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. **BD**e [11] and **BG**e [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 **BD**e and **BG**e 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

L_1 regularization, but uses another penalty term in the score function that ensures the learned BN is a **DAG**. A* **SBN** [22] improves the heuristic search method of **L1MB** by A* **Lasso**.

For biological data, variables are usually related to each other, for example, for microbial interactions, different microbes from same phylum/class may have similar behaviour. It is natural to assume these variables should have similar parents in the hidden BN and this information should be able to learn from data. Unfortunately, all existing methods consider these variables independently without considering the shared information. 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 only one 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%).

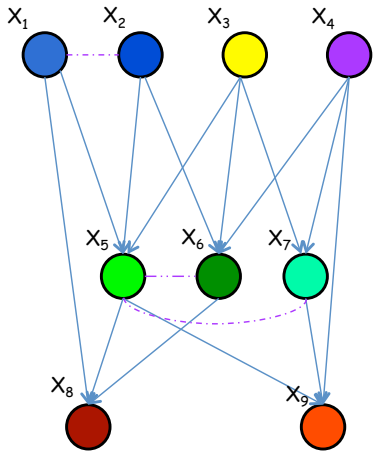

Fig. 1. An example for variable grouping

II. Background and Notations

BN is a probabilistic graphical model, representing a *joint probability distribution* of a set of variables [14], and defined over a *directed acyclic graph* (**DAG**) denoted by $G = (V,E)$ where $|V| = n$. Each node V_j in G is associated with a random continuous variable X_j. The *probability distribution* $p(X_1,...,X_n)$ associated with G can be decomposed into *products* of the distribution of *each variable conditioned on its parents*, e.g. variable X_j conditioned on its parents $P_a(X_j)$: $X_j|P_a(X_j)$. Thus

$$p(X_1,...,X_n) = \prod_{j=1}^{n} p(X_j|P_a(X_j)).$$

In this paper we only consider Gaussian BN, in which variables $X_{1,...,n}$ follow a multivariable Gaussian distribution, i.e. each variable X_j conditioned on its parents $P_a(X_j)$ follows a Gaussian distribution with mean (linear combinations of its parents X_k in $P_a(X_j)$) and variance (σ_j^2):

$$X_j|P_a(X_j) \sim \mathcal{N}(\sum_{X_k \in P_a(X_j)} X_k \mathbf{B}_{kj}, \sigma_j^2);$$

where $\mathbf{B}$ is the $n \times n$ regression coefficient matrix which encodes the edges of G, e.g. if $\mathbf{B}_{kj}$ is nonzero, there exists an edge from node V_k to V_j; otherwise, there is no edge. After

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:

$$X_j = \sum_{X_k \in P_a(X_j)} X_k \mathbf{B}_{kj} + \varepsilon_j, \varepsilon_j \sim \mathcal{N}(0, \sigma_j^2).$$

where ε_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 $\mathbf{X}_{m\times n}$ with m samples and n variables, where i^{th} row $\mathbf{X}_{i*}$ is the variable vector for the i^{th} sample, and j^{th} column $\mathbf{X}_{*j}$ is the sample vector for the j^{th} variable, we want to find the optimal **DAG** $\hat{G}$ encoded by the regression matrix $\hat{\mathbf{B}}$ with minimum $\{f(\mathbf{B}) + g(\mathbf{B})\}$ (error term $f(\mathbf{B})$ represents the error in predicting X_j from its parents $P_a(X_j)$ and regularization term $g(\mathbf{B})$ represent the penalty for some special network topologies):

$$\hat{\mathbf{B}} = \arg\min_{\mathbf{B}} \{f(\mathbf{B}) + g(\mathbf{B})\}, \quad s.t.\ \mathscr{G}(\mathbf{B}) \in \mathbf{DAG} \tag{1}$$

where

$$f(\mathbf{B}) = \sum_{j=1}^{n} (\mathbf{X}_{*j} - \mathbf{X}\mathbf{B}_{*j})^T \cdot (\mathbf{X}_{*j} - \mathbf{X}\mathbf{B}_{*j}),$$

$\mathscr{G}(\mathbf{B})$ represents the **DAG** with edges defined by nonzero elements of $\mathbf{B}$. Different algorithms have different definitions for $g(\mathbf{B})$. For **L1MB**, L_1 penalty [18], [20] is used on each column of $\mathbf{B}$ for sparsity in $\hat{G}$, i.e. $g(\mathbf{B}) = \lambda \sum_{j=1}^{n} ||\mathbf{B}_{*j}||_1$ where $||\mathbf{B}_{*j}||_1 = \sum_{i=1}^{n} |\mathbf{B}_{ij}|$. The positive regularization parameter λ 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 $\mathbf{X}_{*j}$ 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 **GSGBN** 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 $\mathbf{B}$ 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: $\mathbf{W}_{ij}\sum_{k=1}^{n} ||\mathbf{B}_{ki}|_0 - |\mathbf{B}_{kj}|_0|$ such that $\mathbf{W}_{ij}$ ($\mathbf{W}_{ij} > 0$) and the grouping matrix $\mathbf{W}$ represents the similarity of two variables X_i and X_j. The summation is for differences of parents. Mathematically, $|a|_0$ is the L_0 regularization with $|a|_0 = 1$ if $a \neq 0$ otherwise $|a|_0 = 0$. Intuitively, if variables X_i and X_j are similar, $\mathbf{W}_{ij}$ 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, $\mathbf{W}_{ij}$ 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 $\mathbf{W}_{ij}\sum_{k=1}^{n} ||\mathbf{B}_{ki}| - |\mathbf{B}_{kj}||$ which is an approximation to L_0 ($\mathbf{W}_{ij}\sum_{k=1}^{n} ||\mathbf{B}_{ki}|_0 - |\mathbf{B}_{kj}|_0|$).

C. Problem Modeling

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

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

where C is the normalization constant,

$$L(\mathbf{B}) = \prod_{i \neq j} Lap(\mathbf{B}_{ij}),\ F(\mathbf{B}) = \prod_{i \neq j} Grp(\mathbf{B}, i, j);$$
$$Lap(\mathbf{B}_{ij}) = \frac{\lambda_1}{2} e^{-\lambda_1 |\mathbf{B}_{ij}|}, Grp(\mathbf{B}, i, j) = \frac{\lambda_2 \mathbf{W}_{ij}}{2} e^{-\lambda_2 \mathbf{W}_{ij} \sum_k ||\mathbf{B}_{ki}| - |\mathbf{B}_{kj}||}; \tag{3}$$

$$\sum_{i,j} \mathbf{W}_{ij} = 1 \ and\ \mathbf{W}_{ii} = 0,\ \forall i;$$
$$\mathbf{W}_{ij} = \mathbf{W}_{ji}, 0 < \mathbf{W}_{ij} < 1, \forall i \neq j. \tag{4}$$

In (2), $L(\mathbf{B})$ is used for modeling **Sparse Network** and $\mathbf{B}_{ij}$ follows an independent Laplacian distribution $Lap(\mathbf{B}_{ij})$, i.e. $\mathbf{B}_{ij} \sim Laplace(0, \lambda_1^{-1})$. The Laplacian distribution assumption of $\mathbf{B}$ is to impose L_1 penalty on $\mathbf{B}$ to make $\mathbf{B}$ sparse [20]. The probability distribution function for a variable x that follows a $Laplace(0, \lambda^{-1})$ is $p(x) = \frac{\lambda}{2} e^{-\lambda|x|}$.

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 $\mathbf{W}$ where $\mathbf{W}_{ij}$ is the parameter for the Laplacian distribution of $S(X_i, , X_j)$, i.e.

$\sum_k ||\mathbf{B}_{ki}|-|\mathbf{B}_{kj}|| \sim Laplace(0,(\lambda_2\mathbf{W}_{ij})^{-1})$. If $\mathbf{W}_{ij}$ is large, the $S(X_i,X_j)$ is small which we say X_i and X_j are similar. However, small $\mathbf{W}_{ij}$ indicates $S(X_i,X_j)$ is large, in this case, X_i and X_j are very different. λ_1 and λ_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 $P(\mathbf{X}|\mathbf{B})\cdot P(\mathbf{B})$, we get the following equation:

$$\hat{\mathbf{B}},\hat{\mathbf{W}} = \arg\min_{\mathbf{B},\mathbf{W}} \{f(\mathbf{B})+g(\mathbf{B},\mathbf{W})\}, \quad s.t.\ \mathscr{G}(\mathbf{B})\in \mathbf{DAG}. \quad (5)$$

where

$$f(\mathbf{B}) = \sum_{j=1}^{n} (\mathbf{X}_{*j}-\mathbf{X}\mathbf{B}_{*j})^T\cdot(\mathbf{X}_{*j}-\mathbf{X}\mathbf{B}_{*j});$$

$$g(\mathbf{B},\mathbf{W}) = \lambda_1\sum_{j=1}^{n}||\mathbf{B}_{*j}||_1+\lambda_2\sum_{i\neq j}\mathbf{W}_{ij}\sum_k||\mathbf{B}_{ki}|-|\mathbf{B}_{kj}||-\sum_{i\neq j}\log\mathbf{W}_{ij};$$

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

D. Parameter Estimation

We use a two-stage approach to solve (5). At the first stage, we learn grouping matrix $\hat{\mathbf{W}}$ without the **DAG** constraint; at the second stage, we heuristically search the **DAG** and $\mathbf{B}$ that optimize (5) with learned $\hat{\mathbf{W}}$ 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 $\mathbf{W}$, find and update the optimal $\mathbf{B}$; with fixed $\mathbf{B}$, find and update optimal $\mathbf{W}$. Repeat the above procedure until convergence or reaching the maximum number of iterations.

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

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

$$\lambda_2\sum_{j\neq i,j=1}^{n}\mathbf{W}_{ij}\leq\lambda_1,\forall i=1,...,n. \quad (6)$$

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

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

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

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

- With fixed $\mathbf{W}$, we find the optimal $\mathbf{B}$ by coordinate descent algorithm [8]. Since we cannot separate the regularization term $g(\mathbf{B})$, i.e. $g(\mathbf{B})=\sum_{ij}g_{ij}(\mathbf{B}_{ij})$, coordinate descent algorithm may not guarantee to find the global optimal $\mathbf{B}$ [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 $\mathbf{B}$. Coordinate descent always reports the same results as ADMM by comparison. At each step of coordinate descent, obtaining

$$\hat{\mathbf{B}}_{ij} = \arg\min_{\mathbf{B}_{ij}} \{f(\mathbf{B})+g(\mathbf{B},\mathbf{W})\}$$

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

$$\hat{t} = \arg\min_t\ a_0(t-b_0)^2+\sum_{i=1}^{l}a_i|t-b_i| \quad s.t.\ a_i>0,\ b_i\geq 0,\ \forall i=0,1,...,l-1,l. \quad (7)$$

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

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

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

Algorithm 1: Find optimal $\mathbf{W}$ with fixed $\mathbf{B}$

Input:
$\mathbf{B}$, $\lambda_1 and \lambda_2$;
γ: learning rate for gradient descent algorithm;
ε: convergence threshold.

Output:
$\mathbf{W}$: an $n\times n$ variable grouping matrix.

Set $\mathbf{W}_{ij}^{(0)}=\frac{1}{n\cdot(n-1)}$, $\forall i\neq j$ and $\mathbf{W}_{ii}^{(0)}$=0, $\forall i$;
$t=0$;
repeat
  $\mathbf{W}^*=\mathbf{W}^{(t)}-\gamma\nabla g(\mathbf{B},\mathbf{W}^{(t)})$ ($\nabla g(\mathbf{B},\mathbf{W}^{(t)})$ = $[\frac{\partial g(\mathbf{B},\mathbf{W}^{(t)})}{\partial \mathbf{W}_{ij}^{(t)}}]$);
  $\mathbf{W}^{(t+1)}$= Project $\mathbf{W}^*$ to the hyperplane of (4) $\cap$ (6);
  $t=t+1$;
until $|W^{(t)}-W^{(t-1)}|<\varepsilon$;
$\mathbf{W}=\mathbf{W}^t$;

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 time we only consider swapping adjacent variables:

$$(...,X_{i_j},X_{i_{j+1}},...)\rightarrow(...,X_{i_{j+1}},X_{i_j},...).$$

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/ch11y/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_j with no parents, $\mathbf{X}_{*j} \sim \mathcal{N}(0, 0.3)$; for other node X_j with at least one parent, $\mathbf{X}_{*j} = \sum_{X_k \in Pa(X_j)} \mathbf{X}_{*k} \cdot \mathbf{B}_{kj} + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, 0.3)$, $\mathbf{B}_{kj}$ is 1 or -1 randomly; order of variables was shuffled. We got the estimated grouping matrix $\hat{\mathbf{W}}$ 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 $\hat{\mathbf{W}}_{ij}$ 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 $\hat{\mathbf{W}}_{ij}$. Thus **GSGBN** can learn the parameter $\hat{\mathbf{W}}_{ij}$ well and tends to penalize pairs of similar variables with larger value of $\hat{\mathbf{W}}_{ij}$ and pairs of not-so-similar variables with smaller value of $\hat{\mathbf{W}}_{ij}$. This behaviour can explain the reason why **GSGBN** outperforms other algorithms.

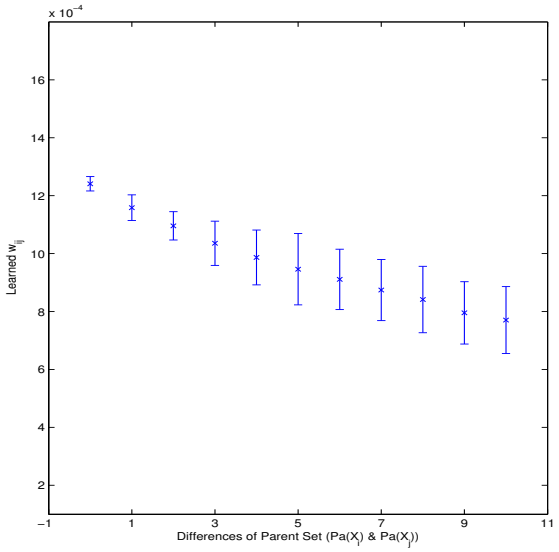

Fig. 2. The distributions between $S(X_i, X_j)$ and $\hat{\mathbf{W}}_{ij}$ for Water dataset

B. Benchmark Datasets

We followed similar way as [18], [19] to generate $\mathbf{X}$ and $\mathbf{B}$: if there exists edge $X_i \to X_j$, $\mathbf{B}_{ij} \sim \mathcal{N}(\pm 1, \frac{1}{16})$, otherwise $\mathbf{B}_{ij} = 0$; for node X_j with no parents, $\mathbf{X}_{*j} \sim \mathcal{N}(0, 0.3)$; for node X_j with at least one parent,

$$\mathbf{X}_{*j} = \sum_{X_k \in Pa(X_j)} \mathbf{X}_{*k} \cdot \mathbf{B}_{kj} + \varepsilon, \ \varepsilon \sim \mathcal{N}(0, 0.3);$$

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 λ_1 and λ_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($\frac{\mathbf{TP}}{\mathbf{TP}+\mathbf{FP}}$), Specificity($\frac{\mathbf{TN}}{\mathbf{FP}+\mathbf{TN}}$), 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 $g(\mathbf{B}, \mathbf{W})$. To remove edges with relatively small $\mathbf{B}_{ij}$ 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 $\to$ 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 $\mathbf{W}_{ij}$ 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

TABLE I
RESULTS ON EIGHT BENCHMARK DATASETS

Dataset	Measures	**GSGBN**	L1MB	SBN	SC	GS	TC-bw	IAMB	HITON
Factors	Sen./Spe.	**0.897/0.991**	0.338/0.944	0.294/0.950	0.397/0.977	0.735/0.988	0.706/0.977	0.779/0.985	0.338/ 0.938
Alarm	Sen./Spe.	**0.804**/0.994	0.304/0.986	0.152/0.995	0.435/0.991	0.609/0.990	0.261/0.989	0.717/**0.996**	0.152/0.978
Mildew	Sen./Spe.	**0.609/0.981**	0.544/0.977	0.217/0.979	0.217/0.868	0.544/0.969	0.370/0.970	0.500/0.967	0.109/0.953
Insurance	Sen./Spe.	**0.716/0.984**	0.423/0.971	0.212/0.978	0.346/0.960	0.442/0.971	0.289/0.972	0.615/0.978	0.197/0.908
Barley	Sen./Spe.	**0.655**/0.988	0.310/0.979	0.167/0.986	0.345/0.985	0.595/0.987	0.083/**0.993**	0.488/0.982	0.119/0.958
Carpo	Sen./Spe.	**0.824/0.996**	0.338/0.989	0.230/0.997	0.595/**0.996**	0.703/0.994	-/-	0.716/0.995	0.176/0.992
HailFinder	Sen./Spe.	**0.667/0.992**	0.318/0.981	0.258/0.989	0.439/0.989	0.591/0.987	-/-	0.561/0.985	0.061/0.975
Water	Sen./Spe.	**0.833/0.986**	0.288/0.950	0.152/0.980	0.409/0.981	0.379/0.952	0.273/0.980	0.349/0.949	0.197/0.908

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
RMSE RESULTS ON TOP 50, 100, 200 ABUNDANT OTUS AND SELECTED 199 OTUS

Method	Top 50	Top 100	Top 200	Selected 199
GSGBN	**1.2382**	**1.0715**	**0.9331**	**0.8897**
L1MB	1.3753	1.2044	1.0770	0.9791
SBN	1.4348	1.2959	1.2068	1.0097
SC	1.4251	1.2538	1.1007	1.0019
GS	1.3417	1.1701	1.0237	0.9515
TC-bw	1.3924	1.2480	1.5282	1.2497
IAMB	1.3431	1.1749	1.0257	0.9415
HITON	1.3569	1.1820	1.0414	0.9744

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 Researcher Award 2010-11, Shenzhen basic research project (NO.JCYJ20120618143038947) and NSFC(11171086).

REFERENCES

[1] C. F. Aliferis, I. Tsamardinos, and A. Statnikov. Hiton: a novel markov blanket algorithm for optimal variable selection. In *AMIA Annual Symposium Proceedings*, volume 2003, page 21. American Medical Informatics Association, 2003.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[3] D. M. Chickering. Learning bayesian networks is NP-Complete. In *Learning From Data*, pages 121–130. Springer, 1996.

[4] D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of bayesian networks is NP-Hard. *JMLR*, 5:1287–1330, 2004.

[5] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, pages 272–279. ACM, 2008.

[6] N. Fierer and R. B. Jackson. The diversity and biogeography of soil bacterial communities. *PNAS*, 103(3):626–631, 2006.

[7] J. Friedman, T. Hastie, H. Höfling, R. Tibshirani, et al. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[8] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

[9] N. Friedman, I. Nachman, and D. Peér. Learning bayesian network structure from massive datasets: the sparse candidate algorithm. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 206–215. Morgan Kaufmann Publishers Inc., 1999.

[10] D. Geiger and D. Heckerman. Learning gaussian networks. In *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*, pages 235–243. Morgan Kaufmann Publishers Inc., 1994.

[11] D. Heckerman. *A tutorial on learning with Bayesian networks*. Springer, 1998.

[12] S. Huang, A. Fleisher, K. Chen, J. Li, J. Ye, T. Wu, E. Reiman, et al. A sparse structure learning algorithm for gaussian bayesian network identification from high-dimensional data. *PAMI*, 35(6):1328–1342, 2013.

[13] M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *The Journal of Machine Learning Research*, 8:613–636, 2007.

[14] D. Kollar and N. Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

[15] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, pages 801–808, 2006.

[16] D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In S. Solla, T. Leen, and K.-R. Müller, editors, *Proceedings of Conference on Neural Information Processing Systems*. MIT Press, 1999.

[17] J.-P. Pellet and A. Elisseeff. Using markov blankets for causal structure learning. *JMLR*, 9:1295–1342, 2008.

[18] M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using l1-regularization paths. In *AAAI*, volume 7, pages 1278–1283, 2007.

[19] M. Teyssier. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *In UAI*, 2005.

[20] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[21] I. Tsamardinos and C. F. Aliferis. Towards principled feature selection: Relevancy, filters and wrappers. In *Proceedings of the ninth international workshop on Artificial Intelligence and Statistics*. Morgan Kaufmann Publishers: Key West, FL, USA, 2003.

[22] J. Xiang and S. Kim. A* lasso for learning a sparse bayesian network structure for continuous variables. In *Advances in Neural Information Processing Systems*, pages 2418–2426, 2013.

[23] J. Yang, H. Leung, S. Yiu, Y. Cai, and F. Y. Chin. Intra- and inter-sparse multiple output regression with application on environmental microbial community study. In *BIBM 2013*, pages 404–409. IEEE, 2013.