Xu JG, Zhao Y, Chen J *et al.* A structure learning algorithm for Bayesian network using prior knowledge. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(4): 713–724 July 2015. DOI 10.1007/s11390-015-1556-8

A Structure Learning Algorithm for Bayesian Network Using Prior Knowledge

Jun-Gang Xu[1] (徐俊刚), *Member, CCF, ACM, IEEE*, Yue Zhao[1] (赵 越)
Jian Chen[2] (陈 健), *Member, CCF, ACM, IEEE*, and Chao Han[2] (韩 超)

[1]*School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 101408, China*
[2]*School of Software Engineering, South China University of Technology, Guangzhou 510006, China*

E-mail: xujg@ucas.ac.cn; zhaoyue11@mails.ucas.ac.cn; ellachen@scut.edu.cn; hanchaos@163.com

Received January 31, 2015; revised March 25, 2015.

Abstract Learning structure from data is one of the most important fundamental tasks of Bayesian network research. Particularly, learning optional structure of Bayesian network is a non-deterministic polynomial-time (NP) hard problem. To solve this problem, many heuristic algorithms have been proposed, and some of them learn Bayesian network structure with the help of different types of prior knowledge. However, the existing algorithms have some restrictions on the prior knowledge, such as quality restriction and use restriction. This makes it difficult to use the prior knowledge well in these algorithms. In this paper, we introduce the prior knowledge into the Markov chain Monte Carlo (MCMC) algorithm and propose an algorithm called Constrained MCMC (C-MCMC) algorithm to learn the structure of the Bayesian network. Three types of prior knowledge are defined: existence of parent node, absence of parent node, and distribution knowledge including the conditional probability distribution (CPD) of edges and the probability distribution (PD) of nodes. All of these types of prior knowledge are easily used in this algorithm. We conduct extensive experiments to demonstrate the feasibility and effectiveness of the proposed method C-MCMC.

Keywords Bayesian network, structure learning, Markov chain Monte Carlo, prior knowledge

1 Introduction

Bayesian network is an important probabilistic graphical model that represents a set of random nodes and their conditional dependencies. It is also one of the most effective theoretical models for decision making, especially for uncertain knowledge reasoning[1]. In recent years, Bayesian networks have been increasingly used in a wide range of applications, such as natural language processing, hardware diagnostics, bioinformatics, statistical physics, and econometrics.

Structure learning is a fundamental task for building Bayesian network. It tries to find a network topology that can represent the dataset. This problem is proved NP-hard[2] and the conventional optimal search algorithms take exponential time and space on the number of nodes. There are also numerous studies on reducing the exponential search space, but the presence of the inherent uncertainty in any heuristic algorithm influences the validity of any recovered structure, particularly when the size of the search space is high[3-5].

In order to reduce the uncertainty of structures retrieved by automatic learning algorithms, several studies incorporating prior knowledge into Bayesian network structure learning have been conducted in the last ten years. However, most of the proposed algorithms are complex and difficult to apply in practice. Some algorithms require the prior knowledge in high quality, and they need the users to specify a structure or an ordering of nodes, which both are not easy to achieve. In addition, prior knowledge that is not completely convinced by domain experts is used to learn the structure of Bayesian network. The previous work did not present the confidence of prior knowledge when they

Regular Paper
Special Section on Data Management and Data Mining
This work was supported by the National Natural Science Foundation of China under Grant No. 61372171 and the National Key Technology Research and Development Program of China under Grant No. 2012BAH23B03.

used prior knowledge and this can result in the inaccurate structure of Bayesian network. Besides, some algorithms cannot make full use of the joint probability distribution between nodes. Therefore, the restrictions mentioned above motivate us to propose a better algorithm to learn Bayesian network structure using prior knowledge.

Markov chain Monte Carlo (MCMC) is a good method to learn the structure of Bayesian network[6]. It is a search-and-score algorithm to learn the structure of Bayesian network. In this paper, we intend to focus on adding new expert knowledge (not presented in the dataset) to the MCMC framework. The original MCMC algorithm is an iterative random algorithm and a random structure should be given at the beginning of the algorithm. Each iteration has two steps. The first one is the proposing step: the algorithm gives a proposal of adding, deleting or reversing a random edge of the graph. The second one is the moving step: the proposal of the first step will be accepted with a probability which is calculated by a scoring function. If the proposal is accepted, the algorithm will generate a new graph and start a next iteration; if not, the algorithm will return to the first step and give another proposal. These iterations construct a Markov chain that eventually converges to a posterior distribution of the Bayesian network structure. We can clearly see that the prior knowledge plays a very important role in this classic algorithm.

In our algorithm C-MCMC (Constrained-MCMC), we extend three types of prior knowledge: existence of parent node, absence of parent node, and distribution knowledge including the conditional probability distribution (CPD) of edges and the probability distribution (PD) of nodes. Moreover, we use confidence to denote the confident degree of prior knowledge, and the confidence degree can be set manually. The prior knowledge and its confidence value are incorporated with training data in the moving step of the MCMC algorithm to calculate the accepted probability of the proposal. Compared with the MCMC algorithm, the C-MCMC algorithm can make good use of prior knowledge provided by domain experts and can learn a better Bayesian network structure. Different from current Bayesian network structure learning algorithms using prior knowledge, the C-MCMC algorithm does not have strict restrictions on prior knowledge, and prior knowledge can be used in a simple and effective way. A small amount of knowledge or knowledge that is not so convinced can also help to learn Bayesian network structure. The experimental results show that our algorithm improves the accuracy of the learned structures of Bayesian network much compared with the conventional MCMC algorithm.

The remainder of this paper is organized as follows. Section 2 overviews the related work. Section 3 gives the background knowledge of the MCMC algorithm. Section 4 describes the proposed C-MCMC algorithm. Section 5 presents the experimental evaluation of the algorithm and Section 6 concludes the work.

2 Related Work

It is a challenging task to learn Bayesian network structure from data due to the huge solution space of possible structures. During the past few years, many algorithms have been proposed to use the prior knowledge to reduce the uncertainty of the structures.

Some optimal search methods on Bayesian network structure have been proposed to find the graph with the highest score[7-8]. It is an NP-hard problem and the existing optimal algorithms spend time and memory proportional to $n \times 2^n$ where n represents the number of nodes. The algorithms based on dynamic programming[8] and parallelization[7] can reduce time complexity efficiently. However, Bayesian network has the size limit of 30 nodes due to its exponential space search complexity[7]. So far, various algorithms have been explored through using prior knowledge to reduce search space to find the optimal structure. Perrier *et al.* developed a super-structure constrained optimal search algorithm to find the optimal structure which needs to be a sub-graph of the given undirected super-structure[9]. For larger Bayesian network, Kojima *et al.* divided a super structure into clusters and performed optimal search algorithm to find the optimal structure[10]. However, a cycle straddling multiple clusters may be generated in this algorithm. de Campos and Ji used two kinds of constraints, indegree of one node and arc between nodes. This method can strongly reduce the time and memory costs of the algorithm[11].

Markov chain Monte Carlo (MCMC) is a class of algorithms for sampling from a probability distribution and it can be used to learn the structure of Bayesian network. Ehlers provided a computational tool for estimating and comparing GARCH models using MCMC and an approximation to the Bayesian factor[12]. Grzegorczyk and Husmeier proposed a novel MCMC algorithm with new and more extensive edge reversal move in the moving step and it significantly improves the convergence[13]. To speed up the learning process, Corander *et al.* proposed a method to learn structures of graphical model from training samples using the non-reversible parallel interacting MCMC-style computation[14]. Several heuristic algorithms also use prior knowledge to learn Bayesian network structure and obtain better results[15-17]. The algorithm proposed by Teyssier and Koller is based on the fact that the best structure is consistent with a node ordering, searched not over the space of structures but over the space orderings to reduce the search space[15]. Cano *et al.* proposed an algorithm in which the domain experts are requested to submit their knowledge during the process and the system only asks the domain experts about the most uncertain structural feature and the presence or the absence of an edge[16]. de Campos and Castellano used a heuristic algorithm with statistic data and three types of prior knowledge: existence of arcs and/or edges, absence of arcs and/or edges, and ordering restrictions to learn the structure of Bayesian network[17].

However, we find that the Bayesian network structure learning algorithms described above have the following disadvantages. 1) Some prior knowledge is not easy to be used in the current algorithms. For example, in some existing algorithms, a completely correct node ordering is needed, but it is hard for us to make sure that every single detail of the given ordering is right. And in some other existing algorithms, we have to give a super structure first, but most of the time, we only have some local knowledge of the structure and it is hard to achieve an overall structure. 2) The prior knowledge described above is considered as "hard" restrictions; therefore every candidate Bayesian network must satisfy them. However, we are not always fully confident about the prior knowledge. Therefore, the prior knowledge that we are not confident cannot be used in these algorithms. 3) Sometimes, we can achieve the joint probability distribution between nodes, but as for most of the existing algorithms, we cannot make full use of this kind of prior knowledge.

The C-MCMC algorithm we propose also uses domain knowledge to help to learn Bayesian network structure. Compared with the existing algorithms, the C-MCMC algorithm has the following features. 1) It has very few restrictions on the prior knowledge. Almost all prior knowledge including local structure, the CPD of some edges, the PD of nodes and so on, can help to improve the performance of the algorithm. 2) Each prior knowledge is set with a confidence, and the prior knowledge with a small confidence can also be used. 3) It is easy to define the knowledge and the local structure of Bayesian network. The PD of each node and the CPD of each edge can be easily incorporated into the algorithm. In summary, we can use the prior knowledge and statistical data to learn the Bayesian network structure in a convenient and efficient way.

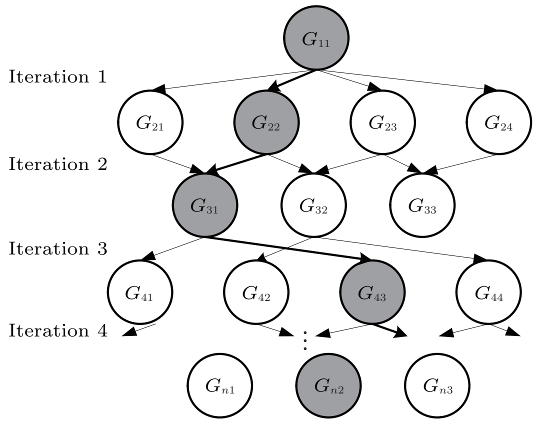
3 MCMC Bayesian Network Structure Learning Algorithm

The MCMC algorithm learns the Bayesian network structure through constructing a Markov chain in the solution space and converging to a posterior distribution of the structure.

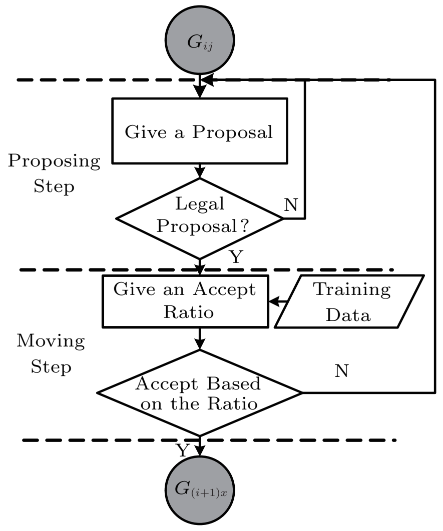
At the beginning of the algorithm, an initial structure and the number of iterations should be given. In each iteration, the algorithm will find some legal neighbors of the current structure and choose one neighbor as the next node of the Markov chain. As shown in Fig.1(a), each node represents one kind of the network structure, and the Markov chain is composed of the grey nodes. A legal neighbor is a structure that can be derived from the current structure with one edge operation and does not introduce directed cycles. The detailed process to generate a node of Markov chain includes the proposing step and the moving step, which is shown in Fig.1(b).

3.1 Notation

First, we provide some notations used in Bayesian network. The structure of a Bayesian network is represented by a directed acyclic graph (DAG) $\boldsymbol{G}$. This structure is defined over a set of n nodes $\{x_1, x_2, \ldots, x_n\}$, and each node takes a value in a finite domain $val(x_i)$ and has a set of parent nodes $pa(x_i)$. For any node x_j, $x_j \in pa(x_i)$ means that there exists an arc from x_j to x_i. Therefore, the graph $\boldsymbol{G}$ can also be decomposed as a vector of parent node sets $\boldsymbol{G} = (pa(x_1), pa(x_2), \ldots, pa(x_n))$. The structure also has m edges $\{e_1, e_2, \ldots, e_m\}$. For an edge e, it connects two nodes, a parent node and a child node. We use $paNode(e)$ to denote the parent node and use $chNode(e)$ to denote the child node. In addition, we have a fully observed training dataset $\boldsymbol{D}$ with N instances, use ζ to denote all possible networks and use $\#(nbd(\boldsymbol{G}))$ to denote the legal neighbor number of $\boldsymbol{G}$.



(a)



(b)

Fig.1. Framework of the MCMC algorithm. (a) Structure of a Markov chain. (b) Process of one iteration.

3.2 Proposing Step

The proposing step randomly proposes a legal neighbor structure $\boldsymbol{G'}$ of the current structure $\boldsymbol{G}$. There are three kinds of operations to generate $\boldsymbol{G'}$ from $\boldsymbol{G}$: addition, removal, and reversal. The addition operation is to add an edge from node x_i to node x_j, and it should not introduce any path from x_j to x_i to guarantee the legality of the network. The removal operation is to remove an edge from $\boldsymbol{G}$, and it is always legal. The reversal operation is to change the direction of an edge, and it can be decomposed into two operations of removal and addition. After these operations, we will get a legal neighbor structure $\boldsymbol{G'}$.

3.3 BDeu Scoring Function

Before the moving step, we briefly describe the BDeu function[18], the scoring function of the MCMC algorithm, and it is also used as the scoring function of our proposed algorithm. For DAG models, the BDeu function has a typical assumption that a Dirichlet distribution is assigned for each node, conditionally on the configurations of its parents. Hence, as shown in [18], the marginal likelihood of a discrete DAG model is defined as (1).

$$L(\boldsymbol{G}) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})}. \tag{1}$$

In (1), i denotes the i-th node of the network, while n denotes the number of nodes of the network, q_i is the number of configurations of the parents of the i-th node, j is the j-th configuration of the parent, r_i is the number of states of x_i, k is the k-th state of the node, N_{ijk} denotes the observed count in the training set, and N'_{ijk} is the hyper parameter. The value of N'_{ijk} is calculated by (2).

$$N'_{ijk} = N'/(r_i \times q_i). \tag{2}$$

In (2), N' denotes equivalent instance size, and it is given by domain experts. The value of N' is commonly set to 1[6].

After getting the prior distribution of the parameters, the Bayesian model can be calculated based on the Bayes theorem. For a given graph $\boldsymbol{G}$ and training dataset $\boldsymbol{D}$, $p(\boldsymbol{D}) = \sum_{\boldsymbol{G}\in\varsigma} p(\boldsymbol{D}|\boldsymbol{G})p(\boldsymbol{G})$ denotes the marginal likelihood of the training data, and $p(\boldsymbol{G}|\boldsymbol{D})$ can be described as (3).

$$p(\boldsymbol{G}|\boldsymbol{D}) = \frac{p(\boldsymbol{D}|\boldsymbol{G})p(\boldsymbol{G})}{p(\boldsymbol{D})}. \tag{3}$$

3.4 Moving Step

In the moving step, the proposed neighbor structure $\boldsymbol{G'}$ generated by the proposing step is accepted with a probability α, which is between R_a and 1, and it is described as (4).

$$\alpha = \min\{R_a, 1\}, \tag{4}$$

where

$$R_a = \frac{\#(nbd(\boldsymbol{G}))p(\boldsymbol{G'}|\boldsymbol{D})}{\#(nbd(\boldsymbol{G'}))p(\boldsymbol{G}|\boldsymbol{D})}. \tag{5}$$

In (5), we consider that the occurrences of $\boldsymbol{G}$ and $\boldsymbol{G'}$ have an equal prior probability and the value of $p(\boldsymbol{G}|\boldsymbol{D})$

can be calculated with (3). Thus R_a can also be defined as (6).

$$R_a = \frac{\#(nbd(\boldsymbol{G}))p(\boldsymbol{D}|\boldsymbol{G}')}{\#(nbd(\boldsymbol{G}'))p(\boldsymbol{D}|\boldsymbol{G})}. \tag{6}$$

Accepted ratio R_a can be calculated through the Bayesian factor $B = p(\boldsymbol{D}|\boldsymbol{G}')/p(\boldsymbol{D}|\boldsymbol{G})$. Thus R_a can be written as (7).

$$R_a = \frac{\#(nbd(\boldsymbol{G}))}{\#(nbd(\boldsymbol{G}'))} \times B. \tag{7}$$

For the i-th node, the Bayesian factor can be calculated using two terms of likelihood $L(\boldsymbol{G})$, which is defined as (8).

$$B = \frac{L(\boldsymbol{G}', x_i)}{L(\boldsymbol{G}, x_i)}. \tag{8}$$

In (8), the likelihood $L(\boldsymbol{G}, x_i)$ refers to the likelihood $L(\boldsymbol{G})$ for the i-th node, which is defined as (9) and N'_{ijk} can be calculated with (2).

$$L(\boldsymbol{G}, x_i) = \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})}. \tag{9}$$

When the proposal operation is reversal, the Bayesian factor B should be calculated with (10). In this equation, we use x_{i1} and x_{i2} to denote the two nodes of the reversal edge e.

$$B = \frac{L(\boldsymbol{G}', x_{i1})L(\boldsymbol{G}', x_{i2})}{L(\boldsymbol{G}, x_{i1})L(\boldsymbol{G}, x_{i2})}. \tag{10}$$

4 Constrained MCMC Bayesian Network Structure Learning Algorithm Using Prior Knowledge

In this section, we present and describe the Constrained-MCMC (C-MCMC) algorithm, which incorporates the prior knowledge into the MCMC algorithm to learn the structure of Bayesian network. The difference between the MCMC and the C-MCMC algorithm is the moving step. In the MCMC algorithm, the proposal is accepted with a probability which is only related to the training dataset, while the C-MCMC algorithm uses the prior knowledge and training dataset to calculate this probability.

4.1 Prior Knowledge

In C-MCMC algorithm, three kinds of prior knowledge are used: existence, absence, and PD/CPD knowledge. All prior knowledge should be given by domain experts. Existence knowledge means that for any node x, we have a node set $pa_e(x)$ that includes all parent nodes of the node x. Absence knowledge means that for a node x, we have a node set $pa_a(x)$ that does not include any parent node of the node x. PD/CPD knowledge means that the PD of a node and the CPD of an edge are known.

Considering that the prior knowledge may not be consistent and reliable, confidence λ is assigned by domain experts on each of the prior knowledge. For the existence and absence knowledge, we assign each node in $pa_e(x)$ or $pa_a(x)$ a confidence λ. For the PD/CPD knowledge, we assign a confidence λ to each node or edge which has the PD/CPD knowledge.

λ ranges from 0 to 1. If we are very confident about this knowledge, λ can be set to 1. It means that the knowledge is assumed to be true and therefore all the candidate structures must satisfy it. If we are not very sure about this knowledge, λ can be given a value between 0 and 1, which denotes the certainty level of this knowledge. The smaller the confidence, the more serious the impact of the training dataset. Adjusting the value of λ can effectively control the balance of prior knowledge and training dataset. If the value of λ is 0, it means that the prior knowledge does not have any influence on the moving step and the accepted probability of the proposal is totally determined by the training dataset.

4.2 Use of Prior Knowledge

In the proposing step, the algorithm gives a proposal on a random edge e which has a parent node $paNode(e)$ and a child node $chNode(e)$. In the moving step, we use prior knowledge and training data to calculate the probability α defined in (4) to accept this proposal. The detailed usages of three kinds of prior knowledge are described as follows.

4.2.1 Use of Existence Knowledge

We use existence knowledge if we have the knowledge of $pa_e(chNode(e))$. If $paNode(e) \in pa_e(chNode(e))$ and the corresponding confident is λ, we will believe this knowledge is true with a probability λ and set the accepted probability α to 1 with the probability λ. This kind of prior knowledge is believed to be false with a probability $(1-\lambda)$ and we will use the method described in Subsection 4.3 to calculate the probability R_a, and then get the value of α with the training dataset. If $paNode(e) \notin pa_e(chNode(e))$, it means that we do not have any existence knowledge on this proposal, and the algorithm will transfer to Subsection 4.3 to calculate the probability R_a.

4.2.2 *Use of Absence Knowledge*

We use absence knowledge if we have the knowledge of $pa_a(chNode(e))$. If $paNode(e) \in pa_a(chNode(e))$ and the corresponding confident is λ, we will believe this knowledge is true with a probability λ and set the accepted probability α to 0 with the probability λ. This kind of prior knowledge is believed to be false with a probability $(1-\lambda)$ and we will use the method described in Subsection 4.3 to calculate the probability R_a to get the value of α with the training dataset. If $paNode(e) \notin pa_a(chNode(e))$, it means that we do not have any absence knowledge on this proposal and the algorithm will transfer to Subsection 4.3 to calculate the probability R_a.

4.2.3 *Use of PD/CPD Knowledge*

If we have neither PD knowledge of the i-th node nor CPD knowledge of the edge e when $chNode(e) = x_i$ and $paNode(e)$ belongs to the j-th parent configuration of the node x_i, the probability R_a can be calculated with (7)∼(9), which is the same as the MCMC algorithm. If we have the PD/CPD knowledge, the probability R_a can be calculated with (7), (8) and (11).

$$L(\boldsymbol{G}, x_i) = \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(M_{ij}+N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(M_{ijk}+N'_{ijk})}{\Gamma(N'_{ijk})}. \qquad (11)$$

In (9), N_{ijk} denotes the observed count in the training set. In (11), we use M_{ijk} to replace N_{ijk}. When we calculate the value of N_{ijk}, we need to count the observed times in the training set for each value's combinations of the i-th node and its j-th configuration of the parent nodes. Thus, we can get the CPD information (denoted as P_n) of these two nodes. And the CPD knowledge of these two nodes is denoted as P_k. Then we use P_n and P_k to calculate a new CPD P_m with (12), and use this new CPD P_m to generate the value of M_{ijk}. The distribution of M_{ijk} should follow the distribution P_m and the total number of all observed counts should not be changed.

$$P_m = \lambda \times P_k + (1-\lambda) \times P_n. \qquad (12)$$

Similar to the circumstance of two nodes, if we have PD knowledge P_k of the i-th node and one of its parent nodes in the node set which is corresponding to N_{ijk}, the marginal probability distribution of these two nodes can also be calculated with (12). The use of PD/CPD knowledge is equivalent to changing the probability distribution of nodes to calculate a new value of $L(\boldsymbol{G}, x_i)$. The confidence λ is used to adjust the influence of the prior knowledge and the training dataset.

4.3 Influences of Prior Knowledge

For existence knowledge and absence knowledge, if the proposal is consistent with them, it would be more likely to be accepted compared with only using the dataset to decide whether to accept this proposal. With the first two kinds of knowledge, the probability of accepting this proposal can be denoted as (13).

$$\alpha = \min\{\lambda + (1-\lambda) \times R_a, 1\}. \qquad (13)$$

As defined in (7), the value of R_a is calculated with the training data and it is always greater than 0. We can prove that $\lambda + (1-\lambda) \times R_a$ is always greater than R_a that is calculated only based on the training dataset. We can prove this through (14).

$$\begin{aligned}&(\lambda + (1-\lambda) \times R_a)/R_a \\ &= \lambda/R_a + (1-\lambda) = 1 + \lambda(1/R_a - 1) > 1. \qquad (14)\end{aligned}$$

Therefore, the algorithm could accept a proposal which is consistent with the prior knowledge with a greater probability.

For PD/CPD knowledge, the BDeu scoring function has a simple interpretation in terms of equivalent sample size. We use a new PD/CPD to get the value of M_{ijk} which is corresponding to a new dataset with the help of prior knowledge, and our goal is to find a network topology that best fits this new dataset. Therefore, if the proposal fits this dataset better, it would have a greater probability to be accepted. And the statistic information of this dataset is influenced by PD/CPD knowledge with confidence λ. The greater the confidence is, the more the learned network fits the prior knowledge.

In this way, prior knowledge can be introduced into the algorithm. And then, the probability is calculated by (4) and the proposal can be accepted with this probability. If the proposed $\boldsymbol{G}'$ is accepted, the algorithm will set $\boldsymbol{G}'$ as the current structure and start next iteration. If not, the algorithm will go back to the proposing step and give another proposal.

Using prior knowledge is equivalent to adding an extra dataset to the original training dataset and generating a new training dataset, and then the structure is learned with this new training dataset. Thus the convergence of the algorithm is not influenced by introducing prior knowledge.

5 Experiments

5.1 Experimental Methodology

Three metrics are used to evaluate the performance of the C-MCMC algorithm. The first metric is ADI, which uses three sub-metrics to measure the steps to reconstruct the original network from the learned network: the number of added arcs (A), the number of deleted arcs (D), and the number of inverted arcs (I). The second metric is Kullback-Leibler (KL) distance, which is commonly used to evaluate the learned structures of Bayesian network. The smaller the KL-divergence is, the better the learned network fits the training dataset. The third metric is BDeu scoring function, which is also used to guide the local search method for the MCMC algorithm. The experimental result of the first metric is calculated based on the learned network and the original network. The experimental results of the last two metrics are calculated based on the learned networks and the testing dataset.

Synthetic datasets are generated from four commonly used Bayesian networks, the Asia network with eight nodes[19], the Insurance network with 27 nodes[20], the Alarm network with 37 nodes[21], and the Bat network with 52 nodes[22]. For each Bayesian network, we generate 10 training datasets, each of which contains 1 000 instances. At the same time, we also generate a testing dataset for each Bayesian network, and each testing dataset contains 1 000 instances. For each network, the experiment on each training dataset with one parameter configuration is repeated 10 times, and 100 learned structures and their metric values can be achieved.

For each network, we randomly select a fixed percentage of prior knowledge extracted from the whole set of prior knowledge of the corresponding original network. More precisely, for a node x_i in the original network that has node set $\{x_1, x_2, \ldots, x_n\}$, consider that the node set $\{x_1, x_2, \ldots, x_{i-1}\}$ is the parent set of the node x_i and the nodes in the set $\{x_{i+1}, x_{i+2}, \ldots, x_n\}$ are not in the parent set of x_i. Thus, the knowledge that x_1 is a parent node of the node x_i is a prior knowledge in the existence prior knowledge set. The knowledge that x_{i+1} is not a parent node of the node x_i is a prior knowledge in the absence prior knowledge set. In this way, the knowledge retrieved from the original network is added to the existence and the absence prior knowledge sets respectively. The CPD/PD knowledge is also retrieved from the original network. If the original network has m edges, we will have m pieces of CPD knowledge and n pieces of PD knowledge. The selected percentage of prior knowledge is set to 20%, 40%, 60% and 80% respectively. We run the algorithm for each percentage of prior knowledge respectively and get the results of the experiments.

In the experiments, we compare the performance of the C-MCMC algorithm with that of the MCMC algorithm. As both of them are random algorithms, in each experiment, we set the iterations to 10 000 to ensure the convergence, set burn-in to 200, start from the independence model, and use the same dataset generated above. The equivalent instance size N' is set to 1.

5.2 Evaluation with ADI

In this experiment, we set the percentage of prior knowledge to 20%, 40%, 60% and 80% respectively and use ADI to evaluate the performance of the algorithms. The ADI can be directly used to compare the learned structure with the original structure, discover the difference, and count the number of operations required to achieve the learned network from the original network. In this experiment, the confidences of all prior knowledge are all set to 0.85. Fig.2 shows the results of this experiment.

From Fig.2, it is easy to find that we have a higher probability to get a better structure by increasing the percentage of prior knowledge from 20% to 80%. The MCMC algorithm does not use any prior knowledge, and thus its A, D and I are worse than those of the C-MCMC algorithm. We also find that the effect of prior knowledge is more significant in the network with less nodes, such as the Asia network and the Insurance network. When we learn the structure of a Bayesian network with a large number of nodes, plenty of prior knowledge used in the learning process brings down the inherent uncertainty. It is more difficult to learn the structure of a large Bayesian network with 1 000 training instances.

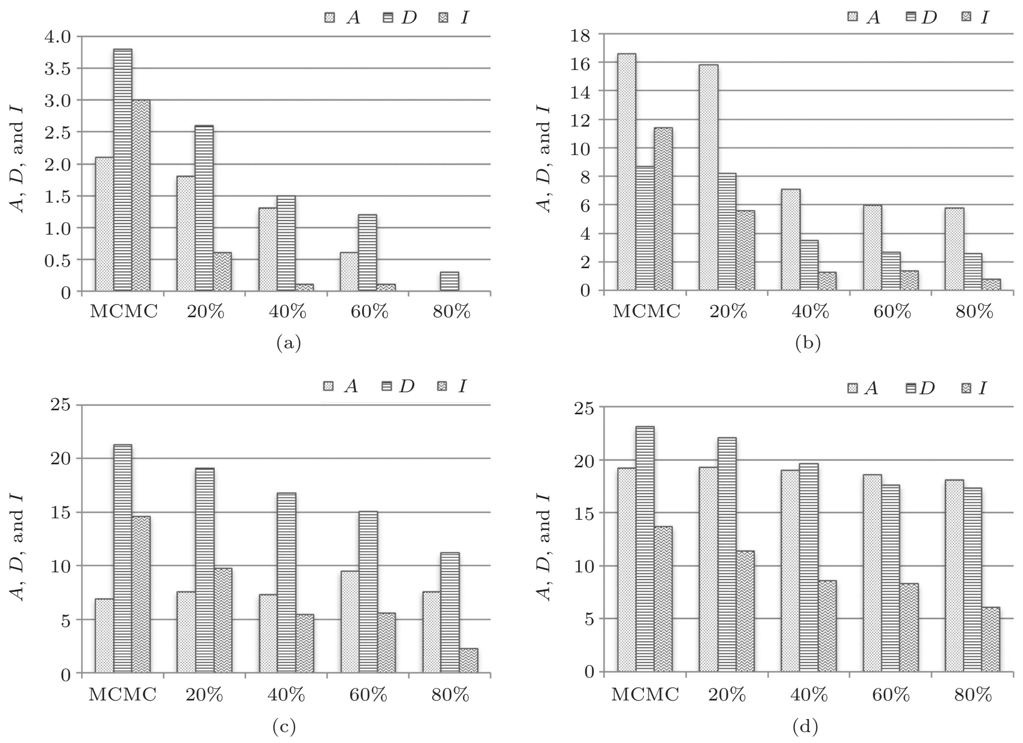
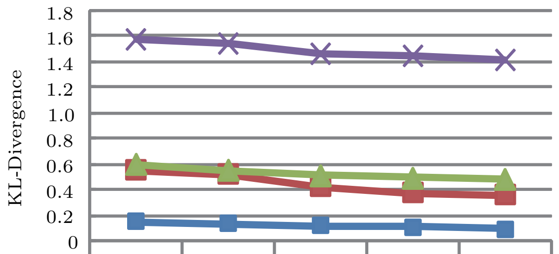


Fig.2. ADI values of the learned structures using MCMC without prior knowledge and C-MCMC with 20%, 40%, 60%, and 80% of prior knowledge respectively. (a) ADI values of Asia. (b) ADI values of Insurance. (c) ADI values of Alarm. (d) ADI values of Bat.

5.3 Evaluation with KL-Divergence

In this experiment, we calculate KL-divergence between the learned network and the testing dataset to evaluate the performance of the algorithm. The KL-divergence is a non-symmetric measure of the difference between two probability distributions P and Q. P typically represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution. Q typically represents a theory, model, description, or approximation of P. KL-divergence is commonly used to evaluate the learned structures of Bayesian network, and we calculate the KL-divergence following the method proposed by Acid and de Campos[23]. The smaller the KL-divergence is, the better the learned network fits the training dataset. The confidences of all prior knowledge are also set to 0.85. Fig.3 shows the results of this experiment.

From Fig.2, we can see that the use of prior knowledge can help to get a better structure of Bayesian network, and with the increase of percentage of prior knowledge from 20% to 40%, the improvement will be more remarkable. Compared with the first experiment, we believe that when the percentage of prior knowledge increases from 20% to 40%, the inherent uncertainty reduces faster than that in 0∼20% and 40%∼100%.



	MCMC	20%	40%	60%	80%
Asia	0.149 6	0.134 4	0.115 7	0.104 2	0.085 4
Alarm	0.547 2	0.517 5	0.426 5	0.368 5	0.354 7
Insurance	0.596 5	0.551 9	0.509 8	0.491 2	0.482 7
Bat	1.578 3	1.546 2	1.467 8	1.444 3	1.413 5

Fig.3. Values of KL-divergence between learned structures and the original structure using the MCMC algorithm without prior knowledge and the C-MCMC algorithm with 20%, 40%, 60%, and 80% of prior knowledge.

5.4 Evaluation with BDeu Function

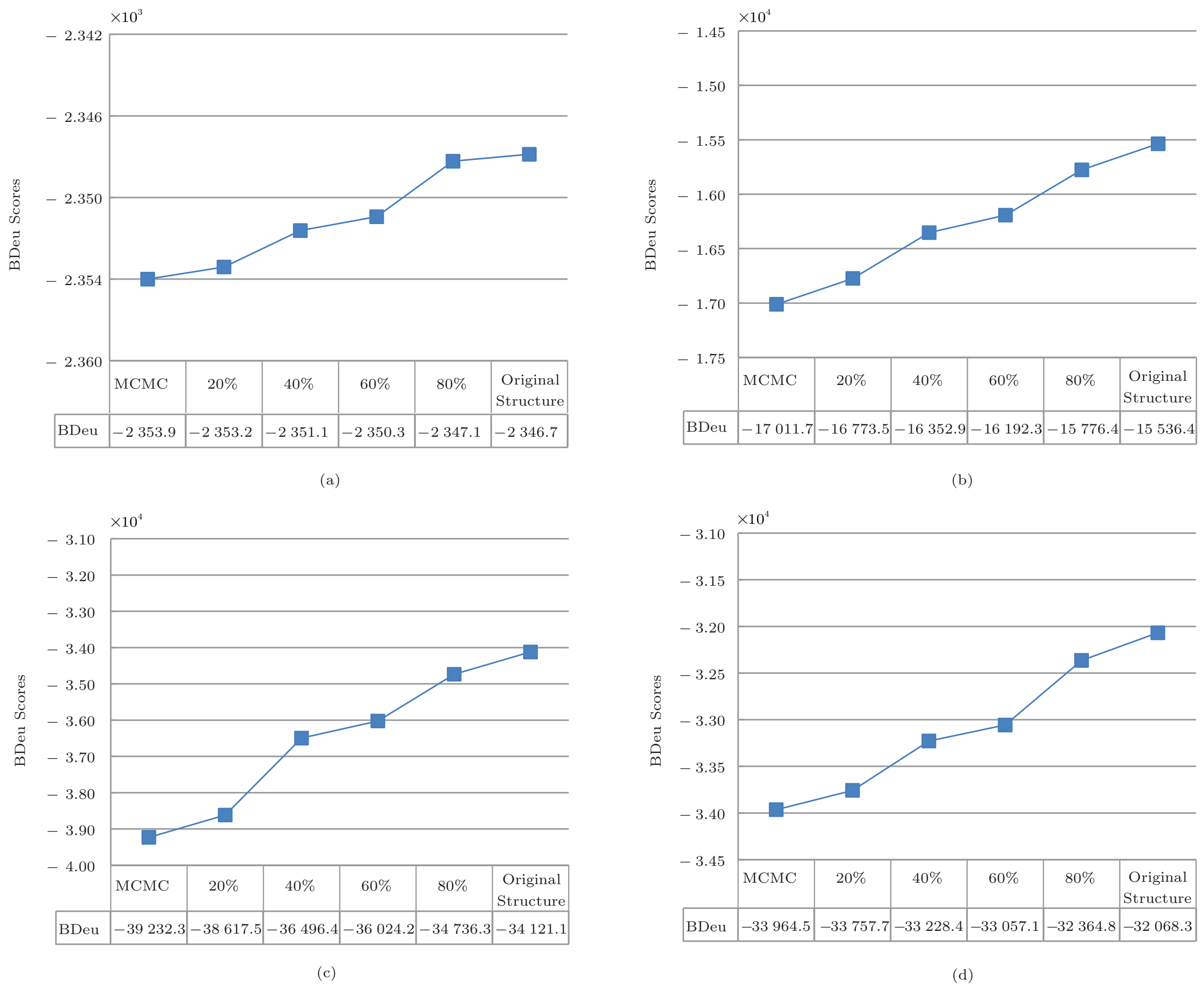
In this experiment, we calculate BDeu with the learned network and the testing dataset. BDeu is a likelihood-equivalent Bayesian scoring metric and it is described in detail in Subsection 3.3. The greater the BDeu is, the better the structure learned by the algorithm is. The confidences of all prior knowledge are also set to 0.85. The local search process of the MCMC algorithm is based on the BDeu function while the local search process of C-MCMC algorithm is based on prior knowledge and the BDeu function.

As shown in Fig.4, we can easily find that the results of the C-MCMC algorithm are better than those of the MCMC algorithm. Based on prior knowledge, the C-MCMC algorithm has a higher probability to find a better structure. Furthermore, the MCMC algorithm may get into overfitting, and thus it cannot perform well with the testing dataset. With the help of prior knowledge, the C-MCMC algorithm can avoid overfitting with a limited size of training dataset. At the same time, we find that the BDeu scores are also consistent with the results of previous experiments.

5.5 Performance Evaluation with Different Confidences of Prior Knowledge

In the previous three experiments, all the confidences of prior knowledge are set to 0.85. In this experiment, we vary the confidence to observe the experimental results. Considering that the results of the previous three experiments are related, we only use the ADI metric to evaluate the effects of the C-MCMC algorithm with different confidences of prior knowledge, including 0.45, 0.65 and 0.85 respectively. The selected percentage of prior knowledge is 80%. The experimental results are shown in Fig.5.

Fig.5 shows that if we set prior knowledge to a high confidence, we will have a higher probability to find a



(a)

	MCMC	20%	40%	60%	80%	Original Structure
BDeu	−2 353.9	−2 353.2	−2 351.1	−2 350.3	−2 347.1	−2 346.7

(b)

	MCMC	20%	40%	60%	80%	Original Structure
BDeu	−17 011.7	−16 773.5	−16 352.9	−16 192.3	−15 776.4	−15 536.4

(c)

	MCMC	20%	40%	60%	80%	Original Structure
BDeu	−39 232.3	−38 617.5	−36 496.4	−36 024.2	−34 736.3	−34 121.1

(d)

	MCMC	20%	40%	60%	80%	Original Structure
BDeu	−33 964.5	−33 757.7	−33 228.4	−33 057.1	−32 364.8	−32 068.3

Fig.4. BDeu scores of the original structure, the learned structures using the MCMC algorithm without prior knowledge, and the C-MCMC algorithm with 20%, 40%, 60% and 80% of prior knowledge. (a) BDeu scores of Asia. (b) BDeu scores of Insurance. (c) BDeu scores of Alarm. (d) BDeu scores of Bat.

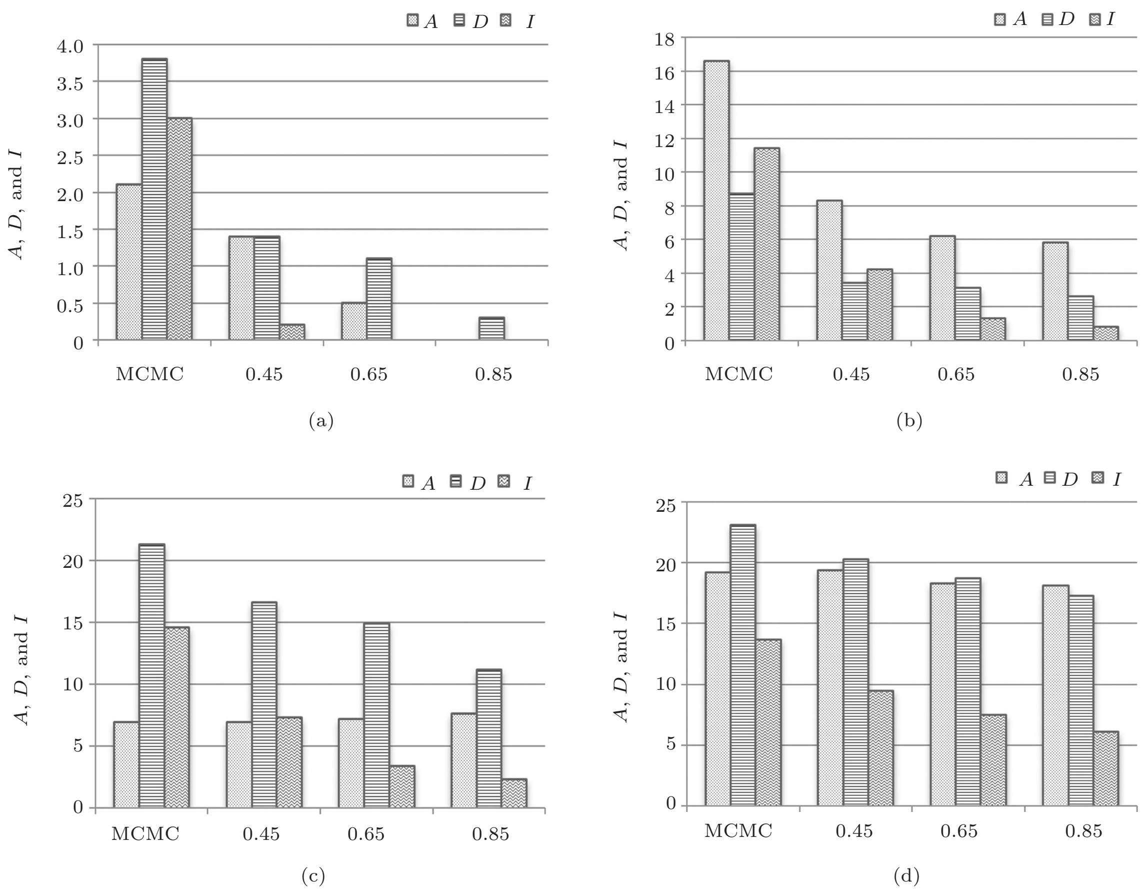


Fig.5. ADI values of the learned structures using the MCMC algorithm and the C-MCMC algorithm with confidences of 0.45, 0.65, and 0.85 respectively. (a) ADI values of Asia. (b) ADI values of Insurance. (c) ADI values of Alarm. (d) ADI values of Bat.

better structure. However, prior knowledge with small confidence can also help the structure learning process. Even though the confidence is very small, the prior knowledge can also be adopted by the algorithm with an inherent probability and have an influence on the moving step to find a better structure. This is because that for an edge operation proposal given by the proposing step, if this proposal is consistent with the prior knowledge, it will have a greater probability to be accepted. And this probability is affected by the confidence. The greater the confidence is, the less influence the training dataset has. Thus we can get a better structure with more confident prior knowledge.

6 Conclusions

Bayesian network structure learning is a challenging task. An algorithm called Constrained-MCMC (C-MCMC) was proposed in this paper. This algorithm incorporates the prior knowledge into the MCMC algorithm to learn the structure of the Bayesian network. This algorithm is one practical algorithm with fewer restrictions on prior knowledge and network scale. The experimental results show that even a small amount of prior knowledge can help improve the learning process of Bayesian network structure learning. We believe that the C-MCMC algorithm can be widely applied in real applications due to its convenience and flexibility.

In the future, we will consider incorporating prior knowledge into other algorithms for Bayesian network structure learning. Furthermore, the training dataset is often incomplete or small; therefore learning Bayesian network structure based on incomplete and small training dataset is another interesting problem to be explored.

Acknowledgement We thank anonymous reviewers for their constructive and valuable comments. We also thank Professor Jian-Feng Zhan at Institute of Computing Technology, Chinese Academy of Sciences, Beijing, for his technical suggestions on this paper.

References

[1] Jensen F V. Introduction to Bayesian Networks. Secaucus, USA: Springer-Verlag, 1996.

[2] Chickering D M, Heckerman D, Meek C. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 2004, 5: 1287-1330.

[3] Heckerman D, Geiger D, Chickering D M. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 1995, 20(3): 197-243.

[4] Chickering D M. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2002, 2: 445-498.

[5] Pernkopf F, Bilmes J. Efficient heuristics for discriminative structure learning of Bayesian network classifiers. *Journal of Machine Learning Research*, 2010, 11: 2323–2360.

[6] Giudici P, Castelo R. Improving Markov chain Monte Carlo model search for data mining. *Machine Learning,* 2003, 50 (1/2): 127-158.

[7] Koivisto M, Sood K. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 2004, 5: 549-573.

[8] Silander T, Myllymaki P. A simple approach for finding the globally optimal Bayesian network structure. In *Proc. the 22nd Conference on Uncertainty in Artificial Intelligence*, July 2006.

[9] Perrier E, Imoto S, Miyano S. Finding optimal Bayesian network given a super-structure. *Journal of Machine Learning Research*, 2008, 9: 2251-2286.

[10] Kojima K, Perrier E, Imoto S, Miyano S. Optimal search on clustered structural constraint for learning Bayesian network structure. *Journal of Machine Learning Research*, 2010, 11: 285-310.

[11] de Campos C P, Ji Q. Efficient structure learning of Bayesian networks using constraints. *Journal of Machine Learning Research*, 2011, 12: 663-689.

[12] Ehlers R S. Computational tools for comparing asymmetric GARCH models via Bayes factors. *Mathematics and Computers in Simulation*, 2012, 82(5): 858-867.

[13] Grzegorczyk M, Husmeier D. Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. *Machine Learning*, 2008, 71(2/3): 265-305.

[14] Corander J, Ekdahl M, Koski T. Parallel interacting MCMC for learning of topologies of graphical models. *Data Mining and Knowledge Discovery*, 2008, 17(3): 431-456.

[15] Teyssier M, Koller D. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proc. the 21st Conference on Uncertainty in Artificial Intelligence*, June 2005, pp.548-549.

[16] Cano A, Masegosa A R, Moral S. A method for integrating expert knowledge when learning Bayesian networks from data. *IEEE Transaction on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2011, 41(5): 1382-1394.

[17] de Campos L M, Castellano J G. Bayesian network learning algorithms using structural restrictions. *International Journal of Approximate Reasoning,* 2007, 45(2): 233-254.

[18] Heckerman D, Geiger D, Chickering D M. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning,* 1995, 20(3): 197-243.

[19] Lauritzen S L, Spiegelhalter D J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B (Methodological)*, 1988, 50(2): 157-224.

[20] Binder J, Koller D, Russell S, Kanazawa K. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 1997, 29(2/3): 213-244.

[21] Beinlich I A, Suermondt H J, Chavez R M, Cooper G F. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proc. the 2nd European Conference on Artificial Intelligence in Medicine*, August 1989, pp.247-256.

[22] Forbes J, Huang T, Kanazawa K, Russell S. The BATmobile: Towards a Bayesian automated taxi. In *Proc. the 14th International Joint Conference on Artificial Intelligence*, August 1995, pp.1878-1885.

[23] Acid S, de Campos L M. Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 2003, 18: 445-490.

Jun-Gang Xu is an associate professor of the School of Computer and Control Engineering (SCCE) at University of Chinese Academy of Sciences (UCAS), Beijing, where he joined as a faculty member in 2005. He received his B.S. and M.S. degrees in mechanical engineering from Southwest Petroleum University, Chengdu, and Shandong University, Jinan, in 1996 and 1999 respectively, and his Ph.D. degree in computer science from Graduate University of Chinese Academy of Sciences, Beijing, in 2003. During 2003∼2005, he was a postdoctoral researcher in Tsinghua University, Beijing. His current research interests include data mining, information retrieval, big data management and deep learning.

Yue Zhao enrolled in the Graduate Division of the School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing, in 2011. He received his B.S. degree in software engineering from Wuhan University in 2011. His current research interests include Bayesian network and its application, data mining and information retrieval.

Jian Chen is currently a professor of the School of Software Engineering (SSE) at South China University of Technology, Guangzhou, where she joined as a faculty member in 2005. She received her B.S. and Ph.D. degrees from Sun Yat-sen University, Guangzhou, in 2000 and 2005 respectively, both in computer science. Her research interests can be summarized as developing effective and efficient data analysis techniques for complex data and the related applications. Particularly, she is interested in various techniques of data mining, Web search, information retrieval, recommendation techniques as well as their applications.

Chao Han enrolled in the Graduate Division of the School of Software Engineering at South China University of Technology, Guangzhou in 2013. Her research interests mainly include data mining, recommend system and big data processing.