Learning Sparse Gaussian Bayesian Network Structure by Variable Grouping

Jie Yang†, Henry C.M. Leung†, S.M. Yiu†, Yunpeng Cai‡, Francis Y.L. Chin†*
† Department of Computer Science, The University of Hong Kong, Hong Kong
Email: chin@cs.hku.hk
‡ Shenzhen Institutes of Advance Technology & Key Lab for Health Informatics
Chinese Academy of Sciences, China

Abstract—Bayesian networks (BNs) are popular for modeling conditional distributions of variables and causal relationships, especially in biological settings such as protein interactions, gene regulatory networks and microbial interactions. Previous BN structure learning algorithms treat variables with similar tendency separately. In this paper, we propose a grouped sparse Gaussian BN (GSGBN) structure learning algorithm which creates BN based on three assumptions: (i) variables follow a multivariate Gaussian distribution, (ii) the network only contains a few edges (sparse), (iii) similar variables have less-divergent sets of parents, while not-so-similar ones should have divergent sets of parents (variable grouping). We use $L_1$ regularization to make the learned network sparse, and another term to incorporate shared information among variables. For similar variables, GSGBN tends to penalize the differences of similar variables' parent sets more, compared to those not-so-similar variables' parent sets. The similarity of variables is learned from the data by alternating optimization, without prior domain knowledge. Based on this new definition of the optimal BN, a coordinate descent algorithm and a projected gradient descent algorithm are developed to obtain edges of the network and also similarity of variables. Experimental results on both simulated and real datasets show that GSGBN has substantially superior prediction performance for structure learning when compared to several existing algorithms.

I. INTRODUCTION

A Bayesian network (BN) can be used to represent the probabilistic relationship of variables [14]. It is a graphical model for a set of dependent variables defined over a directed acyclic graph (DAG). Each node in the network represents a random variable, whose probability distribution can be calculated from the variables represented by its parent nodes in the DAG. Thus, a BN represents a joint probability distribution over all variables and the joint distribution can be decomposed into the products of the probability distribution of each variable conditioned on its parents. We focus on BNs for bioinformatics related applications. Learning the structure of a BN is important for many tasks in bioinformatics involving, for example, protein interactions, gene regulatory networks and microbial interactions. There are two kinds of algorithms for learning a BN’s structure: constraints-based and score-based approaches [13]. In constraints-based approaches, BNs are learned independently by testing conditional independencies of subsets of variables. For example, the GS [16] algorithm, which is constraints-based, identifies the potential parents of each variable by testing conditional relationships (e.g. testing Markov blankets) and then heuristically modifies the DAG based on the potential parents detected (e.g. removing edges or changing edge directions). Other constraints-based methods such as TC-bw [17], IAMB [21] and HITON [1] follow a similar approach. Unfortunately, constraints-based approaches are usually prone to errors when the number of samples is small compared with the number of variables because some edges might be missing in the first stage.

Because the amount of biological data is usually limited, we shall focus on score-based structure learning algorithms to obtain the best BN for modeling a set of variables. Score-based algorithms have two main components: (1) a global scoring function to evaluate the goodness of a network and (2) a search procedure to find the optimal network. Many score functions have been proposed to evaluate the goodness of a BN. BDe [11] and BGe [10] scores were proposed with the assumption of multinomial and Gaussian distribution for discrete and continuous variables, respectively. However, searching for the optimal BN with good BDe and BGe scores is NP-hard [3] even if we restrict each node to having at most two parents [4]. Biological data is usually limited (hundreds of samples only), but the number of estimated parameters in the BN is large even when the number of variables is small (e.g. approximately 2500 parameters for only 50 variables). In these circumstances, the BN will usually overfit the biological data. To avoid over-fitting and to reduce the size of searching space, we should further restrict the BN. One common approach is to assume that the optimal BN is simple. The sparse candidate algorithm [9] restricts the size of the BN by bounding the number of parents for each node. However, this restriction makes the algorithm not useful for many applications where some nodes in the network may have more parents.

Lasso ($L_1$ regularization) [20] is the most popular approach to enforce sparsity for linear regression. Each variable in BN follows a multivariable Gaussian distribution conditioned on its parents, and finding the relationship with its parents is equivalent to fitting a linear regression over its parents. The Lasso method can be applied for learning sparse BN (a network with a few edges). Unlike the sparse candidate algorithm which needs to specify the number of parents, L1MB [18] penalizes the regression matrix with $L_1$ regularization for determining parents automatically and later heuristically finds the optimal DAG. SBN [12] is similar to L1MB with
We define **similar variables** as variables having less-divergent sets of parents and with similar behaviour (note that variables with opposite values in all samples are also considered as similar variables), while the **not-so-similar variables** should have divergent sets of parents. Let $P_a(X)$ be the set of $X$’s parent variables. The similarity $S(X,Y)$ between two variables $X$ and $Y$ is defined as the difference of their sets of parent nodes, i.e. $P_a(X) \oplus P_a(Y)$ where $\oplus$ is the symmetric difference. This measure is equivalent to the Hamming distance between two binary vectors which represent the set of elements. Thus, the two variables $X$ and $Y$ are similar if $S(X,Y) = 0$ and their difference is measured according to the value of $S(X,Y)$. E.g., in Fig. 1, variables $X_6$ with $P_a(X_6) = \{X_2,X_3,X_4\}$ and $X_7$ with $P_a(X_7) = \{X_3,X_4\}$ are similar, as they have one only different parent $P_a(X_6) \oplus P_a(X_7) = \{X_2\}$. Variables $X_7$ and $X_8$ are not similar as they do not share any parents in the BN and have five different parents $P_a(X_7) \oplus P_a(X_8) = \{X_1,X_3,X_4,X_5,X_6\}$. With this measure of similarity, we can formulate the variable grouping idea as a minimization problem. With the observation that $S(X,Y) \geq 0$ and its probability distribution decreases exponentially with its increased value, we propose a score function which approximates the distribution of node similarity $S(X,Y)$ by different and independent Laplacian distributions. Based on this score function which includes $S(X,Y)$, the similar variables can be grouped together by sharing many common parents, while those not-so-similar variables might not share any common parents. Thus the searching space for the hidden BN can be reduced. In this paper, we propose a sparse Gaussian BN structure learning algorithm **GSGBN** based on $L_1$ regularization and variable grouping. GSGBN contains two stages: firstly, we learn the pairs of similar nodes from the data by alternating optimization; secondly, we find the DAG heuristically by an ordering-based search as in [19].

Our contributions are: a). **GSGBN** is the first sparse BN structure learning algorithm that considers the shared information among variables through their similarity; b). **GSGBN** is capable of learning the similarity of the variables (by constructing a grouping matrix) from the data automatically without any prior domain knowledge; c). **GSGBN** uses an efficient and effective algorithm to learn the similarity of the variables and the BN through regression and regularization by alternating optimization. Two algorithms have been developed: the coordinate descent algorithm (for finding regression matrix) and the projected gradient descent algorithm [5] (for finding grouping matrix), which guarantee finding the optimal regression matrix when the grouping matrix is fixed and finding the optimal grouping matrix when the regression matrix is fixed. When compared with other learning algorithms on eight benchmark datasets, **GSGBN**, which considers shared information (variable grouping), has the best performances in terms of both sensitivity and specificity for predicting the structure of BNs. For example, in the experiment on the “Water” benchmark dataset (Table I), **GSGBN** increased sensitively from 0.409 (second best result) to 0.833 with slight increase in specificity (from 0.980 to 0.986). Application of these eight algorithms on real biological data for studying microbial interactions also shows that **GSGBN** performs better for predicting the observed values of the variables (decreasing the error in prediction by about 10%).

![Fig. 1. An example for variable grouping](image-url)
calculating the log likelihood of \( p(X_j|P_a(X_j)) \), it is equivalent to fitting a linear regression model from \( P_a(X_j) \) to \( X_j \):

\[
X_j = \sum_{x_i \in P_a(X_j)} X_i B_{kj} + \epsilon_j, \epsilon_j \sim \mathcal{N}(0, \sigma_j^2).
\]

where \( \epsilon_j \) is the noise which follows a Gaussian distribution.

Suppose we have \( m \) samples each with \( n \) variables, and we want to learn a Gaussian BN that can best explain these samples. Formally, given a matrix \( X_{m \times n} \) with \( m \) samples and \( n \) variables, where \( i^{th} \) row \( X_i \) is the variable vector for the \( i^{th} \) sample, and \( j^{th} \) column \( X_j \) is the sample vector for the \( j^{th} \) variable, we want to find the optimal DAG \( \hat{G} \) encoded by the regression matrix \( \hat{B} \) with minimum \( \{ f(B) + g(B) \} \) (error term \( f(B) \) represents the error in predicting \( X_j \) from its parents \( P_a(X_j) \) and regularization term \( g(B) \) represent the penalty for some special network topologies):

\[
\hat{B} = \arg\min_B \{ f(B) + g(B) \}, \quad s.t. \mathcal{G}(B) \in \text{DAG} \tag{1}
\]

where

\[
f(B) = \sum_{j=1}^n (X_{ij} - XB_{ij})^T (X_{ij} - XB_{ij}),
\]

\( \mathcal{G}(B) \) represents the DAG with edges defined by nonzero elements of \( B \). Different algorithms have different definitions for \( g(B) \). For L1MB, L1 penalty [18], [20] is used on each column of \( \hat{B} \) for sparsity in \( G \), i.e. \( g(B) = \lambda \sum_{j=1}^n ||B_{ij}||_1 \) where \( ||B_{ij}||_1 = \sum_{k=1}^{n} ||B_{kj}|| \). The positive regularization parameter \( \lambda \) is for restricting the number of parents of each node, which in practice can be learned from data (e.g by cross validation). Without the DAG constraint, Eq. (1) is equivalent to Lasso [20]. Without loss of generality, each column \( X_{ij} \) is normalized with zero mean and unit variance.

### III. Method

#### A. Motivation

Since finding the optimal BN is NP-hard [3], it is unlikely to find the optimal BN other than exhaustion. In addition, existing algorithms do not employ the information of similar variables to reduce the size of searching space. Our GSSGBN algorithm mainly takes advantages of the following three assumptions:

a). **Gaussian Distribution** Each variable \( X_j \) follows a multivariate Gaussian distribution conditioned on its parents, which is equivalent to fitting a linear regression model from parent variables \( P_a(X_j) \) to \( X_j \); b). **Sparse Network** The optimal BN should be sparse (with only small number of edges), which significantly reduces the size of parent set for each variable; c). **Variable Grouping** To incorporate shared information among variables (similar variables with less-divergent sets of parents, while not-so-similar ones with divergent sets of parents).

The first two assumptions can be modelled by enforcing \( L_1 \) penalty on each column of matrix \( B \) when performing linear regression. The technique for modeling **Variable Grouping** will be presented in next section.

B. **Grouping**

To making similar variables have less-divergent set of parents, it is intuitive to assign the penalty: \( W_{ij} \sum_{k=1}^n ||B_{kj}||_1 - ||B_{kj}|| \) such that \( W_{ij} (W_{ij} > 0) \) and the grouping matrix \( W \) represents the similarity of two variables \( X_i \) and \( X_j \). The summation is for differences of parents. Mathematically, \( |a| \) is the \( L_0 \) regularization with \( |a| = 1 \) if \( a \neq 0 \) otherwise \( |a| = 0 \). Intuitively, if variables \( X_i \) and \( X_j \) are similar, \( W_{ij} \) should be large, which forces the corresponding parent sets to be similar; however, if two variables \( X_i \) and \( X_j \) are not-so-similar, \( W_{ij} \) should have small value and parent sets of \( X_i \) and \( X_j \) can be more different. However solving the \( L_0 \) regularization problem is NP-hard, we consider \( L_1 \) regularization \( (W_{ij} \sum_{k=1}^n ||B_{kj}||_1 - ||B_{kj}|| \) which is an approximation to \( L_0 (W_{ij} \sum_{k=1}^n ||B_{kj}||_1 - ||B_{kj}||) \).

C. **Problem Modeling**

Given a known dataset \( X \), the probability \( P(B|X) \) is proportional to \( P(X|B) \cdot P(B) \). The conditional probability \( P(X|B) \) follows a Gaussian distribution. Therefore, finding \( B \) that maximizes likelihood \( P(X|B) \) is equivalent to finding \( B \) that minimizes \( f(B) \) in Eq. (1). Considering the assumptions of **Sparse Network and Variable Grouping**, we model the prior probability distribution of \( B \) as the product of \( L(B) \) (for sparsity) and \( F(B) \) (for variable grouping).

\[
P(B) = \frac{1}{C} L(B) \cdot F(B);\tag{2}
\]

where \( C \) is the normalization constant.

\[
L(B) = \prod_{i \neq j} \text{Lap}(B_{ij}), \quad F(B) = \prod_{i \neq j} \text{Grp}(B,i,j);\tag{3}
\]

\[
\text{Lap}(B_{ij}) = \frac{\lambda_1}{2} e^{-\lambda_1 ||B_{ij}||}, \quad \text{Grp}(B,i,j) = \frac{\lambda_2}{2} W_{ij} e^{-\lambda_2 W_{ij} \sum ||B_{kj}||} ||B_{kj}||;\tag{4}
\]

\[
\sum_{i,j} W_{ij} = 1 \text{ and } W_{ii} = 0, \forall i;\tag{5}
\]

\[
W_{ij} = W_{ji}, 0 < W_{ij} < 1, \forall i \neq j.
\]

In (2), \( L(B) \) is used for modeling **Sparse Network** and \( B_{ij} \) follows an independent Laplacian distribution \( \text{Lap}(B_{ij}) \), i.e. \( B_{ij} \sim \text{Laplace}(0, \lambda_1^{-1}) \). The Laplacian distribution assumption of \( B \) is to impose \( L_1 \) penalty on \( B \) to make \( B \) sparse [20]. The probability distribution function for a variable \( x \) that follows a Laplace \( (0, \lambda_1^{-1}) \) is \( p(x) = \frac{\lambda_1}{2} e^{-\lambda_1 |x|} \).** Variable Grouping**: We have analyzed the known DAG structures for benchmark datasets by studying the distribution of frequencies, i.e. number of pairs of variables \( X, Y \) with \( t = S(X,Y) \) different parents. Unfortunately, we have not found any single distribution (Gaussian, Poisson etc.) can fit the distribution well. Thus we want to learn the distribution from the data. As we found that the number of pairs with parent differences of \( t \) decrease exponentially, we use Laplacian distributions to approximate the hidden distribution of differences of parents. To formulate Variable Grouping assumption, we define the grouping matrix \( W \) where \( W_{ij} \) is the parameter for the Laplacian distribution of \( S(X_i, X_j) \), i.e.
\[ \sum_k \|B_k \| - \|B_k^0 \| \sim \text{Laplace}(0, (\lambda_2 W_{ij})^{-1}). \] If \( W_{ij} \) is large, the \( S(X_i, X_j) \) is small which we say \( X_i \) and \( X_j \) are similar. However, small \( W_{ij} \) indicates \( S(X_i, X_j) \) is large, in this case, \( X_i \) and \( X_j \) are very different. \( \lambda_1 \) and \( \lambda_2 \) (\( \lambda_1, \lambda_2 > 0 \)) are regularization parameters that can be selected by cross-validation in practice. By calculating the negative log-likelihood of \( P(X|B) \cdot P(B) \), we get the following equation:

\[
\hat{B}, \hat{W} = \arg\min_{B,W} \{ f(B) + g(B,W) \}, \text{ s.t. } \forall X_i \in \text{DAG}
\] (5)

where

\[
f(B) = \sum_{j=1}^{n} (X_{-j} - XB_{-j})^T \cdot (X_{-j} - XB_{-j});
\]

\[
g(B,W) = \lambda_1 \sum_{j=1}^{n} \|B_{ij}\|_1 + \lambda_2 \sum_{i \neq j} W_{ij} \sum_k \|B_k - B_k^0\| - \sum_{i \neq j} \log W_{ij};
\]

and \( W \) is constrained by (4). Instead of knowing the prior knowledge of the grouping matrix \( W \), we learn \( W \) from the data.

D. Parameter Estimation

We use a two-stage approach to solve (5). At the first stage, we learn grouping matrix \( W \) without the DAG constraint; at the second stage, we heuristically search the DAG and \( B \) that optimize (5) with learned \( W \) at the first stage.

1) Similarity Estimation: At this stage we only consider solving (5) without DAG constraint. We propose to use alternating optimization [15] to solve (5), that is to say: with fixed \( W \), find and update the optimal \( B \); with fixed \( B \), find and update optimal \( W \). Repeat the above procedure until convergence or reaching the maximum number of iterations.

**Theorem 1.** Eq. (5) is convex with respect to \( W \) if \( B \) is fixed.

However, (5) may not be convex with fixed \( W \). To make (5) convex, we further restrict \( W \) by the following equation:

\[
\lambda_2 \sum_{j \neq i, j = 1}^{n} W_{ij} \leq \lambda_1, \forall i = 1, \ldots, n.
\] (6)

Even though Eq. (5) is non-convex, it is convex if either \( B \) or \( W \) is fixed and Eq.(5) satisfies the constraints given by (6). We can apply alternating optimization by alternatively solving two subproblems until \( f(B) + g(B,W) \) converges.

**Theorem 2.** Eq. (5) is convex with respect to \( B \) if \( W \) is fixed and satisfies the constraints as given by (4) and (6).

**Theorem 3.** The alternating optimization approach to solve Eq. (5) by fixing \( W \) and \( B \) alternatively as discussed above can always converge.

Thus, we can apply alternating optimization by alternatively solving two subproblems until \( f(B) + g(B,W) \) converges. The proofs for theorems are omitted. The detailed procedure is:

- With fixed \( W \), we find the optimal \( B \) by coordinate descent algorithm [8]. Since we cannot separate the regularization term \( g(B) \), i.e. \( g(B) = \sum_g g(g(B_{ij})) \), coordinate descent algorithm may not guarantee to find the global optimal \( B \) [7]. However it works well in practice with many advantages (accurate, easy for implementation and fast). We also implemented ADMM (alternating direction method of multipliers) algorithm [2], which guarantees finding the global optimal \( B \). Coordinate descent always reports the same results as ADMM by comparison. At each step of coordinate descent, obtaining

\[
\hat{B}_{ij} = \arg\min_{B_{ij}} \{ f(B) + g(B,W) \}
\]

is equivalent to getting the optimal \( \hat{t} \) of the following function:

\[
\hat{t} = \arg\min_{t} \{ a_0 (t - b_0)^2 + \sum_{i = 1}^{l} a_i |t - b_i| \}
\] (7)

Solving (7) can be done in \( O(n \log n) \) time.

- With fixed \( B \), we find the optimal \( W \) by Algorithm 1 (projected gradient descent algorithm) [5].

Since repeating the above procedure always decreases the value of \( f(B) + g(B,W) \), it always stops.

**Algorithm 1:** Find optimal \( W \) with fixed \( B \)

**Input:**

- \( B \), \( \lambda_1 \) and \( \lambda_2 \);
- \( \gamma \): learning rate for gradient descent algorithm;
- \( \varepsilon \): convergence threshold.

**Output:**

- \( W \): an \( n \times n \) variable grouping matrix.

1. Set \( W_{ij}^{(0)} = \frac{1}{n(n-1)} \), \( \forall i \neq j \) and \( W_{ii}^{(0)} = 0, \forall i; \)
2. \( t = 0; \)
3. repeat
4. \( W^* = W^{(t)} - \gamma \nabla g(B,W^{(t)}) \) \( \nabla g(B,W^{(t)}) = [\frac{\partial g(B,W^{(t)})}{\partial W_{ij}^{(t)}}]; \)
5. \( W^{(t+1)} = \text{Project } W^* \text{ to the hyperplane of (4) \& (6)}; \)
6. \( t = t + 1; \)
7. until \( |W^{(t)} - W^{(t-1)}| < \varepsilon; \)
8. \( W = W^*; \)

2) DAG Search: We heuristically search the DAG by ordering-based search [18], [19]. We start with an arbitrary order of variables and calculate the optimal DAG. At each step we only consider swapping adjacent variables:

\[ \ldots, X_i, X_{i+1}, \ldots \rightarrow \ldots, X_{i+1}, X_i, \ldots \].

We calculate the scores of all \( n - 1 \) successor moves (swapping adjacent variables) and only greedily choose the one with minimum negative log-likelihood. Repeat this strategy until convergence. Since the algorithm may not converge to global optimal solution, we try different starting orderings.

IV. Empirical Studies

We compared the performance of GSGBN (Availability: https://github.com/ch1ly/GSGBN.git) to another seven Gaussian BN structure learning algorithms: L1MB [18], SBN [12],
SC [9], GS [16], TC-bw [17], IAMB [21] and HITON [1]. Experiments show that GSGBN performs best on both simulation benchmark datasets and real-world microbial dataset.

A. Similarity Learning

We used eight benchmark datasets in BN Repository (http://www.bnlearn.com/bnrepository/). The number of nodes and edges for 8 benchmark datasets are Factor (27, 68), Alarm (37, 46), Mildew (35, 46), Insurance (27, 52), Barley (48, 84), Carpo (61,74), Hailfinder (56, 66), Water (32,66), i.e. “Factor” dataset has 27 nodes and 68 edges. To confirm the similarity estimation method, we generated 50 samples for each dataset as follows: for each node $X_i$ with no parents, $X_{sij} \sim \mathcal{N}(0, 0.3)$; for other node $X_j$ with at least one parent, $X_{sij} = \sum_{X_k \in \text{Pa}(X_i)} X_{sk} \cdot B_{kij} + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, 0.3)$, $B_{kij}$ is 1 or -1 randomly; order of variables was shuffled. We got the estimated grouping matrix $\hat{W}$ after running the similarity estimation algorithm. We fixed the parameters $\lambda_1 = 1.0, \lambda_2 = 0.3 \times n$. The plot between $S(X_i, X_j)$ and $\hat{W}_{ij}$ for “Water” dataset is shown in Fig. 2. In Fig. 2, there is a nearly linear relationship between $S(X_i, X_j)$ and the value of $\hat{W}_{ij}$. Thus GSGBN can learn the parameter $\hat{W}_{ij}$ well and tends to penalize pairs of similar variables with larger value of $\hat{W}_{ij}$ and pairs of not-so-similar variables with smaller value of $\hat{W}_{ij}$. This behaviour can explain the reason why GSGBN outperforms other algorithms.

![Fig. 2. The distributions between $S(X_i, X_j)$ and $\hat{W}_{ij}$ for Water dataset](image)

B. Benchmark Datasets

We followed similar way as [18], [19] to generate $X$ and $B$: if there exists edge $X_i \rightarrow X_j$, $B_{ij} \sim \mathcal{N}(\pm 1, \frac{1}{5})$, otherwise $B_{ij} = 0$; for node $X_j$ with no parents, $X_{sij} \sim \mathcal{N}(0, 0.3)$; for node $X_j$ with at least one parent, $X_{sij} = \sum_{X_k \in \text{Pa}(X_i)} X_{sk} \cdot B_{kij} + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, 0.3)$.

50 samples for each dataset were generated; order of variables was shuffled. We tuned the parameters of other algorithms as suggested parameters in their papers and codes. We also tried several different parameters and compared with those with best performance. Parameters $\lambda_1$ and $\lambda_2$ in GSGBN were tuned by cross-validation. We compared the performance of GSGBN to L1MB and SBN in terms of the following measures: Sensitivity($\frac{TP}{TP+FN}$), Specificity($\frac{TN}{FP+TN}$), where TP, TN, FP, FN represent the numbers of True Positive, True Negative, False Positive, False Negative edges respectively. The results for benchmark datasets are shown in Table I. Note that at last we fit a simple linear regression model to remove the bias caused by the penalty term $\|B, W\|$. To remove edges with relatively small $B_{ij}$ values, a threshold of 0.3 was used to filter out those weak strength edges after the linear regression step.

For “Water” dataset, GSGBN greatly improves the performance (Sensitivity: 0.409 $\rightarrow$ 0.833, with comparable specificity) since there are many local structures similar to Fig. 1. GSGBN has much improvement in performance when there are many local structures with similar parents. Good performances with sensitivity above 0.80 in other datasets (“Factors”, “Alarm”, “Carpo”) also confirm this observation. For all eight datasets GSGBN reports much higher sensitivity than others, also it achieves slightly better specificity for most datasets. It is because GSGBN learns a nearly linear relationship between the value of $\hat{W}_{ij}$ and $S(X_i, X_j)$. This behaviour demonstrates that regularization on $S(X_i, X_j)$ can help improve the structure learning. Note that even though in some datasets (“Alarm”, “Barley” and “Carpo”) the specificity of GSGBN is slightly worse than others (with less than 1% difference), the sensitivity is much higher than other algorithms (e.g. in Alarm dataset, GSGBN and IAMB have similar specificity (0.994 and 0.996), while the sensitivities are 0.804 and 0.717). TC-bw does not report any edges for datasets Carpo and Hailfinder no matter how we tuned the parameter.

C. Results on Microbial Datasets

1) ICoMM dataset: We compared GSGBN with other algorithms on the ICoMM (http://icomm.mbl.edu) dataset. The dataset consists of 8,570,814 sequences, which are sequenced from the V6 region of the prokaryote 16S rRNA, from 487 bacteria communities (in 246 sites). Finally we extracted the OTU ((operational taxonomy unit) abundance matrix with those 271 samples. Since the value range of OTU counts varies a lot, we rescaled all values by log(1 + x) where x is the OTU count. The detailed procedure for preprocessing the data can be find in [23].

2) Experimental Results: Since microbes’ abundance was shown to be very important for studying the diversity of microbial communities [6], instead of using all 18620 OTUs, we only compared with other algorithms by building BNs for top 50, 100, 200 abundant OTUs and 199 OTUs selected by LapMOR [23]. The abundances for top 50, 100, 200 and the selected 199 OTUs are 70.4165%, 79.6556%, 88.187% and 73.0885% respectively. We used 5-fold cross-validation to
tune our parameters (randomly select 200 samples for training and validation and the rest 71 samples for prediction). At last, we used the parameter $\lambda_1 = 20$, $\lambda_2 = 5 \times 50, 5 \times 100, 5 \times 200$ for the first three datasets, and $\lambda_1 = 30$, $\lambda_2 = 5 \times 199$ for the last dataset. The RMSEs (root-mean-square errors) for the remaining 71 samples are shown in Table II. GSGBN consistently outperforms other algorithms.

### Table II

<table>
<thead>
<tr>
<th>Method</th>
<th>Top 50</th>
<th>Top 100</th>
<th>Top 200</th>
<th>Selected 199 OTUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSGBN</td>
<td>1.2382</td>
<td>1.0715</td>
<td>0.9331</td>
<td>0.8897</td>
</tr>
<tr>
<td>LIMB</td>
<td>1.3753</td>
<td>1.2044</td>
<td>1.0770</td>
<td>0.9791</td>
</tr>
<tr>
<td>SBN</td>
<td>1.4348</td>
<td>1.2599</td>
<td>1.2088</td>
<td>1.0097</td>
</tr>
<tr>
<td>SC</td>
<td>1.4251</td>
<td>1.2538</td>
<td>1.1087</td>
<td>1.0019</td>
</tr>
<tr>
<td>GS</td>
<td>1.3417</td>
<td>1.1701</td>
<td>1.0237</td>
<td>0.9351</td>
</tr>
<tr>
<td>TC-bw</td>
<td>1.3924</td>
<td>1.2480</td>
<td>1.3282</td>
<td>1.2497</td>
</tr>
<tr>
<td>IAMB</td>
<td>1.3431</td>
<td>1.1749</td>
<td>1.0257</td>
<td>0.9415</td>
</tr>
<tr>
<td>HITON</td>
<td>1.3569</td>
<td>1.1820</td>
<td>1.0414</td>
<td>0.9744</td>
</tr>
</tbody>
</table>

V. CONCLUSIONS

In this paper, we have proposed GSGBN algorithm which is the first study on the shared information among variables for learning BNs. As a future work for microbial data, since there is some prior knowledge on the microbes, we should study how to integrate this information to model our grouping matrix $W$ in GSGBN for better performance.

VI. ACKNOWLEDGMENTS

This work was supported by Hong Kong GRF HKU 7111/12E, HKU 719709E and 719611E, HKU Outstanding Research Award 2010-11, Shenzhen basic research project (NO.JCYJ20120618143038947) and NSFC(11171086).

### REFERENCES


