Bayesian Model Averaging of Bayesian Network Classifiers Over Multiple Node-Orders: Application to Sparse Datasets

Kyu-Baek Hwang and Byoung-Tak Zhang

***Abstract*—Bayesian model averaging (BMA) can resolve the overfitting problem by explicitly incorporating the model uncertainty into the analysis procedure. Hence, it can be used to improve the generalization performance of Bayesian network classifiers. Until now, BMA of Bayesian network classifiers has only been performed in some restricted forms, e.g., the model is averaged given a single node-order, because of its heavy computational burden. However, it can be hard to obtain a good node-order when the available training dataset is sparse. To alleviate this problem, we propose BMA of Bayesian network classifiers over several distinct node-orders obtained using the Markov chain Monte Carlo sampling technique. The proposed method was examined using two synthetic problems and four real-life datasets. First, we show that the proposed method is especially effective when the given dataset is very sparse. The classification accuracy of averaging over multiple node-orders was higher in most cases than that achieved using a single node-order in our experiments. We also present experimental results for test datasets with unobserved variables, where the quality of the averaged node-order is more important. Through these experiments, we show that the difference in classification performance between the cases of multiple node-orders and single node-order is related to the level of noise, confirming the relative benefit of averaging over multiple node-orders for incomplete data. We conclude that BMA of Bayesian network classifiers over multiple node-orders has an apparent advantage when the given dataset is sparse and noisy, despite the method's heavy computational cost.**

***Index Terms*—Bayesian model averaging (BMA), Bayesian networks, classification, Markov chain Monte Carlo (MCMC), sparse data.**

I. Introduction

Bayesian networks [1]–[3] are a kind of probabilistic graphical model that compactly represents the joint probability distribution over a set of random variables based on conditional independencies among them. A Bayesian network assumes a directed acyclic graph (DAG) structure where each node corresponds to a variable and an edge denotes a direct probabilistic dependency between the two connected nodes. Formally, the DAG structure asserts that each node is independent of all its nondescendants conditioned on its parent nodes. By these assertions, the Bayesian network consisting of n variables[1] $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ represents the joint probability distribution as

$$P(\mathbf{X}) = \prod_{i=1}^{n} P\left(X_i | \mathbf{Pa}_G(X_i)\right) \tag{1}$$

Manuscript received May 3, 2004; revised December 7, 2004. This work was supported in part by the Korea Ministry of Science and Technology under the NRL and Systems Biology Projects and in part by the Korea Ministry of Education and Human Resources Development under the BK21-IT program. Kyu-Baek Hwang was supported by the Korea Science and Engineering Foundation (KOSEF). The ICT at Seoul National University provided research facilities for this study. This paper was recommended by Editor S. Shimony.

The authors are with the School of Computer Science and Engineering, Seoul National University, Seoul 151-742, Korea (e-mail: kbhwang@bi.snu.ac.kr; btzhang@cse.snu.ac.kr).

Digital Object Identifier 10.1109/TSMCB.2005.850162

[1]In this correspondence, we represent a random variable as a capital letter (e.g., X, Y, and Z) and a set of variables as a boldface capital letter (e.g., $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$). The corresponding lowercase letters denote the instantiation of the variable (e.g., x, y, and z) or all the members of the set of variables (e.g., $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$), respectively.

where $\mathbf{Pa}_G(X_i)$ indicates the set of parent nodes of X_i in the DAG structure G. The term $P(X_i|\mathbf{Pa}_G(X_i))$ is called the local conditional probability distribution of X_i. We denote the set of parameters of all local conditional probability distributions of the Bayesian network as $\mathbf{\Theta}_G$. Thus, a Bayesian network model is represented by the tuple $\langle G, \mathbf{\Theta}_G \rangle$.

Because the conditional probability distribution can be calculated from the joint probability distribution, a Bayesian network consisting of a class variable and feature variables is readily applicable to the classification task. The Bayesian network classifier [4] is distinguished from other classification methods by its ability to portray the probabilistic relationships between class and feature variables via a comprehensible DAG format [see Fig. 1(a)]. In particular, key features for classification can easily be discriminated from less relevant and redundant ones based on the network structure. Such a property is useful in domains where not only classification performance but also an understanding of the process by which the conclusion is inferred is important.

A Bayesian network classifier can be learned from data using score-based search techniques. In these approaches, a scoring metric (e.g., the likelihood score) is employed to evaluate the fitness of a structure. Given the scoring metric, a search method such as greedy search is applied to finding a good model [2], [5]. However, these methods suffer from the lack of distinguishability of scoring metrics when the given dataset is sparse and there can be several distinct Bayesian networks describing the sparse training data equally well. It is hard to evaluate the generalization performance of each Bayesian network classifier to choose one good model. One way to alleviate this data sparseness problem is to restrict the structure of the Bayesian network. The naive Bayes classifier or its tree-augmented variant is a popular choice [4]. Using these models, however, one cannot fully exploit the advantages of Bayesian network classifiers. A promising solution is to employ Bayesian model averaging (BMA) [6]. This provides a principled approach to the model uncertainty problem by integrating all possible hypotheses weighted by their respective posterior probabilities. Thus, the method is helpful in avoiding the overfitting problem and should improve the generalization performance. The idea of combining individual predictions to improve performance is also applied in terms of ensemble machines [7]. Another method for dealing with the model uncertainty problem is the stacking technique [8]. Unlike these methods, BMA offers an intuitive and natural means for integrating hypotheses based on Bayesian probability theory. Utilization of BMA of Bayesian network classifiers was pioneered in [9]–[11]. Dash and Cooper [9] showed that BMA of naive Bayes classifiers can be performed efficiently. In their experiments, BMA generally improved classification performance. Cerquides and López de Màntaras [10] proposed a method for the BMA of tree-augmented naive Bayes classifiers, a restricted form of the general Bayesian network classifier. They also showed that BMA leads to the enhancement of classification accuracy. Recently, Dash and Cooper [11] adopted BMA of general Bayesian network classifiers given a fixed node-order. Its classification performance was better than that of the general single Bayesian network classifier and the naive Bayes model in most experiments.

In this correspondence, we extend BMA of general Bayesian network classifiers by averaging over several distinct node-orders and apply the extended approach to the classification of sparse and noisy data. Our assumption is that a good node-order is hard to obtain and the quality of node-order may affect classification performance in such cases. We expect that averaging over several distinct node-orders may alleviate these problems. For the averaging process, we exploit Friedman and Koller's division of model averaging problem [12].

Their approach provides an efficient and useful approximation of model averaging across all the possible network structures based on the Markov chain Monte Carlo (MCMC) technique [13]. We show the effectiveness of our approach through experiments on two synthetic and four real-life classification problems, simulating diverse situations with respect to sparseness and noise level. This correspondence is organized as follows. In Section II, we describe related work on BMA of Bayesian network classifiers given a single node-order. BMA of Bayesian network classifiers over several node-orders is detailed in Section III. We evaluate our technique's performance through experiments on various sparse and noisy datasets in Section IV. Finally, concluding remarks are drawn in Section V.

II. Related Work

A. Bayesian Classification

We assume that the problem domain is described by n discrete variables $\mathbf{U}$. The variable set $\mathbf{U}$ contains one class variable C and $(n-1)$ feature variables $\{F_1, F_2, \ldots, F_{n-1}\}$ $(= \mathbf{F})$, i.e., $\mathbf{U} = \{C, \mathbf{F}\}$. From the Bayesian perspective, we can classify a new example consisting of feature vector $\mathbf{f} = (f_1, f_2, \ldots, f_{n-1})$ into C^* given the training data D as follows:[2]

$$\text{The class label } C^* \text{ for } \mathbf{f} = \arg\max_C \{P(C|\mathbf{f}, D)\}. \tag{2}$$

The conditional probability distribution in the above equation is calculated by

$$\begin{aligned} P(C|\mathbf{f}, D) &= \frac{P(C, \mathbf{f}|D)}{P(\mathbf{f}|D)} \\ &= \frac{P(C, \mathbf{f}|D)}{\sum_C P(C, \mathbf{f}|D)} \end{aligned} \tag{3}$$

where the summation is taken over all class labels. In the above equation, $P(C, \mathbf{f}|D)$ can easily be calculated from the joint probability distribution over all variables $P(\mathbf{U}|D)$, because $\mathbf{U}$ consists of $\{C, \mathbf{F}\}$. Then, $P(\mathbf{U}|D)$ is described by BMA of Bayesian networks as

$$\begin{aligned} P(\mathbf{U}|D) &= \sum_G P(\mathbf{U}, G|D) \\ &= \sum_G P(G|D) \cdot P(\mathbf{U}|G, D) \\ &= \sum_G P(G|D) \cdot \int P(\mathbf{U}, \boldsymbol{\Theta}_G|G, D) d\boldsymbol{\Theta}_G \\ &= \sum_G P(G|D) \cdot \int P(\boldsymbol{\Theta}_G|G, D) P(\mathbf{U}|\boldsymbol{\Theta}_G, G) d\boldsymbol{\Theta}_G \end{aligned} \tag{4}$$

where the summation is taken over all the possible network structures G and $\boldsymbol{\Theta}_G$ denotes the set of parameters of all local conditional probability distributions. The integral in the above equation can be calculated in a closed form with some reasonable assumptions [2], [14]. However, the summation over all the network structures is nearly impossible even when the number of variables n is about ten, because the number of possible network structures is superexponential in n.

B. Model Averaging Given a Fixed Node-Order

Dash and Cooper [11] approximated (4) with respect to a single ordering on the nodes of a Bayesian network, i.e., they considered only the network structures G consistent with a fixed node-order.[3] We now describe their method as a preliminary to our work. Equation (4) with respect to a given node-order $\prec$ is calculated by

$$\begin{aligned} P(\mathbf{U}|D, \prec) &= \frac{P(\mathbf{U}, D|\prec)}{P(D|\prec)} \\ &= \frac{\sum_G P(\mathbf{U}, G, D|\prec)}{\sum_G P(D, G|\prec)} \\ &= \frac{\sum_G P(G|\prec) \cdot P(D|G) \cdot P(\mathbf{U}|G, D)}{\sum_G P(G|\prec) \cdot P(D|G)} \end{aligned} \tag{5}$$

where we used the equivalence $P(D|G, \prec) = P(D|G)$, because D is independent of the ordering $\prec$ given the structure G. To solve (5), $P(G|\prec)$, the prior probability of the network structure G given the node-order $\prec$ should be determined. We here assume that $P(G|\prec)$ is zero if G is not compatible with $\prec$ and $\beta \cdot P(G)$ otherwise, where β is a normalizing constant. Then, (5) can be rewritten as

$$P(\mathbf{U}|D, \prec) = \frac{\sum_{G \in \mathcal{G}_\prec} P(G) \cdot P(D|G) \cdot P(\mathbf{U}|G, D)}{\sum_{G \in \mathcal{G}_\prec} P(G) \cdot P(D|G)} \tag{6}$$

where $\mathcal{G}_\prec$ denotes the set of network structures that obey the given node-order $\prec$.

Now, (6) can be decomposed into a product of terms for the nodes if we use a decomposable prior on the network structure G, i.e., $P(G) = \prod_i \rho(X_i|\mathbf{Pa}_G(X_i))$, where $\rho(\cdot)$ means a function calculating the contribution of each node to $P(G)$.[4] The posterior probability distribution $P(\mathbf{U}|G, D)$ as well as the marginal likelihood of data $P(D|G)$ can be decomposed as follows, given several assumptions including *parameter independence* and *parameter modularity* [2], [14]:

$$\begin{aligned} P(D|G) &= \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \\ P(\mathbf{U}|G, D) &= \prod_{i=1}^{n} \frac{\alpha_{ij_\mathbf{U}k_\mathbf{U}} + N_{ij_\mathbf{U}k_\mathbf{U}}}{\alpha_{ij_\mathbf{U}} + N_{ij_\mathbf{U}}} \end{aligned} \tag{7}$$

where q_i is the number of configurations of $\mathbf{Pa}_G(X_i)$ and r_i is the number of values of X_i. N_{ijk} is the number of cases in D where X_i assumes its kth value when its parents assume their jth configuration. α_{ijk} denotes a hyperparameter from a Dirichlet distribution for the parameters of each local conditional probability distribution. $N_{ij} = \sum_k N_{ijk}$ and $\alpha_{ij} = \sum_k \alpha_{ijk}$. $\Gamma(\cdot)$ is the Gamma function. The indices $j_\mathbf{U}$ and $k_\mathbf{U}$ denote the j and k values based on the configuration of $\mathbf{U}$, respectively. For simplicity, we denote the product $\rho(X_i|\mathbf{Pa}_G(X_i)) \cdot \prod_{j=1}^{q_i}(\Gamma(\alpha_{ij}))/(\Gamma(\alpha_{ij} + N_{ij})) \cdot \prod_{k=1}^{r_i}(\Gamma(\alpha_{ijk} + N_{ijk}))/(\Gamma(\alpha_{ijk}))$ as $S(X_i; \mathbf{Pa}_G(X_i))|G, D)$, because the term only depends on the local structure around X_i. With these definitions, (6) can be rewritten as

$$P(\mathbf{U}|D, \prec) = \frac{\sum_{G \in \mathcal{G}_\prec} \prod_i S(X_i; \mathbf{Pa}_G(X_i))|G, D) \cdot \frac{\alpha_{ij_\mathbf{U}k_\mathbf{U}} + N_{ij_\mathbf{U}k_\mathbf{U}}}{\alpha_{ij_\mathbf{U}} + N_{ij_\mathbf{U}}}}{\sum_{G \in \mathcal{G}_\prec} \prod_i S(X_i; \mathbf{Pa}_G(X_i))|G, D)}. \tag{8}$$

Finally, the above equation can be efficiently calculated using the following transformation [11]:

$$P(\mathbf{U}|D, \prec) = \frac{\prod_i \sum_{\mathbf{Z} \in \mathcal{Z}_{i,\prec}} S(X_i; \mathbf{Z}|G, D) \cdot \frac{\alpha_{ij_\mathbf{U}k_\mathbf{U}} + N_{ij_\mathbf{U}k_\mathbf{U}}}{\alpha_{ij_\mathbf{U}} + N_{ij_\mathbf{U}}}}{\prod_i \sum_{\mathbf{Z} \in \mathcal{Z}_{i,\prec}} S(X_i; \mathbf{Z}|G, D)} \tag{9}$$

[2] $D = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M\}$, where $\mathbf{u}_m$ $(1 \le m \le M)$ denotes a complete instantiation of the variables in $\mathbf{U}$.

[3] A Bayesian network structure imposes an ordering on its nodes, because it assumes a DAG structure.

[4] Here, X_i denotes class variable C if $i = n$, and feature variable F_i otherwise $(1 \le i \le n-1)$. In this section, we use this notation for convenience. In fact, Bayesian networks do not distinguish class from feature variables.

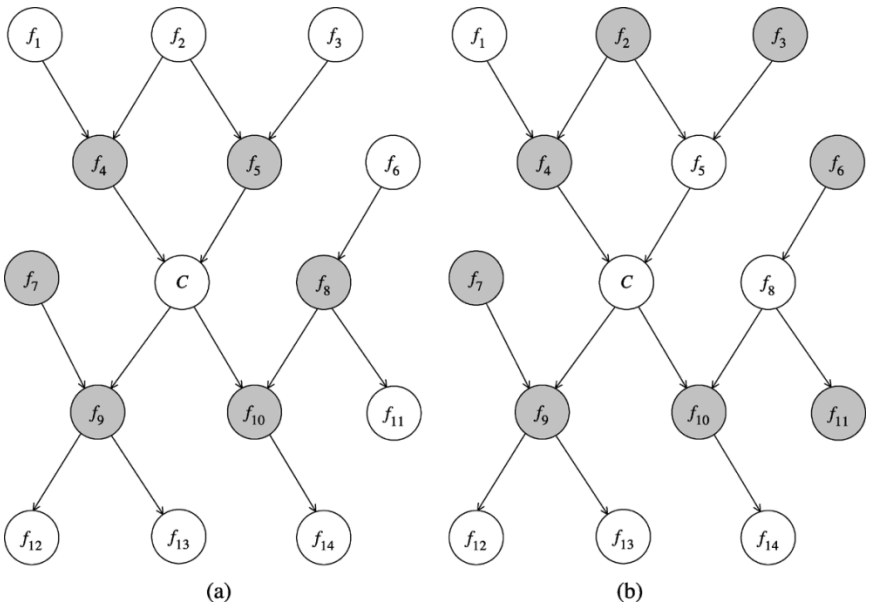

Fig. 1. Extension of Markov blanket. Only the Markov blanket members of the class variable, $\{f_4, f_5, f_7, f_8, f_9, f_{10}\}$, affect the classification if all of the variables are instantiated (a). If the values of some Markov blanket members, say, f_5 and f_8, are not given, then class variable C also becomes dependent on the four additional variables (f_2, f_3, f_6, and f_{11}). Thus, the effective Markov blanket size in this case increases from six to eight (b).

where $\mathcal{Z}_{i,\prec}$ represents the possible parent sets of X_i given node-order $\prec$. The transformation from (8) to (9) is possible because the selection of each node's parents is independent, given a fixed ordering on the nodes. In (9), all the possible parent sets for each node are considered just once, which greatly simplifies the calculation of (8). Thus, the exponential complexity of (8) is reduced to a high-order polynomial (9). Complexity calculation for BMA over a single node-order is detailed in the Appendix.

III. BMA of Bayesian Network Classifiers Over Multiple Node-Orders

A. Motivation

There are various ways for choosing a node-order for BMA of Bayesian network classifiers. The simplest one is to use a random node-order. Dash and Cooper [11] adopted a kind of greedy search for selecting a node-order for model averaging. Larrañaga *et al.* [15] applied a genetic algorithm for finding a good node-order. These methods can perform well if there are enough data examples. However, as we will show in our experiments, a good node-order is hard to obtain when the given training dataset is sparse. The reason for this is that several distinct node-orders can describe the sparse dataset well. In such cases, it is difficult to select one good node-order for BMA.

The quality of node-order can affect the classification performance of BMA as follows. In the classification problem, only the Markov blanket members[5] of the class variable have an effect on classification if there are no missing variables. In Bayesian networks, the Markov blanket of a variable includes its children, parents, and spouses. At least, these variables should be positioned appropriately in the node-order for model averaging. Fig. 1(a) shows an example Bayesian network in which the Markov blanket of the class variable consists of six variables, i.e., $\{f_4, f_5, f_7, f_8, f_9, f_{10}\}$. These six variables must be suitably positioned in the averaged node-order for good classification performance. For example, f_4 and f_5 should precede C in the node-order. In the experiments, we will illustrate the impact of the node-order quality on the classification accuracy (see Fig. 7).

The effect of node-order quality becomes stronger when there are some unobserved variables in test cases. Fig. 1(b) shows the situation when two of the Markov blanket members (f_5 and f_8) are not instantiated. Then, four additional variables (f_2, f_3, f_6, and f_{11}) become dependent on the class variable. These variables are d-separated [1] from the class variable by two of the Markov blanket members. As a result, the number of variables that must be located properly (the effective Markov blanket size) in the averaged node-order grows to eight.

To resolve these problems, we propose to use several distinct node-orders for BMA of Bayesian network classifiers when the given dataset is sparse and noisy. To obtain multiple node-orders, an MCMC sampling technique can be used, which will be described in the next section.

B. Method

To exploit the methodology for model averaging given a single node-order (5), we can rewrite (4) as

$$P(\mathbf{U}|D) = \sum_{\prec} P(\mathbf{U}, \prec |D) = \sum_{\prec} P(\prec |D) \cdot P(\mathbf{U}|D, \prec) \tag{10}$$

where the summation is taken over all possible orderings of the nodes. This decomposition of the averaging task was suggested by Friedman and Koller [12], but they did not apply it to the classification task. Exhaustive summation of (10) is generally impossible because there are $n!$ possible orderings when the number of nodes is n. Instead we can rely on the MCMC methodology [13], which provides a reasonable approximation for the averaging task. It has been successfully applied to diverse applications [16]–[18]. Generally, the MCMC method consists of two parts: 1) Monte Carlo integration and 2) a Markov chain generating a sequence of samples.

Monte Carlo integration approximates the quantity of interest by averaging over samples obtained according to the posterior probability

[5]The Markov blanket of C, $\mathbf{MB}(C)$, is a subset of $\mathbf{F}$ that satisfies $P(C|\mathbf{F}) = P(C|\mathbf{MB}(C))$. In other words, C is conditionally independent from $\mathbf{F} - \mathbf{MB}(C)$ given the value of $\mathbf{MB}(C)$.

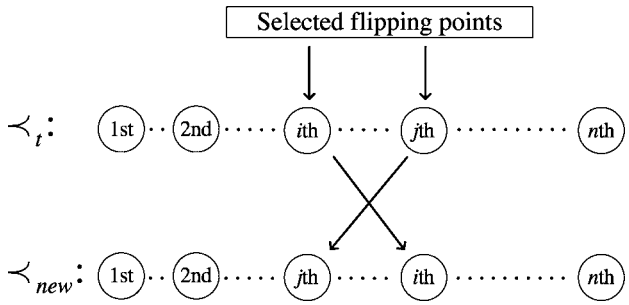

Fig. 2. In the experiments, a move in the Markov chain was defined as flipping two randomly chosen positions in the current node-order.

distribution over the sample space given the dataset. In our problem setting, an appropriate number of node-orders, e.g., T orders $\{\prec_1, \prec_2, \ldots, \prec_T\}$, are generated according to the posterior distribution, $P(\prec|D)$. Then, (10) is approximated as

$$\sum_{\prec} P(\prec|D) \cdot P(\mathbf{U}|D, \prec) \simeq \frac{1}{T}\sum_{t=1}^{T} P(\mathbf{U}|D, \prec_t). \qquad (11)$$

We can use the approximate solution for $P(\mathbf{U}|D)$ obtained by the above equation for Bayesian classification instead of calculating (4). The calculation of (11) is T times more complex than BMA over a single node-order [(9)].

To generate samples based on the posterior probability distribution, a Markov chain that eventually converges to a stationary distribution regardless of its initial state is constructed. Here, the posterior probability distribution over node-orders given the dataset $P(\prec|D)$ corresponds to the stationary distribution. This chain is simulated until a sufficient number of samples is obtained. In practice, the samples obtained initially are thrown out to minimize the effect of the initial seed. This initial period is usually called the *burn-in* phase. The Markov chain can be constructed rather simply by the Metropolis–Hastings algorithm as follows:

- Initialize $\prec_0$; set $t = 0$.
- Repeat the following.
  - — Sample a new order $\prec_{new}$ from $q(\cdot|\prec_t)$.
  - — Sample a uniform random variable R from (0, 1).
  - — If $R \leq \alpha(\prec_t, \prec_{new})$ set $\prec_{t+1}=\prec_{new}$,
  - — Else set $\prec_{t+1}=\prec_t$.
  - — Increment t by 1.

Here, $q(\cdot|\prec_t)$ is the proposal distribution conditioned on the current node-order $\prec_t$. The candidate for the next order $\prec_{new}$ is sampled according to this distribution. $\alpha(\prec_t, \prec_{new})$ denotes the acceptance probability and is defined as

$$\min\left[1, \frac{P(\prec_{new}|D)q(\prec_t|\prec_{new})}{P(\prec_t|D)q(\prec_{new}|\prec_t)}\right]. \qquad (12)$$

A chain constructed in this way guarantees that once a sample is obtained from the posterior distribution $P(\prec|D)$, all subsequent samples are from the same distribution [13].

In the experiments, each move in the Markov chain consisted of flipping the positions of two nodes in the current node-order, as shown in Fig. 2. These two nodes were uniformly selected among all nodes. In other words, a symmetric proposal probability

$$q(\prec'|\prec) = q(\prec|\prec')$$

was used in our experiments, simplifying (12) to $\min[1, (P(\prec_{new}|D))/(P(\prec_t|D))]$. Other parameters for the experiments were set as follows. The prior probability of network structure in (6), $P(G)$, was set as

$$P(G) \propto 2^{-\sum_{i=1}^{n}\left(\log n + \log\left(|\mathbf{Pa}_G^n(X_i)|\right)\right)} \qquad (13)$$

where n is the number of nodes (variables) and $|\cdot|$ denotes the size of a set. This decomposable prior penalizes complex network structures. The Dirichlet prior α_{ijk} in (7) was set to the uninformative value 1.0. When applying the Bayesian network classifier, we also restricted the maximum number of parents of each node to a small constant.[6]

IV. Experimental Evaluation

A. Datasets

1) Synthetic Datasets From the ALARM Network: To examine the proposed method in diverse situations, we used synthetic datasets generated from the ALARM network [19]. This Bayesian network consists of 37 discrete variables, each of which has two to four values. Fig. 3 shows the entire structure of the ALARM network. For the class variable, we chose two binary variables, *Catecholamine* and *Shunt*. These two variables have different Markov blanket structures, as shown in Fig. 4. We experimented with varying training dataset sizes, i.e., 25, 50, 100, 500, and 1000, to investigate how much our method is affected by the sparseness of a given problem. To alleviate the risk of sampling bias, we tried ten independent experiments with each training dataset size. The classification performance of the learned model was evaluated on an independent test dataset consisting of 3000 data examples.

2) Real-Life Benchmark Datasets: We also evaluated the proposed method on three benchmark datasets: *breast*, *mushroom*, and *chess* from the University of California–Irvine machine learning repository (http://www.ics.uci.edu/~mlearn/MLRepository.html). Table I describes the characteristics of each dataset. They are binary classification problems in which the two classes are evenly distributed in each dataset. From each benchmark dataset, we reserved a test dataset; their sizes are specified in the rightmost column of Table I. Then, training datasets, consisting of 25, 50, 75, and 100 examples, were sampled without replacement from the remaining data examples. Ten such datasets were generated for each training sample size and benchmark problem, to reduce the sampling bias.

3) Gene Expression Dataset: Finally, we validated our method on real microarray data, which is known to be very noisy and is usually sparse. Microarray technology [20] is a very useful tool for biological and biomedical research and can measure the expression levels of thousands of genes simultaneously. Hence, microarray data are also called gene expression data. The gene expression dataset used in our experiments is from Cheok *et al.* [21]. The dataset consists of 120 samples. Each sample is a measurement of the expression level of 12 600 genes of a leukemia patient. The classification problem here is to discriminate between samples before drug treatment and those after drug treatment based on the gene expression pattern. Each class is evenly distributed in the dataset, i.e., 60 : 60. For the experiments with the Bayesian network classifier, we processed the original dataset as follows. First, we binarized each gene expression level based on the median expression level of each sample. Then, we selected the 30 genes most closely related to the classification using the mutual information value [22] between gene expression and class label. Hence, the final dataset assumes

[6]The constraint on maximum in-degree of Bayesian network structure is usually imposed, mainly because the number of possible configurations of the parent set increases exponentially with the number of parents, requiring extremely large amounts of data for reliable estimation of local conditional probability distributions.

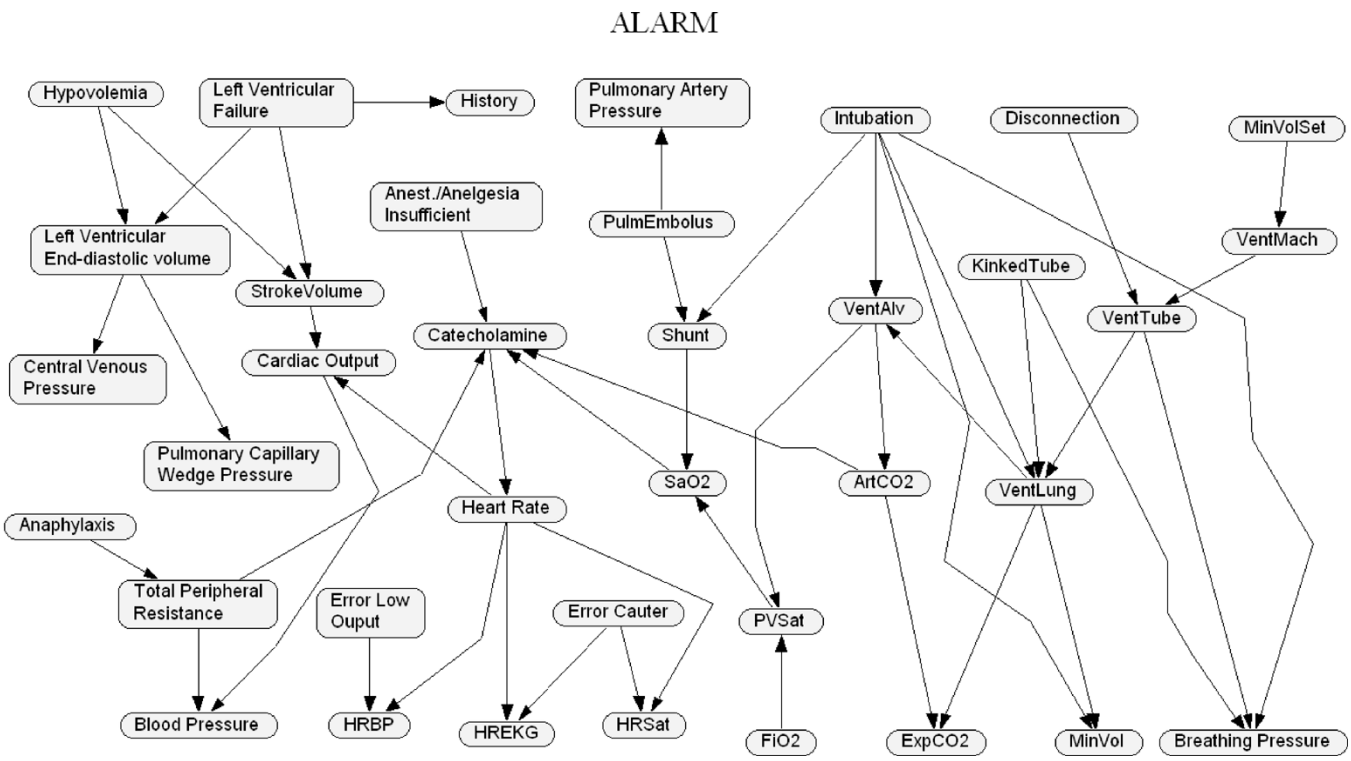

Fig. 3. ALARM network structure. This Bayesian network consists of 37 discrete variables. The figure was drawn by Netica software (Norsys Software Corp., Vancouver, BC, Canada). The local conditional probability distribution of each node (variable) is not shown here.

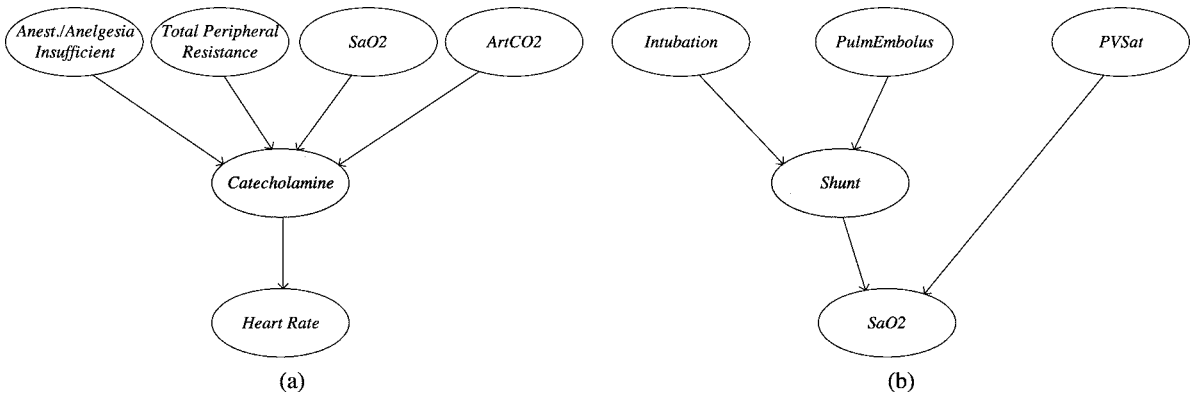

Fig. 4. Local network structure around (a) *Catecholamine* and (b) *Shunt*. Nodes other than the Markov blanket members of each class variable are not shown here. If the test case has all these variables instantiated (except the class variable), then only these Markov blanket members affect the prediction of the class label. However, the effective Markov blanket size can be increased if any one node is not given a value in a test case.

TABLE I
DESCRIPTION OF THE THREE BENCHMARK DATASETS. THE DATASETS ARE PREPARED IN THE FOLLOWING PROCEDURE. FIRST, A TEST DATASET OF THE SPECIFIED SIZE (RIGHTMOST COLUMN OF THE TABLE) WAS RESERVED FROM EACH BENCHMARK DATASET. THEN, TRAINING DATASETS OF VARYING SIZES WERE SAMPLED WITHOUT REPLACEMENT FROM THE REMAINING DATA EXAMPLES. TO MINIMIZE THE SAMPLING BIAS, TEN TRAINING DATASETS OF THE SAME SIZE WERE GENERATED FOR EACH BENCHMARK PROBLEM

	# of attributes	# of values of each attribute	# of classes	Total dataset size	Training dataset sizes	Test dataset size
breast	9	10	2	683	25, 50, 75, 100	500
mushroom	22	$2 \sim 12$	2	8124	25, 50, 75, 100	8000
chess	36	$2 \sim 3$	2	3196	25, 50, 75, 100	3000

a matrix with 30 attributes (excluding the class variable) and 120 data examples. Because there are not enough samples, three-, five-, and ten-fold cross-validations were used to assess the classification accuracy on this dataset.

B. Quality of Sampled Node-Orders

Before analyzing the classification experiments, we note the effectiveness of our sampling procedure from the viewpoint of the scoring metric. In the following experiments, the maximum in-degree of each

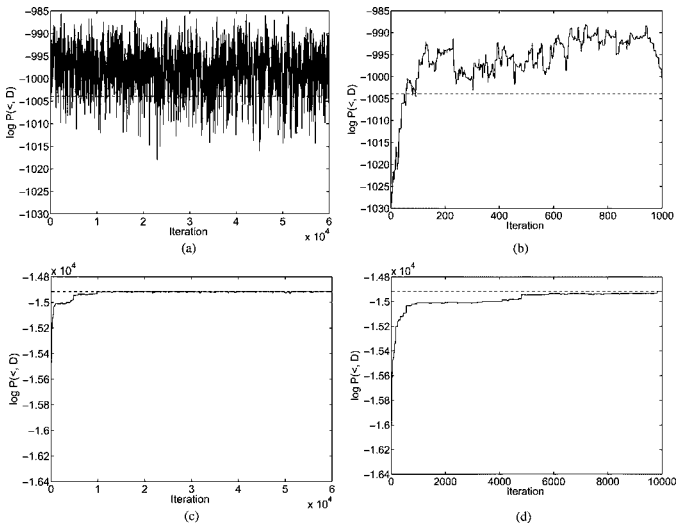

Fig. 5. MCMC sampling process for node-orders from the ALARM datasets. (a), (b) Training dataset of size 25. (c), (d) Training dataset of size 1000. The x axis shows iterations of the sampling process. The y axis denotes the log score of node-order ($\log P(\prec, D)$ using logarithms to base 2). The straight dashed-line shows the log score of one of the original node-orders of the ALARM network. The rightmost figures [(b) and (d)] show the initial portion of the entire sampling processes [(a) and (c)]. After a sufficient burn-in phase, the sampling process reaches a stable state where the sampled node-orders achieve scores as good as the original one. For a training dataset of size 25 [(a) and (b)], the sampling process might seem to be somewhat unstable; however, it is a scale effect of the y axis. The range of log scores in the graph is so narrow that small changes in score are emphasized.

node of Bayesian networks was set to 3. Fig. 5 shows the node-order sampling process from the ALARM datasets. Here, the starting order is the reversed version of one of the original node-orders of the ALARM network.[7] In the figure, the sampling process seems to have reached a stable stage regardless of the training dataset size. Specifically, each run eventually produced a node-order with as good a score as one of the original node-orders of the ALARM network. In addition, with smaller dataset sizes, the sampling process stabilized earlier. Although not shown here, other cases (different training dataset sizes, independently generated datasets of the same size, and four real-life datasets) also showed similar tendencies.

When the training dataset is sparse, it is possible for several distinct node-orders to give as good a score as the original. To examine this, we calculated Spearman's rank correlation coefficient between each sampled node-order and the original node-order of the ALARM network. Fig. 6 shows the rank correlation coefficient averaged over all sampled node-orders after the burn-in phase (>10 000th iteration) from the ten datasets for each training dataset size. As expected, the average rank correlation coefficient clearly increases with the training dataset size. In the most sparse case (training dataset size: 25), the average rank correlation coefficient is less than 0.1. The correlation value is close to 0.5 when the sample size is moderately small (500 and 1000). Also, the standard deviation of the correlation coefficient gradually decreases as the sample size increases. This result verifies our contention that it is hard to obtain a good node-order similar to the original when the given dataset is extremely sparse. As we noted in Section III-A, this will affect the classification performance of BMA of Bayesian network classifiers. In the next section, we empirically examine this through experiments on various datasets.

[7]Usually, the Bayesian network structure imposes only a partial ordering of the nodes. Hence, several node-orders could be consistent with a given network structure.

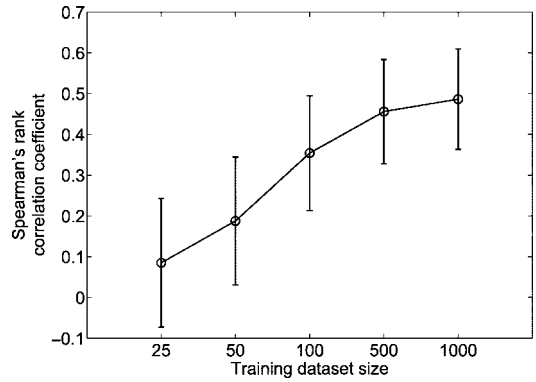

Fig. 6. Rank correlation coefficients between the sampled node-orders and one of the original node-orders of the ALARM network. The sampled node-orders after the burn-in phase (10 001–60 000) were used for the calculation. The correlation values were averaged over all sampled node-orders after the burn-in phase in the ten independent trials for each training dataset size. The average rank correlation coefficient clearly increases as the training dataset size grows, while the standard deviation gradually decreases.

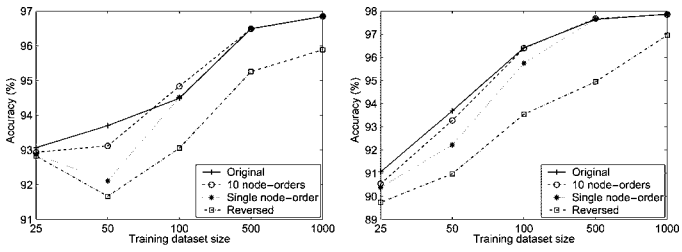

Fig. 7. Classification accuracy of BMA of Bayesian network classifiers according to training dataset size and averaged node-orders on (a) *Catecholamine* and (b) *Shunt*. Here, the training dataset size is related to the degree of sparseness. The influence of averaged node-orders on classification performance is clearly shown in the figure. BMA over the original node-order achieves the best performance in almost all cases. BMA over a reversed version of the original node-order shows the worst performance. Thus, the impact of node-order quality on the classification accuracy is clearly represented here. Further, BMA over ten node-orders is always better than averaging over a single node-order, demonstrating the effect of averaging over multiple node-orders on alleviating the node-order quality problem. (Other cases with 30 and five node-orders also show similar tendencies.) The difference in performance is especially apparent when the training dataset size is 50 and 100.

In the classification experiments, we selected 30 node-orders after the burn-in phase, i.e., every 1667th node-order from the 10 001st iteration for averaging over node-orders of (11). Among these node-orders, 30, 10 (every third node-order), and 5 (every sixth node-order) node-orders were used to investigate the effect of the number of averaged node-orders. Use of only one node-order (the first) was also tested for comparison with averaging over a single node-order.

C. Classification Performance Without Missing Variables

First, we show the experimental results on the test datasets without unobserved variables. In this case, the dependency between the quality of the node-order and classification performance is weaker than the case with missing variables, because only the Markov blanket members of the class variable are related to the classification. Fig. 7 shows the classification results on two synthetic problems, *Catecholamine* and *Shunt*. In the figure, the classification accuracy of averaging over the reversed version of the original node-order is always the lowest. In contrast, averaging over the original node-order is the best in almost all cases. Only in one case is its performance lower than that of

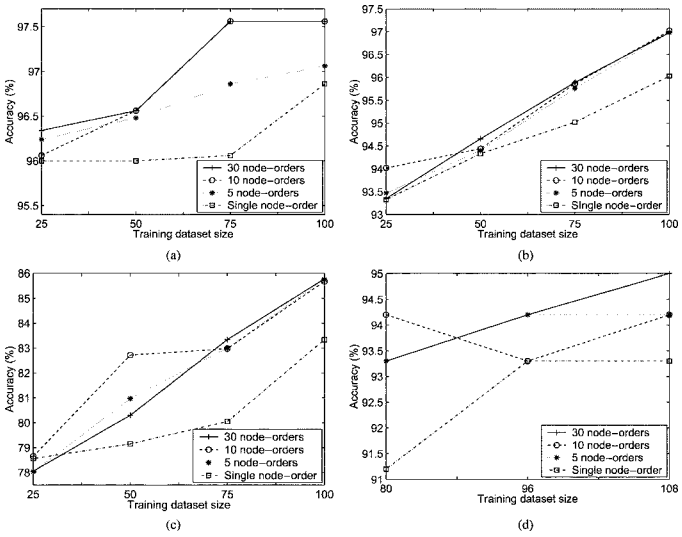

Fig. 8. Performance on real-life datasets. (a) *breast*, (b) *mushroom*, (c) *chess*, and (d) gene expression datasets. Shown are classification accuracy of BMA of Bayesian network classifiers according to the training dataset size and the number of averaged node-orders. In most cases, model averaging over a single node-order shows lower classification accuracy than averaging over multiple node-orders. For the gene expression dataset, we applied three-, five-, and tenfold cross-validations for assessing classification accuracy.

averaging over multiple node-orders. Thus, we demonstrated the effect of node-order quality on the classification performance of BMA of Bayesian network classifiers.

Also in the figure, we can observe that averaging over ten node-orders gives better performance than averaging over a single node-order, regardless of the problem and training dataset size. Here, we only report the results of averaging over ten node-orders, since classification performance with 30 or five node-orders is similar to the case of averaging over ten node-orders. The difference in the classification performance between multiple node-orders and a single node-order nearly vanishes when the training dataset size is 500 or 1000. One explanation is that a sufficiently good node-order can be obtained by MCMC-based sampling only with enough training examples, as we showed in Section IV-B (see Fig. 6). In sparse cases, averaging over multiple node-orders obtained from MCMC sampling can minimize the risk of choosing one bad node-order for averaging. Hence, it can be concluded that averaging over multiple node-orders is especially effective when the given training dataset is very sparse.

Fig. 8 shows the experimental results for the four real-life datasets: three machine-learning benchmark problems and gene expression data. In the three benchmark datasets, BMA over multiple node-orders shows better performance than averaging over a single node-order in most cases, regardless of the training dataset size. (The only exception is the *chess* problem with a training dataset size of 25.) For the gene expression dataset, we applied several k-fold cross-validations to simulate diverse situations. Specifically, three-, five-, and tenfold cross-validations were used in the experiments. Because the given dataset size is 120, the training dataset size corresponds to 80, 96, and 108, respectively. Here, averaging over multiple node-orders also shows higher classification accuracy than averaging over a single node-order. In the experiments on these real-life datasets, we could not assess node-order quality because we do not know the problem structure. However, experimental results on real-life datasets are similar to those for synthetic problems, demonstrating that the effect of averaging over multiple node-orders to reduce the node-order quality problem can also be applied to more realistic situations.

From these experimental results, we conclude that BMA of Bayesian network classifiers over multiple node-orders generally improves classification performance on sparse data. The log score $\log P(\prec | D)$ cannot be a good guide for sampling node-orders when the given dataset is very sparse. Therefore, it is hard to obtain a good node-order for model averaging. Then, averaging over several node-orders obtained from MCMC sampling can be a reasonable solution for improving the classification accuracy.

D. Classification Performance With Missing Variables

In this section, we analyze the effect of missing variables on the classification performance of BMA of Bayesian network classifiers. Here, we assume that only the test data have unobserved variables. In other words, we apply the Bayesian network classifier learned from complete training data to the classification of test cases with missing variables. As noted before, unobserved variables in test cases can have the effect of extending the Markov blanket of the class variable (see Fig. 1). Thus, it is expected that BMA over multiple node-orders is more effective for that case.

We tested with various percentages of missing variables in the data, i.e., 5%, 10%, and 20%. For example, with 10% missing variables with the ALARM dataset, we removed the values of four randomly chosen variables (10% of 36 feature variables). There are several ways to deal with missing variables in classifier systems, e.g., imputing random values, interpolation, and adding another value for the missing variable. In addition, probabilistic graphical models such as Bayesian networks can handle the situation with missing variables neatly, in principle. They can infer the conditional probability of interest given the observations even if some variables are not instantiated. However, all missing variables should be marginalized out to calculate the necessary conditional probabilities. This averaging task often has a heavy computational cost, which is exponential in the number of missing variables. In fact, probabilistic inference of arbitrary probability from a Bayesian network is known to be an NP-hard problem [23]. In the following experiments, we imputed random values because the computational complexity of BMA of Bayesian network classifiers is also high (see the Appendix). It should be noted that while computing the marginal probability in Bayesian network is generally hard, it is also applicable in some of our experiments. We avoided the additional cost of marginalization because our focus was on the effect of unobserved variables on classification performance with regard to the number of averaged node-orders.

Fig. 9 shows the classification accuracy of BMA of Bayesian network classifiers for datasets with missing variables. Here, the experimental results are from training datasets of size 100 or tenfold cross-validation (other cases also show similar tendencies). Here, it turns out that averaging over a single node-order always shows lower performance than averaging over multiple node-orders. In addition, the number of averaged node-orders in BMA over multiple node-orders influences classification performance, although the variation is not obvious in some cases. Moreover, the degree of divergence in classification accuracy between a single node-order and multiple node-orders generally increases with the number of missing variables. This result coincides with our expectation that averaging over multiple node-orders would be especially effective when there are unobserved variables in test cases, because of the possible extension of Markov blanket size of the class variable. From the experimental results, we can conclude that BMA of Bayesian network classifiers over multiple node-orders is especially helpful in classifying examples with missing variables compared to the averaging over a single node-order.

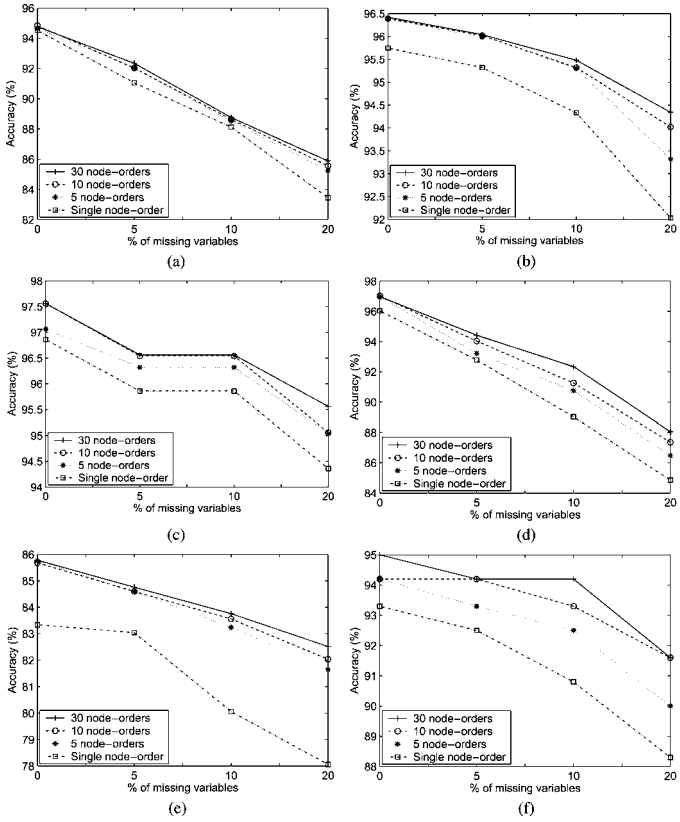

Fig. 9. Performance on datasets with unobserved variables. (a) *Catecholamine*, (b) *Shunt*, (c) *breast*, (d) *mushroom*, (e) *chess*, and (e) gene expression data. Shown are classification accuracy of BMA of Bayesian network classifiers according to the percentage of missing variables and the number of averaged node-orders. The training dataset size is 100 (except for the gene expression dataset, where it is 108). It can be seen that the classification performance of model averaging over multiple node-orders is always better than averaging over a single node-order. In particular, the difference in classification accuracy is nearly proportional to the number of unobserved variables. For the *breast* dataset [(c)], the number of missing variables at 5% and 10% is the same.

V. Conclusion

We presented a method for improving the classification accuracy of general Bayesian network classifiers on sparse data using Bayesian model averaging over multiple node-orders. The rationale behind our approach is that it can be hard to obtain a good node-order for averaging when the given training dataset is sparse. We demonstrated this through experiments on synthetic datasets generated from a known Bayesian network. We also showed that the quality of node-order can influence the classification performance. The experimental results on various sparse datasets further confirmed that model averaging over multiple node-orders outperforms averaging over a single node-order. Another issue examined is the effect of unobserved variables in test cases. Missing variables in test cases are expected to extend the effective Markov blanket size of the class variable. In our experiments, averaging over multiple node-orders always showed better classification performance than using a single node-order. Moreover, the difference in classification accuracy increases as the number of unobserved variables increases. Thus, the results confirm the relative benefit of averaging over multiple node-orders on sparse and noisy data. Finally, it should be noted that averaging over multiple node-orders imposes a very high computational cost despite its advantage with respect to classification accuracy. We suggest that the use of BMA of Bayesian network classifiers over multiple node-orders for the situation requiring maximal classification accuracy on sparse and noisy data are justified, despite the cost in time and space.

Appendix
Computational Complexity of BMA Over Multiple Node-Orders

The computational complexity of classification using model averaging over a single node-order can be deduced from (9). Here, we enumerate all possible parent sets of each node given a fixed node-order (i.e., $\mathcal{Z}_{i,\prec}$ in the equation) for calculating the likelihood. For each parent set $\mathbf{Z}$, we calculate $S(X_i;\mathbf{Z}|G,D)$, which corresponds to $\rho(X_i|\mathbf{Z})\cdot\prod_{j=1}^{q_i}(\Gamma(\alpha_{ij}))/(\Gamma(\alpha_{ij}+N_{ij}))\cdot\prod_{k=1}^{r_i}(\Gamma(\alpha_{ijk}+N_{ijk}))/(\Gamma(\alpha_{ijk}))$ (see Section II-B). Here, $\rho(X_i|\mathbf{Z})$ can be calculated in constant time if the decomposable prior used in our experiments is adopted [see (13)]. For $\prod_{j=1}^{q_i}(\Gamma(\alpha_{ij}))/(\Gamma(\alpha_{ij}+N_{ij}))\cdot\prod_{k=1}^{r_i}(\Gamma(\alpha_{ijk}+N_{ijk}))/(\Gamma(\alpha_{ijk}))$, we require $O(q_i\cdot r_i)$ space and $O(M\cdot q_i\cdot r_i)$ time, where M denotes the number of training examples. We can also obtain $(\alpha_{ij_{\mathbf{U}}k_{\mathbf{U}}}+N_{ij_{\mathbf{U}}k_{\mathbf{U}}})/(\alpha_{ij_{\mathbf{U}}}+N_{ij_{\mathbf{U}}})$ through the above procedure. Now, the number of possible parent sets of each node (size of $\mathcal{Z}_{i,\prec}$) corresponds to $\sum_{w=0}^{l}\binom{v-1}{w}$, where l denotes the maximum number of parents and v denotes the position of node in the given node-order. (If $v\leq w$, then $\binom{v-1}{w}=0$.) The complexity of $\sum_{w=0}^{l}\binom{v-1}{w}$ is $O(v^l)$. Therefore, we require $O(v^l\cdot M\cdot(\max(r))^l\cdot\max(r))$ time to calculate $\sum_{\mathbf{Z}\in\mathcal{Z}_{i,\prec}}S(X_i;\mathbf{Z}|G,D)$ with $O((\max(r))^l\cdot\max(r))$ space for each $\mathbf{Z}$.[8] Here, q_i is bounded by $(\max(r))^l$, where $\max(r)$ denotes the maximum r_i value across all i. Finally, if the number of variables is n, we should consider $O(1^l+2^l+\ldots+n^l)$ cases. Because l is positive, the complexity of the worst case amounts to $O(n^{2l})$. Thus, the entire time complexity of (9) is bounded by $O(n^{2l}\cdot M\cdot(\max(r))^l\cdot\max(r))$. To average over multiple node-orders, we calculate (11). The computational complexity is $O(T\cdot n^{2l}\cdot M\cdot(\max(r))^l\cdot\max(r))$, where T is the number of averaged node-orders. In addition, we should simulate a Markov chain for sampling the node-orders. At each iteration, the likelihood of node-order $P(D|\prec)$ should be calculated [see (12) and Section III-B]. This likelihood corresponds to $\prod_i\sum_{\mathbf{Z}\in\mathcal{Z}_{i,\prec}}S(X_i;\mathbf{Z}|G,D)$ [see (5) and (9)]. Thus, each iteration of the Markov chain sampling procedure also takes $O(n^{2l}\cdot M\cdot(\max(r))^l\cdot\max(r))$ time.

Acknowledgment

The authors would like to thank the associate editor and the anonymous reviewers for their insightful comments and valuable suggestions.

References

[1] F. V. Jensen, *Bayesian Networks and Decision Graphs*. New York: Springer-Verlag, 2001.
[2] D. Heckerman, "A tutorial on learning with Bayesian networks," in *Learning in Graphical Models*, M. I. Jordan, Ed. Cambridge, MA: MIT Press, 1999, pp. 301–354.
[3] K. B. Korb and A. E. Nicholson, *Bayesian Artificial Intelligence*. Boca Raton, FL: CRC, 2004.
[4] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Mach. Learn.*, vol. 29, no. 2/3, pp. 131–163, 1997.
[5] N. Friedman and M. Goldszmidt, "Learning Bayesian networks with local structure," in *Learning in Graphical Models*, M. I. Jordan, Ed. Cambridge, MA: MIT Press, 1999, pp. 421–459.
[6] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: A tutorial," *Statist. Sci.*, vol. 14, no. 4, pp. 382–401, 1999.
[7] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, pp. 181–207, 2003.

[8]The time complexity for calculating $\sum_{\mathbf{Z}\in\mathcal{Z}_{i,\prec}}S(X_i;\mathbf{Z}|G,D)$ can be reduced by some techniques such as caching. For example, we can consider all possible parent sets of each node in advance and store the highest $S(X_i;\mathbf{Z}|G,D)$ values.

[8] B. Clarke, "Comparing Bayes model averaging and stacking when model approximation error cannot be ignored," *J. Mach. Learn. Res.*, vol. 4, pp. 683–712, Oct. 2003.

[9] D. Dash and G. F. Cooper, "Exact model averaging with naive Bayesian classifiers," in *Proc. 19th Int. Conf. Machine Learning (ICML'02)*, Sydney, Australia, Jul. 2002, pp. 91–98.

[10] J. Cerquides and R. López de Màntaras, "Tractable Bayesian learning of tree augmented naive Bayes models," in *Proc. 20th Int. Conf. Machine Learning (ICML'03)*, Washington, DC, Aug. 2003, pp. 75–82.

[11] D. Dash and G. F. Cooper, "Model averaging for prediction with discrete Bayesian networks," *J. Mach. Learn. Res.*, vol. 5, pp. 1177–1203, Sep. 2004.

[12] N. Friedman and D. Koller, "Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks," *Mach. Learn.*, vol. 50, no. 1/2, pp. 95–125, 2003.

[13] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, "Introducing Markov chain Monte Carlo," in *Markov Chain Monte Carlo in Practice*, W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, Eds. Boca Raton, FL: CRC, 1998, ch. 1, pp. 1–19.

[14] D. Heckerrnan, D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Mach. Learn.*, vol. 20, no. 3, pp. 197–243, 1995.

[15] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, and Y. Yurramendi, "Learning Bayesian network structures by searching for the best ordering with genetic algorithms," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 26, no. 4, pp. 487–493, Aug. 1996.

[16] W. A. Wright, "Bayesian approach to neural-network modeling with input uncertainty," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1261–1270, 1999.

[17] B.-T. Zhang and D.-Y. Cho, "System identification using evolutionary Markov chain Monte Carlo," *J. Syst. Architect.*, vol. 47, no. 7, pp. 587–599, 2001.

[18] Y. Wang and S.-C. Zhu, "Analysis and synthesis of textured motion: Particles and waves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1348–1363, Oct. 2004.

[19] I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper, "The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks," in *Proc. 2nd Eur. Conf. Artificial Intelligence in Medicine (AIME'89)*, London, U.K., 1989, pp. 247–256.

[20] S. Knudsen, *A Biologist's Guide to Analysis of DNA Microarray Data*. New York: Wiley, 2002.

[21] M. H. Cheok, W. Yang, C.-H. Pui, J. R. Downing, C. Cheng, C. W. Naeve, M. V. Relling, and W. E. Evans, "Treatment-specific changes in gene expression discriminate *in vivo* drug response in human leukemia cells," *Nature Genetics*, vol. 34, no. 1, pp. 85–90, 2003.

[22] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[23] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artif. Intell.*, vol. 42, no. 2/3, pp. 393–405, 1990.