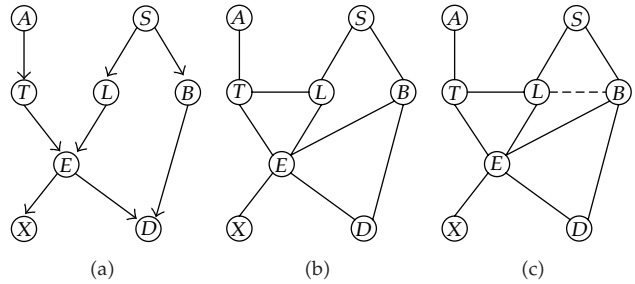
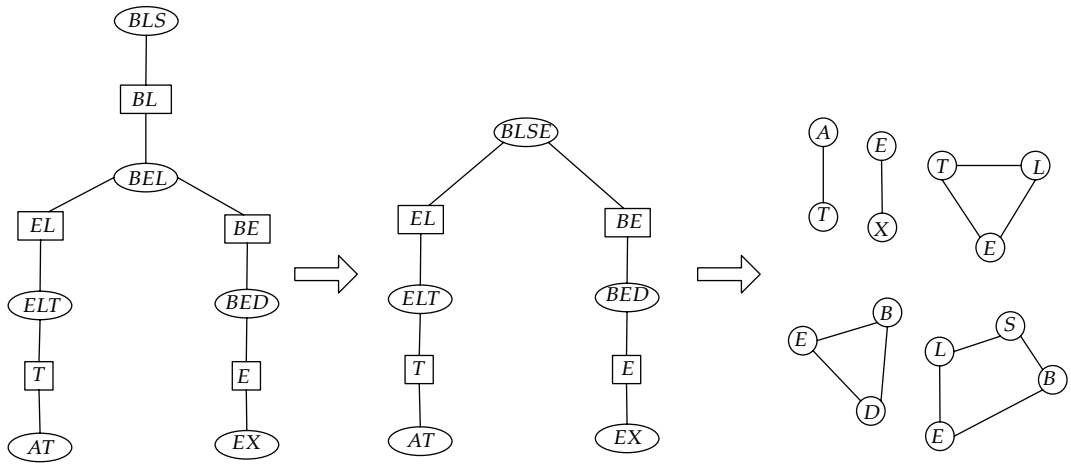


Figure 1: Asia network, its moral graph, and one of its triangulation graphs.



(a) JT of Asia (b) Aggregation of *BLS* and *BEL* (c) Maximal prime subgraphs of Asia

Figure 2: The process of decomposing the Asia moral graph into its maximal prime subgraphs.

reduce the search space. Considering the BN's own characteristic, we combine the constraint approach with maximal prime decomposition to learn a BN structure.

3. The Structure Learning Algorithm and Its Correctness

In this section, theoretical results are presented for learning the structure of BN. We show how the problem of learning the structure over the full set of all variables can be split into its subproblems. Below we first give two theorems based on which we propose the decomposition algorithm for structural learning of BNs.

Theorem 3.1. *Let G^m be a moral graph for a DAG $G = (V, E)$. Then a subset of variables $S \subset V$ d-separates $V_i \in V$ from $V_j \in V$ in G if and only if S separates V_i from V_j in G^m.*

By Theorem 3.1, we can know that the condition of d-separation in a DAG is equivalent to the separation in its moral graph.

(1) **Input**: Data set D; Variable set $V = (V_1, V_2, \ldots, V_n)$; Node ordering ρ.
(2) Construct the moral graph $G^m = (V, E^m)$ from the data set D.
(3) Decompose the moral graph G^m into its maximal prime subgraphs $G_1^m, G_2^m, \ldots, G_k^m$.
(4) For each sub-graph G_l^m $(l = 1, 2, \ldots, k)$ of G^m, call the K2 algorithm with the local node ordering of G_l^m to construct a directed acyclic graph G_l $(l = 1, 2, \ldots, k)$.
(5) Combine $G_1, G_2, \ldots, G_k$ into a directed graph $G = (V, E)$ where $V = V(G_1) \cup V(G_2) \cup \cdots \cup V(G_k)$ and $E = E(G_1) \cup E(G_2) \cup \cdots \cup E(G_k)$.
(6) **Output**: A directed acyclic graph G.

Algorithm 1: Decomposition algorithm for learning BNs.

Example 3.2. Consider the Asia network in Figure 1(a). By the definition of d-separation, we can see that A and X are d-separated by E, because the path $A \to T \to E \to X$ is blocked at E. Since there is a path $A \to T \to E \leftarrow L \leftarrow S \to B \to D$ and E is a head-to-head node, A and D are not d-separated by E. Thus, we can conclude that in the moral graph of Asia network in Figure 1(b), A and X are separated by E and E does not separate A from D. In fact, it is obviously held in Figure 1(b).

Theorem 3.3. *Let G^m be a moral graph for a DAG $G = (V, E)$ and let $G_1^m, G_2^m, \ldots, G_k^m$ be maximal prime sub-graphs of G^m. For any variable $V_i \in V$, there exists at least one maximal prime sub-graph of G^m which contains V_i and $Pa(V_i)$.*

A consequence of Theorem 3.3 is that a problem of structural learning from a data set can be decomposed into problems of local structural learning from a subdata set. The guarantee of such approach is that each subdata must contain sufficient data information of variable and its parent set.

Based on the previous analysis, an improved algorithm was proposed combined with the merits of previous two basic methods for BN structure learning. First, according to the observed data or domain knowledge, we construct the independent graph (the moral map) of the target DAG applying constraint-based approach and then decompose the independent graph. It is shown that each maximal prime sub-graph contains the sufficient information of local variables, so, the search space of the score function can be effectively reduced. Second, the local structure of each sub-graph is learned by using the score function and a directed acyclic graph is obtained for each sub-graph. Finally, we combine all these directed acyclic sub-graphs. Theoretical and experimental results show that the new algorithm is effective and reasonable. Now we formalize our algorithm in Algorithm 1.

As shown in Algorithm 1, the proposed algorithm first decomposed the entire variable set into its subsets. Then the local directed graph of each subset is recovered by K2 algorithm. Unlike the case of the X-G-Z algorithm [19], the final result of our algorithm is a DAG, not a partial directed graph, and the procedure of finding minimal d-separators is avoided. Furthermore, the computational complexity of Algorithm 1 is less than that of the K2 algorithm. In fact, the triangulation used to construct a junction tree from an undirected independence graph is the main cost of the MPD-JT algorithm. Although the problem of optimally triangulating an undirected graph is NP hard, suboptimal triangulation methods may be used provided that the obtained tree does not contain too large cliques. Two most well-known algorithms are the lexicographic search [20] and the maximum cardinality search [21], and their complexities are $O(ne)$ and $O(n + e)$, respectively, where n is the number

(1) **Input**: Data set D; Variable set $V = (V_1, V_2, \ldots, V_n)$; Target variable T.
(2) Initialization: $\overline{\mathrm{PC}} = V \setminus T, \overline{\mathrm{SP}} = \emptyset$.
(3) Order-0 CI test: for each variable $V_i \in \overline{\mathrm{PC}}$, if $\mathrm{Ind}(V_i; T)$ is hold, then $\overline{\mathrm{PC}} = \overline{\mathrm{PC}} \setminus V_i$, $S(V_i) = \emptyset$.
(4) Order-1 CI test: for each variable $V_i \in \overline{\mathrm{PC}}$, if there is a variable $V_j \in \overline{\mathrm{PC}} \setminus V_i$ such that $\mathrm{Ind}(V_i; T \mid V_j)$, then $\overline{\mathrm{PC}} = \overline{\mathrm{PC}} \setminus V_i$, $S(V_i) = S(V_i) \cup V_j$.
(5) Find superset of spouses: for each variable $V_i \in \overline{\mathrm{PC}}$, if there is a variable $V_j \in V \setminus \{T \cup \overline{\mathrm{PC}}\}$, such that $\neg\mathrm{Ind}(V_j; T \mid S(V_j) \cup V_i)$, then $\overline{\mathrm{SP}} = \overline{\mathrm{SP}} \cup V_j$.
(6) Find parents and children of T: call the MMPC algorithm to get $\mathrm{PC}(T) = \mathrm{MMPC}\ (T, D(\overline{\mathrm{PC}} \cup \overline{\mathrm{SP}}))$. For each $V_i \in \overline{\mathrm{PC}} \setminus \mathrm{PC}(T)$, if $T \in \mathrm{MMPC}\ (V_i, D)$, then $\mathrm{PC}(T) = \mathrm{PC}(T) \cup V_i$.
(7) Find spouses of T: for each variable $V_i \in \mathrm{PC}(T)$, if there is a variable $V_j \in \mathrm{MMPC}\ (V_i, D) \setminus \{\mathrm{PC}(T) \cup T\}$ and a subset $Z \subset \overline{\mathrm{PC}} \cup \overline{\mathrm{SP}} \setminus \{T \cup V_j\}$, such that $\mathrm{Ind}(T; V_j \mid Z)$ and $\neg\mathrm{Ind}(T; V_j \mid Z \cup V_i)$, then $\mathrm{SP}(T) = \mathrm{SP}(T) \cup V_j$.
(8) Return $\mathrm{MB}(T) = \mathrm{PC}(T) \cup \mathrm{SP}(T)$.
(9) **Output**: A Markov boundary $\mathrm{MB}(T)$ of T.

Algorithm 2: Algorithm for discovering Markov boundary.

of nodes and e is the number of edges in the graph. Thus, decomposition of G^m is a computationally simple task compared to the exhaustive search using the K2 score. Let n denote the number of variables in V, and m the number of cases in data set D. In the worst case, the complexity of the K2 algorithm is $O(mn^4r)$ [15] where r is the largest number of states of a variable in V. Suppose that G^m is decomposed into k subgraphs, where $k \leq n$. Let n_0 denote the number of variables in the largest subgraph, that is, $n_0 = \max_l |G_l^m|$, where $|G_l^m|$ denotes the number of variables in G_l^m. The complexity for constructing a directed acyclic subgraph G_l in Algorithm 1 is $O(mn_0^4r)$, and thus that of all directed subgraphs is $O(kmn_0^4r)$. Since n_0 is usually much less than n, our algorithm is less computationally complex than the K2 algorithm. We now establish the correctness of our algorithm by showing that our final result is a DAG.

Theorem 3.4. *Given a data set D and a variable (node) set V, then the graph $G = (V, E)$ returned by Algorithm 1 is a DAG.*

4. Construction of the Moral Graph from Observed Data

In general, the moral graph of an underlying BN can be obtained from the observed data based on the conditional independence tests. An edge $V_i - V_j$ is included in the moral graph if and only if V_i and V_j are not conditionally independent given all other variables. However, these require sufficient data for estimating the parameters and for improving the power of tests. To avoid testing high order conditional independence relations, we propose a Markov boundary discovery algorithm which is based on a subroutine MMPC [22]. The Markov boundary of a variable T, denoted as $\mathrm{MB}(T)$, is a minimal set of variables conditioned on which all other variables that are probabilistically independent of the target T. Furthermore, the set of parents, children, and spouses of T is its unique Markov boundary under the faithfulness condition [23]. The MMPC algorithm is sketched in the appendix section.

According to Algorithm 2, we can see that our algorithm starts with a two-phase approach. A shrinking phase attempts to remove the most irrelevant variables to T, followed

by a growing phase that attempts to add as many dependent variables as possible. The growing phase is interleaved with the shrinking phase. Interleaving the two phases allows to eliminate some of the false positives in the current boundary as the algorithm progresses during the growing phase. Theoretically, the more the constraint knowledge obtained by CI tests, the smaller the search space, and the higher the searching efficiency. However, the results of higher-order CI tests may be unreliable. Steps 3 and 4 of Algorithm 2 only use order-0 and order-1 CI tests to reduce the search space whose number of CI tests is bound by $O(n^3)$. In Steps 6 and 7 of Algorithm 2, we only condition on subsets of sizes up to one instead of conditioning on all subsets of the $\overline{\text{PC}} \cup \overline{\text{SP}}$. The order of the complexity is $O(t \cdot |\text{PC}|^2)$ where PC is the largest set of parents and children over all variables in V and t is the number of variables in $\overline{\text{PC}} \cup \overline{\text{SP}}$, $t \ll n$. Thus, the total complexity of Algorithm 2 is $O(n \cdot t \cdot |\text{PC}|^2)$.

Theorem 4.1. *Suppose (G, P) satisfies the faithfulness condition, where $G = (V, E)$ is a DAG and P is a joint probability distribution of the variables in V. For each target variable T, Algorithm 2 returns the Markov boundary of T.*

By Theorem 4.1, a moral graph can be constructed from observed data.

On the other hand, G^m can be constructed based on the prior or domain knowledge rather than conditional independence tests. The domain knowledge may be experts' prior knowledge of dependencies among variables, such as Markov chains, chain graphical models, and dynamic or temporal models. Based on the domain knowledge of dependencies, data patterns of databases can be represented as a collection of variable sets $D = \{D_1, D_2, \ldots, D_k\}$, in which variables contained in the same set may associate each other directly but variables contained in different sets associate each other through other variables. This means that two variables that are not contained in the same set are independent conditionally of all other variables. From the k data patterns, we get separately k undirected subgraphs. Combining them together, we obtain the undirected graph G^m.

5. Experimental Results

In this section, we present the experimental results carried out with our algorithm on two standard network data sets (Alarm [24] and Insurance [25]). The first one is a medical diagnostic system for patient monitoring. It consists of 37 nodes and 46 edges. The random variables in the Alarm network are discrete in nature and can take two, three, or four states (Figure 3). The second example is a network for estimating the expected claim costs for a car Insurance policyholder. It consists of 27 nodes and 52 edges connecting them. Our implementation is based on the Bayesian network toolbox written by Murphy [26] and the Causal Explorer System developed by Aliferis et al. [27]. The experimental platform was a personal computer with Pentium 4, 3.06 GHz CPU, 0.99 GB memory, and Windows XP.

5.1. Learning Networks with Node Ordering

In this subsection, we show simulations of the Alarm network when the node ordering is known. Although our algorithm is based on a constraint-based method and a search-and-score method, given that the result of network returned by Algorithm 1 is a DAG, which has some similarities with the search-and-score methods, it is interesting to include one of these methods in comparison. We have selected the well-known K2 algorithm. We begin with a BN G_0 which is completely specified in terms of structure and parameters, and we obtain a data

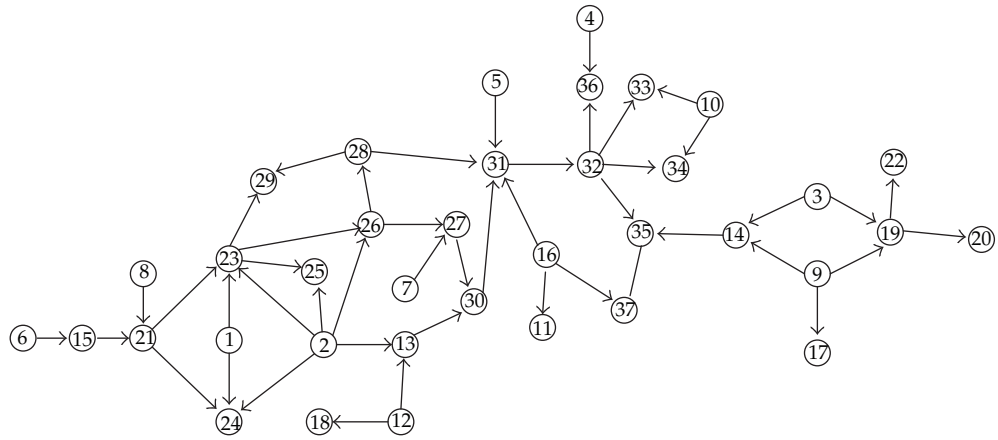


Figure 3: Alarm network.

set of a given size by sampling from G_0. Then, using our algorithm and the K2 algorithm, respectively, we obtain two learned networks G_1 and G_2, which must be compared with the original network G_0. More precisely, the experiments are carried out on different data set randomly selected from the database (100000 data points). The size of the data set is varied from 500 to 10000, and 10 replicates are done for each of the different network parameters and sample sizes. The average number of missing edges, extra edges, and reversed edges in the learned networks with respect to the original one is computed.

We compare our methods with two different significance levels on Alarm network, that is, $\alpha = 0.01$ and. The results are shown in Table 1 for different sample sizes 500, 1000, 2000, 5000, and 10000. In the second row of table, four values in a bracket denote the number of missing edges, extra edges, reversed edges, and computation time, respectively, and other rows give values relative to the second row, which are obtained by dividing their real values by the values in the second row. A relative value larger than 1 denotes that its real value is larger than the corresponding value in the second row. From Table 1, the first thing that can be observed is that these results seem to confirm our intuition about the need to use Algorithm 1 with a smaller significance level α than those typically used for independence tests, since Algorithm 1 with the value $\alpha = 0.01$ offers better results than that with $\alpha = 0.05$. It also shows that the structure obtained by our algorithm ($\alpha = 0.01$) has the least number of missing edges, extra edges, and reversed edges. And further, our algorithm costs the least time.

Table 2 displays the K2 scores obtained for the original Alarm network, network returned by Algorithm 1 and network returned by K2 algorithm, respectively. It is easy to see that the larger the dataset size is, the closer the score to the original one. However, when the dataset size is smaller than 5000, the score returned by our algorithm is more approximate to the original one. At the same time, it can be found that our algorithm performs better than the K2 algorithm in terms of the running results on all data sets. Moreover, the advantage is very obvious when the data set is small; namely, the smaller the data set size, the more obvious the improvement.

5.2. Learning Networks without Node Ordering

From the previous experimental results, we can see that our algorithm is capable of identifying structures that are close to optimal ones, given a prior ordering of the variables.

Table 1: Results relative to Algorithm 1 with $\alpha = 0.01$ and $\alpha = 0.05$: missing edges, extra edges, reversed edges, and computation time.

Algorithm (level α)	$n = 500$	$n = 1000$	$n = 2000$	$n = 5000$	$n = 10000$
	(2.4, 3.1, 1.2, 6.9951)	(1.6, 3.0, 1.1, 7.6500)	(1.2, 3.0, 1.0, 9.9092)	(0.8, 2.9, 0.9, 12.7819)	(0.6, 2.7, 0.7, 18.8523)
Algorithm 1 (0.01)	(1.0, 1.0, 1.0, 1.0000)	(1.0, 1.0, 1.0, 1.0000)	(1.0, 1.0, 1.0, 1.0000)	(1.0, 1.0, 1.0, 1.0000)	(1.0, 1.0, 1.0, 1.0000)
Algorithm 1 (0.05)	(0.9, 2.0, 1.3, 0.9199)	(0.9, 1.8, 1.3, 0.9330)	(1.0, 1.4, 1.2, 1.1031)	(1.1, 1.1, 1.2, 1.1730)	(1.2, 1.0, 1.1, 1.1883)
K2 algorithm	(1.4, 2.2, 1.3, 1.5819)	(1.3, 1.9, 1.2, 1.7235)	(1.2, 1.5, 1.2, 2.1592)	(1.2, 1.4, 1.1, 2.7847)	(1.0, 1.2, 1.0, 2.9670)

Table 2: Scores on Alarm for different dataset size n: score of original network Sc (G_0), score of network returned by Algorithm 1 Sc (G_1), and score of network returned by K2 algorithm Sc (G_2).

Score	$n = 500$	$n = 1000$	$n = 2000$	$n = 5000$	$n = 10000$
Sc (G_0)	$-5.2004e+003$	$-1.0000e+004$	$-1.9440e+004$	$-4.7737e+004$	$-9.4515e+004$
Sc (G_1)	$-4.9831e+003$	$-1.0652e+004$	$-2.0502e+004$	$-4.7558e+004$	$-9.4514e+004$
Sc (G_2)	$-6.8038e+003$	$-1.9824e+004$	$-3.0308e+004$	$-4.7826e+004$	$-9.4533e+004$

However, this ordering information may not be available in real-life applications. Thus, in this subsection, Algorithm 1 is extended to manage the problem in which the node ordering is not available. In fact, we only need to delete the directed cycles after combining all directed subgraphs at the expense of introducing a little complexity.

Now we compare our extended method with the X-G-Z [19], RAI [28], and MMHC [22] algorithms. Similar with Section 5.1, the size of the data set is varied from 1000 to 10000, and 10 replicates are done for each of the different network parameters and sample sizes. We use two significance levels ($\alpha = 0.01$ and $\alpha = 0.05$) in the simulations. Unlike the situation of Section 5.1, algorithms that return a DAG are converted to the corresponding partial directed acyclic graph (PDAG) before comparing the quality of reconstruction. The PDAG is equivalent to its DAG. The average number of missing edges, extra edges, and reversed edges in the learned PDAG with respect to the underlying PDAG is computed.

We summarize the simulation results in Tables 3 and 4. In the second row of tables, five numbers in a bracket denote the number of missing edges, extra edges, reversed edges, the sum of the first three values, and computation time, respectively. Other rows give values relative to the second row, which are obtained by dividing their real values by the values in the second row. A relative value larger than 1 denotes that its real value is larger than the corresponding value in the second row. In each column, the best of the eight results are displayed in bold. From a general point of view, we can see that the X-G-Z algorithm obtains the least number of missing edges, the MMHC obtains the least number of extra edges, and Algorithm 1 has the least number of reversed edges. In terms of the sum of the three kinds of edges, the RAI and MMHC algorithms perform better than the X-G-Z algorithm and Algorithm 1 performs best. Although it can be seen that Algorithm 1 seems to have a similar performance with RAI in most cases, the running time on all data sets cost for Algorithm 1 is least. Moreover, the advantage is very obvious when the data set is large; namely, the bigger the sample size, the more obvious the improvement. The main reason is that Algorithm 1 uses lower-order CI tests and employs maximal prime decomposition technique to effectively decompose the search space which cut down many computings of statistic factors, scorings of the structures, and comparisons of the solutions and thus greatly enhances the time performance. Contrast Table 4 with Table 3, since the Insurance network has more variables than the Alarm network, it can be found that the running time of X-G-Z fast increases as the number of variable increases. However, Algorithm 1 is not sensitive to the increase of the variable capacity. In conclusion, our algorithm has a better overall performance compared to the other state-of-the-art algorithms.

6. Conclusions

In this paper, we have given a more accurate characterization of moral graph and proposed a new algorithm for structural learning, which substantially improves on the K2 algorithm.

Table 3: Results relative to Algorithm 1: missing edges, extra edges, reversed edges, and computation time for Alarm network.

Algorithm (level α)	$n = 1000$	$n = 2000$	$n = 5000$	$n = 10000$
	(10.0, 1.3, 10.2, 21.5, 8.9908)	(6.6, 1.2, 8.9, 16.74, 17.4378)	(4.0, 0.8, 5.4, 10.2, 28.9233)	(2.6, 0.7, 2.7, 6.0, 41.4688)
Algorithm 1 (0.01)	(1.0, 1.0, **1.0**, **1.0**, **1.0000**)	(1.0, 1.0, **1.0**, **1.0**, **1.0000**)	(1.0, 1.0, **1.0**, **1.0**, **1.0000**)	(1.0, 1.0, **1.0**, **1.0**, **1.0000**)
Algorithm 1 (0.05)	(**0.8**, 4.0, 1.8, 1.4, 1.0321)	(**0.8**, 3.6, 1.5, 1.3, **0.9389**)	(**0.8**, 4.8, 1.6, 1.5, 1.1315)	(0.8, 4.9, 1.8, 1.7, 1.0356)
RAI (0.01)	(0.9, 1.6, 1.2, 1.1, 1.2879)	(1.0, 1.5, 1.1, 1.1, 1.6792)	(1.0, 1.8, 1.1, 1.1, 2.6955)	(1.0, 1.8, 1.2, 1.2, 4.4231)
RAI (0.05)	(**0.8**, 3.1, 1.5, 1.3, 1.4489)	(0.9, 2.8, 1.3, 1.3, 1.6977)	(0.9, 3.7, 1.4, 1.4, 3.6929)	(0.9, 3.7, 1.6, 1.5, 4.4982)
X-G-Z (0.01)	(**0.8**, 2.0, 1.1, **1.0**, 5.0260)	(**0.8**, 2.4, 1.1, 1.1, 2.3356)	(0.9, 3.6, 1.1, 1.2, 1.9726)	(0.9, 4.3, 1.9, 1.7, 1.6813)
X-G-Z (0.05)	(**0.8**, 2.2, **1.0**, **1.0**, 3.5679)	(0.9, 2.7, **1.0**, 1.1, 3.8967)	(**0.8**, 3.8, 1.2, 1.2, 3.5876)	(**0.7**, 4.2, 1.3, 1.4, 1.8952)
MMHC (0.01)	(1.3, **0.3**, 1.8, 1.5, 1.4798)	(1.4, **0.3**, 1.2, 1.2, 2.1321)	(1.5, **0.4**, 1.2, 1.2, 3.8258)	(1.5, **0.7**, 1.4, 1.4, 6.7052)
MMHC (0.05)	(1.2, 0.5, 1.5, 1.3, 1.7269)	(1.3, 0.4, 1.1, 1.1, 2.2036)	(1.3, 0.5, 1.2, 1.3, 5.4004)	(1.3, 1.0, 1.1, 1.2, 6.8066)

Table 4: Results relative to Algorithm 1: missing edges, extra edges, reversed edges, and computation time for Insurance network.

Algorithm (level α)	$n = 1000$	$n = 2000$	$n = 5000$	$n = 10000$
	(13.0, **2.4**, 20.6, 36.0, 2.7031)	(10.0, 1.5, 21.5, 33, 4.1406)	(7.4, 1.3, 16.3, 25.0, 6.9063)	(6.7, 1.1, 12.2, 20.0, 11.6094)
Algorithm 1 (0.01)	(**1.0**, 1.0, 1.0, **1.0**, **1.0000**)	(**1.0**, **1.0**, **1.0**, **1.0**, **1.0000**)	(**1.0**, **1.0**, **1.0**, **1.0**, **1.0000**)	(**1.0**, **1.0**, **1.0**, **1.0**, **1.0000**)
Algorithm 1 (0.05)	(1.2, 1.5, 1.0, 1.1, 1.1850)	(1.1, 1.3, **1.0**, **1.0**, 1.0981)	(**1.0**, **1.0**, 1.3, 1.1, 1.0158)	(**1.0**, 1.1, 1.1, 1.1, 1.0202)
RAI (0.01)	(**1.0**, 1.1, 1.0, 1.1, 1.2879)	(**1.0**, 1.1, **1.0**, **1.0**, 1.6792)	(**1.0**, 1.1, 1.1, 1.1, 2.6955)	(**1.0**, **1.0**, **1.0**, 1.1, 4.4231)
RAI (0.05)	(1.1, 1.3, 1.0, 1.1, 1.6636)	(1.1, 1.2, **1.0**, **1.0**, 1.9857)	(**1.0**, 1.1, 1.2, 1.2, 3.3154)	(**1.0**, 1.1, 1.1, 1.1, 5.2848)
X-G-Z (0.01)	(**1.0**, 1.6, 1.0, **1.0**, 7.2486)	(1.2, 2.3, 1.2, 1.3, 6.0151)	(1.3, 2.5, 1.3, 1.4, 7.8868)	(1.4, 2.9, 1.7, 1.7, 8.5505)
X-G-Z (0.05)	(1.1, 1.6, **0.9**, **1.0**, 9.9249)	(1.2, 2.4, 1.1, 1.2, 8.1661)	(1.3, 2.7, 1.4, 1.4, 16.0270)	(1.3, 3.0, 1.9, 1.8, 11.6998)
MMHC (0.01)	(1.2, **0.6**, 1.1, 1.1, 1.4798)	(1.4, **1.0**, 1.1, 1.2, 2.1321)	(1.4, 1.1, 1.2, 1.2, 3.8258)	(1.3, **1.0**, 1.2, 1.2, 6.7052)
MMHC (0.05)	(1.3, **0.6**, 1.1, 1.2, 1.9827)	(1.4, 1.1, 1.2, 1.3, 2.5774)	(1.5, 1.2, 1.2, 1.3, 4.8484)	(1.4, 1.1, 1.2, 1.3, 8.1278)

We also extend our algorithm to manage networks without node ordering. Although the new algorithm depends on the quality of constructed moral graph, simulation studies illustrate that our method yields good results in a variety of situations, especially when the underlying data set is small.

The results in this paper also raised a number of interesting questions for future research. We briefly comment on some of those questions here. First, maximal prime decomposition plays a key role in Algorithm 1. Although decomposition of an undirected graph into its maximal prime sub-graphs has been discussed a lot, we believe that there is room for further improvements. Second, we have applied the K2 algorithm for the learning of local structures in Algorithm 1. It will be interesting to see whether there exists some alternative approach to serve the same purpose here. Finally, although we assume in this paper that the data are completely observed, missing data or data with latent variables may arise in practice. Generalization of the proposed algorithm to missing data or data with latent variables is of great research interest.

Appendix

Proofs of Theorems

In this appendix, we give the proofs of all the theorems.

Proof of Theorem 3.1. The proof of Theorem 3.1 can be obtained from the document [29]. □

Proof of Theorem 3.3. If $Pa(V_i)$ is empty, it is trivial.

If V_i has only one parent, since no set can separate V_i from a parent, there must be a sub-graph of G^m that contains V_i and the parent. Thus we obtained the theorem.

If V_i has two or more parents, we suppose, by reduction to absurdity, that V_i has two parents V_j and V_k which are not contained in a single clique but are contained in two different cliques, say $\{V_i, V_j\} \subseteq G_P^m$ and $\{V_i, V_k\} \subseteq G_q^m$, respectively, since all variables in V appear in G^m. On the path from G_P^m to G_q^m in G^m, all separators must contain V_i; otherwise they cannot separate G_P^m from G_q^m. By Theorem 3.1, V_i d-separates V_j from V_k in G. Thus we got a contradiction. □

Proof of Theorem 3.4. It is easy to see that G is a directed graph. We need only show the absence of cycles in the graph G.

Without loss of generality, we suppose that there is a cycle from node V_p to V_q and the global node ordering is ρ_g: $V_{i_1} \prec V_{i_2} \prec \cdots \prec V_p \prec \cdots \prec V_q \prec \cdots \prec V_{i_n}$. Because each sub-graph G_l $(l = 1, 2, \ldots, k)$ returned by Step 4 is a directed acyclic graph, there exist at least two direct paths $V_p \to \cdots \to V_q$ and $V_q \to \cdots \to V_p$ which are contained in two different sub-graphs, say G_l and G_k, respectively. By the definition of global node ordering ρ_g, we conclude $V_p \prec V_q$ in the local node ordering ρ_k of graph G_k. Furthermore, according to the method in K2 algorithm for constructing sub-graph G_k with the local node ordering ρ_k, we know that the only edges pointing at the direction toward V_p are those from each variable in ρ_k preceding V_p. This contradicts the supposition that graph G_k has the direct path $V_q \to \cdots \to V_p$. □

Proof of Theorem 4.1. Before proving Theorem 4.1, we give the definition of the embedded faithfulness condition and three lemmas. □

(1) **Input**: Data set D; Variable set $V = (V_1, V_2, \ldots, V_n)$; Target variable T.
(2) CPC = $\emptyset$;
(3) **repeat**
(4) **for** $V_j \in (V \setminus \text{CPC} \setminus \{T\})$ **do**
(5) $S(V_j) = \arg \min_{Z \subseteq \text{CPC}} \text{dep}\ (T, V_j \mid Z)$;
(6) **end for**
(7) $Y = \arg \max_{V_j \in (V \setminus \text{CPC} \setminus \{T\})} \text{dep}\ (T, V_j \mid S(V_j))$;
(8) **if** $\neg\text{Ind}(T; Y \mid S(Y))$ **then**
(9) CPC = CPC $\cup \{Y\}$;
(10) **end if**
(11) **until** CPC has not changed;
(12) **for** $V_j \in$ CPC **do**
(13) **if** $\text{Ind}(T; V_j \mid Z)$ for some $Z \subseteq \text{CPC} \setminus \{T\}$ **then**
(14) CPC = CPC $\setminus V_j$;
(15) **end if**
(16) **end for**
(17) **return** CPC
(18) **Output**: The parents and children PC(T) of T.

Algorithm 3: MMPC.

Definition A.1 (see [23]). Let P^W be a joint probability distribution of the variables in W where $W \subseteq V$, and let $G = (V, E)$ be a DAG. $\langle G, P^W \rangle$ satisfies the embedded faithfulness condition if G entails all and only conditional independencies in P^W for subsets including only elements of W.

Lemma A.2. *Let P be a joint probability distribution of the variables in V with $W \subseteq V$, and let $G = (V, E)$ be a DAG. If (G, P) satisfies the faithfulness condition and P^W is the marginal distribution of W, then (G, P^W) satisfies the embedded faithfulness condition.*

The proof of Lemma A.2 can be found in [23].

Lemma A.3. *Suppose (G, P) satisfies the faithfulness condition, where $G = (V, E)$ is a DAG and P is a joint probability distribution of the variables in V. For each target variable T, $\overline{PC} \cup \overline{SP}$ returned by the Algorithm 2 is a superset of MB(T).*

Proof. By the faithfulness condition, $\text{MB}(T) = \text{PC}(T) \cup \text{SP}(T)$, where PC($T$) is the set of parents and children of T and SP(T) is the set of spouses of T. We only need to show that $\text{PC}(T) \subseteq \overline{\text{PC}}$ and $\text{SP}(T) \subseteq \overline{\text{SP}}$. If $V_i \in \text{PC}(T)$, then because of the faithfulness condition, for any subset $Z \subseteq V$, $\neg\text{Ind}(V_i; T \mid Z)$. Thus, V_i will be not removed by Steps 3 and 4 of Algorithm 2. Similarly, if $V_j \in \text{SP}(T)$, then, for each variable $V_i \in \text{PC}(T) \subseteq \overline{\text{PC}}$, there is a subset $Z \subseteq V$, $\text{Ind}(T; V_j \mid Z)$ and $\neg\text{Ind}(T; V_j \mid Z \cup V_i)$. We set $Z = S(V_j)$, where $S(V_j)$ is the d-separation set between T and V_j. By Step 5 of Algorithm 2, $V_j \in \overline{\text{SP}}$. Thus, $\text{SP}(T) \subseteq \overline{\text{SP}}$. □

Remark A.4. $\neg\text{Ind}(X; Y \mid Z)$ denotes that X is not independent of Y conditioned on Z.

Lemma A.5. *Suppose (G, P) satisfies the faithfulness condition, where $G = (V, E)$ is a DAG and P is a joint probability distribution of the variables in V with $W \subseteq V$. For each target variable T, if*

$MB(T) \subseteq W \subseteq V$, then, there is a unique Markov boundary $MB_W(T)$ of T over W and $MB_W(T) = MB(T)$.

Proof. By the faithfulness condition and Lemma A.2, it is obvious that T admits a unique Markov boundary over W. We only need to show that $\mathrm{MB}_W(T) = \mathrm{MB}(T)$ is held under the condition $\mathrm{MB}(T) \subseteq W \subseteq V$. Clearly, $\mathrm{MB}(T) \subseteq \mathrm{MB}_W(T)$ as $\mathrm{MB}(T) \subseteq W \subseteq V$. Next, we show $\mathrm{MB}_W(T) \subseteq \mathrm{MB}(T)$.

Without loss of generality, we suppose that there is a variable $V_i \in \mathrm{MB}_W(T)$, $V_i \notin \mathrm{MB}(T)$. Then, on the one hand, we would have $\mathrm{Ind}_P(V_i; T \mid \mathrm{MB}(T))$ because $\mathrm{MB}(T)$ is a Markov boundary of T in V. On the other hand, since $\mathrm{MB}_W(T) = \mathrm{PC}_W(T) \cup \mathrm{SP}_W(T)$, we consider the problem from two aspects.

If V_i is a parent or child of T in W, that is, $V_i \in PC_W(T)$, then we would not have $D\mathrm{sep}_G(V_i; T \mid \mathrm{MB}(T))$ in G. This contradicts the faithfulness condition.

If V_i is a spouse of T in W, that is, $V_i \in \mathrm{SP}_W(T)$, let V_j be their common child in W. If $V_j \in \mathrm{MB}(T)$, we again would not have $D\mathrm{sep}_G(V_i; T \mid \mathrm{MB}(T))$ in G. If $V_j \notin \mathrm{MB}(T)$, we would have $\mathrm{Ind}_P(V_j; T \mid \mathrm{MB}(T))$ in V, but we would not have $D\mathrm{sep}_G(V_j; T \mid \mathrm{MB}(T))$ in G because T is a parent of V_j in G. So again we would get a contradiction. □

Now we are ready to prove Theorem 4.1.

Proof. Suppose Step 6 of Algorithm 2 returns the parents and children of T. According to the definition of spouse, it is easy to see that Step 7 of Algorithm 2 identifies the spouse of T in G. We only need to show that Step 6 of Algorithm 2 returns all and only the parents and children of T in G.

From Lemmas A.2 and A.3, we have that $(G, P^{\overline{\mathrm{PC}}(T) \cup \overline{\mathrm{SP}}(T)})$ satisfies the embedded faithfulness condition. We set $\overline{\mathrm{PC}}(T) \cup \overline{\mathrm{SP}}(T) = W$, $W \subseteq V$. Since MMPC is correct under the faithfulness condition, Step 6 of Algorithm 2 returns the parents and children of T in W, denoted by $\mathrm{PC}_W(T)$. We next show $\mathrm{PC}_W(T) = \mathrm{PC}_V(T)$.

By Lemma A.5, we know that $\mathrm{MB}_W(T) = \mathrm{MB}_V(T) \subseteq W$, thus, $\mathrm{PC}_V(T) \subseteq \mathrm{PC}_W(T)$. Similar to proof of Lemma A.5, we show below that $\mathrm{PC}_W(T) \subseteq \mathrm{PC}_V(T)$. Without loss of generality, we suppose that there is a variable $V_i \in \mathrm{PC}_W(T)$, $V_i \notin \mathrm{PC}_V(T)$. Because all nonadjacent nodes may be d-separated in G by a subset of its Markov boundary, then $\exists Z \subseteq \mathrm{MB}_V(T) \setminus V_i$ such that $\mathrm{Ind}_P(V_i; T \mid Z)$. As $\mathrm{MB}_W(T) = \mathrm{MB}_V(T)$ owing to Lemma A.5, V_i could be d-separated in $W \setminus \{V_i, T\}$. Therefore, V_i cannot be adjacent to T in W, that is, $V_i \notin \mathrm{PC}_W(T)$. We got a contradiction. □

The MMPC algorithm is sketched in Algorithm 3.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 60974082,61075055), the National Funds of China for Young Scientists (no. 11001214), and the Fundamental Research Funds for the Central Universities (no. K5051270013).

References

[1] L. Bouchaala, A. Masmoudi, F. Gargouri, and A. Rebai, "Improving algorithms for structure learning in Bayesian Networks using a new implicit score," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5470–5475, 2010.

[2] A. Aussem and S. Rodrigues de Morais, "A conservative feature subset selection algorithm with missing data," *Neurocomputing*, vol. 73, no. 4–6, pp. 585–590, 2010.
[3] S. R. de Morais and A. Aussem, "A novel Markov boundary based feature subset selection algorithm," *Neurocomputing*, vol. 73, no. 4–6, pp. 578–584, 2010.
[4] Y. Sun, Y. Tang, S. Ding, S. Lv, and Y. Cui, "Diagnose the mild cognitive impairment by constructing Bayesian network with missing data," *Expert Systems with Applications*, vol. 38, no. 1, pp. 442–449, 2011.
[5] V. Aquaro, M. Bardoscia, R. Bellotti, A. Consiglio, F. De Carlo, and G. Ferri, "A Bayesian Networks approach to Operational Risk," *Physica A*, vol. 389, no. 8, pp. 1721–1728, 2010.
[6] J. Z. Ji, H. X. Zhang, R. B. Hu, and C. N. Liu, "Bayesian Network learning algorithm based on independence test and ant colony optimization," *Acta Automatica Sinica*, vol. 35, no. 3, pp. 281–288, 2009.
[7] Z. Geng, C. Wang, and Q. Zhao, "Decomposition of search for v-structures in DAGs," *Journal of Multivariate Analysis*, vol. 96, no. 2, pp. 282–294, 2005.
[8] X. Xie and Z. Geng, "A recursive method for structural learning of directed acyclic graphs," *Journal of Machine Learning Research*, vol. 9, pp. 459–483, 2008.
[9] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu, "Learning Bayesian networks from data: an information-theory based approach," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 43–90, 2002.
[10] J.-P. Pellet and A. Elisseeff, "Using Markov blankets for causal structure learning," *Journal of Machine Learning Research*, vol. 9, pp. 1295–1342, 2008.
[11] C. Borgelt, "A conditional independence algorithm for learning undirected graphical models," *Journal of Computer and System Sciences*, vol. 76, no. 1, pp. 21–33, 2010.
[12] E. Perrier, S. Imoto, and S. Miyano, "Finding optimal Bayesian network given a super-structure," *Journal of Machine Learning Research*, vol. 9, pp. 2251–2286, 2008.
[13] L. M. de Campos, "A scoring function for learning Bayesian networks based on mutual information and conditional independence tests," *Journal of Machine Learning Research*, vol. 7, pp. 2149–2187, 2006.
[14] N. Friedman, I. Nachmama, and D. Peér, "Learning bayesian network structure from massive datasets: the "Sparse Candidate" algorithm," in *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pp. 206–215, Stockholm, Sweden, 1999.
[15] G. Cooper and E. Hersovits, "A bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, pp. 309–347, 1992.
[16] R. W. Robinson, "Counting unlabeled acyclic digraphs," *Combinational Mathematics*, vol. 622, pp. 28–43, 1977.
[17] K. Olesen and A. Madsen, "Maximal prime sub-graph decomposition of Bayesian networks," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 32, pp. 21–31, 2002.
[18] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistical Society B*, vol. 50, no. 2, pp. 157–224, 1988, With discussion.
[19] X. C. Xie, Z. Geng, and Q. Zhao, "Decomposition of structural learning about directed acyclic graphs," *Artificial Intelligence*, vol. 170, no. 4-5, pp. 422–439, 2006.
[20] D. J. Rose, R. E. Tarjan, and G. S. Lueker, "Algorithmic aspects of vertex elimination on graphs," *SIAM Journal on Computing*, vol. 5, no. 2, pp. 266–283, 1976.
[21] R. E. Tarjan and M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs," *SIAM Journal on Computing*, vol. 13, no. 3, pp. 566–579, 1984.
[22] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," *Machine Learning*, vol. 65, no. 1, pp. 31–78, 2006.
[23] R. E. Neapolitan, *Learning Bayesian Networks*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2004.
[24] I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper, "The ALARM Monitoring System: a case study with two probabilistic inference techniques for belief networks," in *Proceedings of the 2nd European Conference Artificial Intelligence in Medicine*, 1989.
[25] J. Binder, D. Koller, S. Russell, and K. Kanazawa, "Adaptive Probabilistic Networks with Hidden Variables," *Machine Learning*, vol. 29, no. 2-3, pp. 213–244, 1997.
[26] K. Murphy, "The Bayes net toolbox for Matlab," *Computing Science and Statistics*, vol. 33, pp. 331–350, 2001.
[27] C. F. Aliferis, A. R. Statnikov, I. Tsamardinos, and L. E. Brown, "Causal explorer: a causal probabilistic network learning toolkit for biomedical discovery," in *Proceedings of the International Conference on*

Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'03), pp. 371–376, June 2003.

[28] R. Yehezkel and B. Lerner, "Bayesian network structure learning by recursive autonomy identification," *Journal of Machine Learning Research*, vol. 10, pp. 1527–1570, 2009.

[29] S. L. Lauritzen, *Graphical Models*, vol. 17, Clarendon Press, New York, NY, USA, 1996.