A Regression Tree-Based Gibbs Sampler to Learn the Regulation Programs in a Transcription Regulatory Module Network

Jianlong Qi, Tom Michoel and Gregory Butler

Abstract— **Many algorithms have been proposed to learn transcription regulatory networks from gene expression data. Bayesian networks have obtained promising results, in particular, the module network method. The genes in a module share a regulation program (regression tree), consisting of a set of parents and conditional probability distributions. Hence, the method significantly decreases the search space of models and consequently avoids overfitting. The regulation program of a module is normally learned by a deterministic search algorithm, which performs a series of greedy operations to maximize the Bayesian score. The major shortcoming of the deterministic search algorithm is that its result may only represent one of several possible regulation programs. In order to account for the model uncertainty, we propose a regression tree-based Gibbs sampling algorithm for learning regulation programs in module networks. The novelty of this work is that a set of tree operations is defined for generating new regression trees from a given tree and we show that the set of tree operations is sufficient to generate a well mixing Gibbs sampler even in large data sets. The effectiveness of our algorithm is demonstrated by the experiments in synthetic data and real biological data.**

I. Introduction

Cells have a complex mechanism that controls the expression of genes so that they are able to express varied combinations of genes in response to environmental changes or genetic perturbations. A major part of this control is fulfilled by transcription factors (TFs), which are able to bind to upstream of genes and then influence their expression levels. The transcriptional regulatory relationship between genes and their transcription factors can be represented by a network, called a transcription regulatory network (TRN).

Many algorithms have been proposed to learn TRNs from gene expression data, including Bayesian networks [1], information-theoretic approaches [2], and clustering algorithms [3]. In particular, Bayesian networks, a type of probabilistic graphical models, have yielded promising results in many works, such as [4] and [5]. One advantage of Bayesian networks is to apply Bayesian probability theory to deal with uncertainty in the data. Moreover, the method uses directed acyclic graphs to represent TRNs, which facilitates the interpretation of results, i.e., that each gene is regulated by its parents.

Jianlong Qi and Gregory Butler are with the Department of Computer Science, Concordia University, Montreal, Canada (email:{q_jianlo,gregb}@encs.concordia.ca).

Tom Michoel is with the Department of Plant Systems Biology, VIB, Department of Plant Biotechnology and Genetics, Ghent University, Technologiepark 927, B-9052 Ghent, Belgium (email: tom.michoel@psb.vib-ugent.be).

This work was partially supported by NSERC grant.

Despite the success of Bayesian networks in learning TRNs, this method has a major shortcoming. A typical gene expression data set describes thousands of genes, but at most consists of a few hundred instances (experimental conditions). In Bayesian networks, each gene is associated with its individual regulation program, i.e., its own set of parents and conditional probability distributions, so the number of structural features and distribution parameters to be learned is enormous relative to the amount of available data [6]. Hence, learned models may overfit the data and consequently do not represent real regulatory relationships.

The authors in [7] introduced the module network method, which is a special type of Bayesian network, to remedy the overfitting. In the method, a regulatory module is a set of genes that show similar expression profiles and are regulated by a shared set of regulators. Since genes in the same module share parents, the number of parameters to be learned is significantly decreased. The method has yielded promising results in several complex eukaryotic systems such as yeast [8] and mouse [9].

To learn module network models, Segal $et\ al.$ [8] applied the expectation maximization (EM) algorithm [10]. Using the Bayesian score [11] to evaluate a model's fit to the data, the learning algorithm alternates between learning a regulation program for each module (so called M-steps) and assigning each gene to the module whose regulation program best predicts its behavior (so called E-steps). One limitation of the EM algorithm is that it selects a single module network model that represents a local maximum in the posterior distribution of models given the data. One possible solution for solving the limitation is to apply sampling-based methods to sample models from the posterior distribution. However, sampling-based methods may show extremely slow convergence rates when learning module networks, because they have to shift between E-steps and M-steps, which leads to a huge search space of models.

Michoel $et\ al.$ [12] introduced a novel two-step method for learning module networks, which separates clustering genes into modules and learning a regulation program for each module. This design makes it feasible to apply sampling-based algorithms to learn module networks, because the search space of models significantly decreases. In the first stage, a Gibbs sampling algorithm [13] is used to cluster genes into modules based on their expression profiles. In the second stage, an ensemble method is applied to calculate the regulator score for assigning a regulator to a module. The regulator assignment is performed by learning a fuzzy regression tree model on a set of leaf node condition clusters sampled by the Gibbs sampler

978-1-4244-6766-2/10/ $26.00 © IEEE

clustering algorithm [12].

In this paper, we focus on the second step of the method proposed in [12]. The main purpose of this work is to demonstrate that it is feasible to apply a regression tree-based Gibbs sampling algorithm to learn regulation programs in module networks, with similar performance as the fuzzy regression tree model of [12]. Given a module, we use a Gibbs sampler to sample regulation programs, i.e., regression trees, from the posterior distribution of regulation programs given the data. A set of tree operations is defined for generating new regression trees from a given tree. We show that the set of tree operations is sufficient to generate a well-mixing Gibbs sampler even in large data sets. Based on the frequency with which a regulator appears in the sampled regulation programs and the significance that the regulator shows in the sampled regulation programs, we provide the confidence estimate for the regulatory relationship between the regulator and the module.

The remainder of this paper is organized as follows. In Section II, we describe how to apply Gibbs sampling to sample regression trees in module networks. In Section III, we present the experimental results in synthetic and real biological data. Section IV concludes this paper by summarizing the main results.

II. Applying Gibbs Sampling to Learn a Regulation Program in Module Networks

A. The Regulation Program of a Module and its Bayesian Score

The regulation program of a given module can be represented by a regression tree, which consists of two types of nodes: internal nodes and leaf nodes [7]. Each internal node, which has a regulator and a splitting value, corresponds to a test of whether the expression value of the regulator for an experimental condition is greater than the splitting value, and has two child nodes: the right child node is chosen when the answer to the test is true; the left child is chosen when not. Each leaf node represents a set of experimental conditions where the behaviors (upregulation, no change, or downregulation) of regulators match the context specified by the tests on the path to the leaf node. In addition, each leaf node is associated with a normal distribution to describe the expression levels of the module's genes in the experimental conditions associated with the leaf node.

The task of learning regulation programs for a module is to find the program that best explains the change of gene expression levels in the module. A regulation program's fit to a module can be evaluated by the Bayesian score [11] of its regression tree, R, which is defined as:

$$S(R) = \sum_{l \in L} S_l, \tag{1}$$

where L represents the set of leaf nodes in R; S_l is the Bayesian score of leaf node l, which is calculated as:

$$S_l = \log \int \int p(\mu, \tau) \prod_{m \in \varepsilon_l} \prod_{i \in A} p(x_{i,m} | \mu, \tau) \, d\mu d\tau, \tag{2}$$

where ε_l and A denote the set of experimental conditions in leaf node l and the set of genes assigned to the module, respectively; $x_{i,m}$ represents the expression level of gene i in experimental condition m; $p(x|\mu, \tau)$ is a normal distribution with mean μ and precision τ; $p(\mu, \tau) = p(\mu|\tau)p(\tau)$ is a normal-gamma prior distribution over μ and τ with

$$p(\mu|\tau) = \left(\frac{\lambda_0 \tau}{2\pi}\right)^{1/2} e^{-\frac{\lambda_0 \tau}{2}(\mu - \mu_0)^2},$$

$$p(\tau) = \frac{{\beta_0}^{\alpha_0}}{\Gamma(\alpha_0)} \tau^{\alpha_0 - 1} e^{-\beta_0 \tau},$$

$\alpha_0, \beta_0, \lambda_0 > 0$ and $-\infty < \mu_0 < \infty$. In this paper, we use the values $\alpha_0, \beta_0, \lambda_0 = 0.1$ and $\mu_0 = 0.0$.

The double integral in Equation (2) can be solved explicitly by

$$T_l^{(n)} = \sum_{i \in A, m \in \epsilon_l} x_{i,m}^n (n = 0, 1, 2.),$$

and the result is

$$\begin{aligned} S_l = &- \frac{1}{2} T_l^{(0)} \log(2\pi) + \frac{1}{2} \log\left(\frac{\lambda_0}{\lambda_0 + T_l^{(0)}}\right) \\ &- \log\Gamma(\alpha_0) + \log\Gamma\left(\alpha_0 + \frac{1}{2} T_l^{(0)}\right) \\ &+ \alpha_0 \log\beta_0 - \left(\alpha_0 + \frac{1}{2} T_l^{(0)}\right) \log\beta_1 \end{aligned} \tag{3}$$

with

$$\beta_1 = \beta_0 + \frac{1}{2}\left[T_l^{(2)} - \frac{\left(T_l^{(1)}\right)^2}{T_l^{(0)}}\right] + \frac{\lambda_0 \left(T_l^{(1)} - \mu_0 T_l^{(0)}\right)^2}{2\left(\lambda_0 + T_l^{(0)}\right) T_l^{(0)}}.$$

Hence, it is straightforward to compute the Bayesian score of a regulation program for a module.

B. Sampling Regression Trees by a Gibbs Sampler

In [8], the authors used a deterministic search algorithm to learn the regression tree (regulation program) of a module. That is, given a list of regulators, starting from a regression tree with a single leaf node consisting of all experimental conditions, the algorithm performs a series of operations that split a leaf node into two leaf nodes using a regulator and a splitting value as the test. The selection of leaf nodes, regulators and splitting values is done to maximize the increase of the Bayesian score. In each splitting operation, the selected leaf node becomes an internal node as the parent of the new leaf nodes and is associated with the selected regulator and splitting value. This process stops when no splitting operation increases the Bayesian score. The regulators appearing in the learned regression tree are considered as the regulators of the module. The major shortcoming of the deterministic algorithm is that its result may only represent one of several possible models.

In order to account for the model uncertainty in the deterministic search algorithm, we may use Bayesian model averaging [14], which is able to calculate the strength of

a regulator over all possible regression trees. However, in general it is not feasible to enumerate all regression trees due to the huge number of possible trees. Hence, we use a Gibbs sampling algorithm to sample regression trees from the posterior distribution of regression trees given the data, where the log-likelihood of a regression tree R under the posterior distribution is given by the Bayesian score (eq. (1)).

Following the standard Gibbs sampling framework [15], given the current regression tree R_t, we sample the next regression tree R_{t+1} from the neighborhood of R_t ($\mathrm{nbd}(R_t)$), which consists of R_t itself and the regression trees generated by modifying R_t with one of following operations:

- splitting one leaf node into two leaf nodes by a regulator and a splitting value, and converting the split leaf node into an internal node associated with the selected regulator and splitting value;
- trimming two leaf nodes connecting to a same internal node and converting the internal node into a leaf node, i.e., the reverse of the splitting operation;
- replacing the regulator and splitting value in one internal node with another regulator and splitting value.

The probability or transition rate Q_R that a regression tree $R \in \mathrm{nbd}(R_t)$ is selected as R_{t+1} is proportional to the exponential of the Bayesian score difference, i.e.,

$$Q_R = \frac{e^{S(R)-S(R_t)}}{\sum_{R' \in \mathrm{nbd}(R_t)} e^{S(R)-S(R_t)}}.$$

Starting from a randomly generated regression tree, the Gibbs sampling procedure simulates a Markov chain with the posterior distribution as its equilibrium distribution. In other words, after a burn-in period, the probability to sample a particular regression tree R, is proportional to $e^{S(R)}$.

Initial random regression trees for the Gibbs sampling procedure are generated by the following method. For a data set consisting of N experimental conditions, we generate an arbitrary number K, $1 \leq K \leq \sqrt{N}$. Then, starting from a tree with only one leaf node consisting of all experimental conditions, we randomly split a leaf node into two leaf nodes using a randomly selected regulator and splitting value. This process stops when the regression tree has more than K leaf nodes and this tree is used as the starting point for the Gibbs sampling procedure.

In large data sets, the splitting operation is very time-consuming, because we need to enumerate all possible combinations of a regulator and a splitting value in each leaf node. In this case, we can restrict the maximum number of leaf nodes in regression trees to decrease the sample space. In addition, the replacing operation may also cause the sampling process to be very slow when there are many regulators or experimental conditions. In this situation, we may discretize the expression values of regulators to decrease the number of possible splitting values of each regulator.

C. The Regulatory Score of a Regulator

At first view, one approach to identify regulators of a module is to select the regulators that frequently appear in the regression trees sampled by the Gibbs sampling procedure. However, the approach does not consider the significance of regulators in regression trees, e.g., how many conditions they regulate in regression trees.

Hence, we introduce a regulatory score $f(y, R)$, for a regulator y in a regression tree R, representing the significance that y shows in R. This score is defined as:

$$f(y, R) = e^{\frac{\sum_{l \in L_y} S_l}{|A| \cdot C_y}} \cdot \frac{C_y}{C}, \tag{4}$$

where S_l is defined as Equation (2); L_y denotes the set of leaf nodes which can be reached from the internal node where y is assigned to in R; $|A|$ denotes the number of genes assigned to the module; C and C_y denote the total number of experimental conditions in the module and the number of experimental conditions assigned to the leaf nodes in L_y, respectively. Essentially, Equation (4) represents the product of the geometric average of the prediction probability associated with the leaf nodes under y and the weight factor, $\frac{C_y}{C}$, which suggests that regulators regulating more experimental conditions are more significant in a regression tree.

D. The Expected Value of the Regulatory Score of a Regulator

Given a sequence of regression trees $\{R_t, t = 1, 2, ..., N\}$ generated by the Gibbs sampling algorithm, the expected value of the regulatory score of a regulator y with respect to the posterior distribution can be approximated as:

$$E(S(y)) \approx \frac{1}{N} \sum_{t=1}^{N} f(y, R_t). \tag{5}$$

Since $E(S(y))$ takes into account the regulatory score of y shown in each regression tree and the posterior probability of each regression tree, it can be used to evaluate the likelihood that y regulates the module. Hence, given a set of regulators, we can rank the regulators by $E(S(y))$ and then select the top-ranked regulators as the most significant regulators.

In this paper, we use the method proposed in [13] to test convergence of the Gibbs sampling algorithm. That is, given a set of regulators $\{y_i, i = 1, ..., m\}$, we run two independent Gibbs samplers. Let a_i and b_i denote $E(S(y_i))$ produced by the first Gibbs sampler and second Gibbs sampler, respectively. Then, the correlation measure between the two samplers is defined as:

$$\rho = \frac{|\sum_{i=1}^{m} a_i b_i|}{\sqrt{(\sum_{i=1}^{m} a_i^2)(\sum_{i=1}^{m} b_i^2)}}. \tag{6}$$

If the correlation, ρ, between two Gibbs samplers is 1, then they reach full convergence.

III. Experimental results and discussion

In this section, we present the results produced by the regression tree-based Gibbs sampler for synthetic data and real biological data.

A. Synthetic Data

We require that the number of genes in a simulated network should be close to that in a real biological module and the regulatory relationships in a simulated network should not be obvious. Hence, we used SynTReN [16] to generate simulated data sets for gene networks with 30 genes of which 10-15 act as regulators. The topology of the networks was sub-sampled from the transcriptional network of Saccharomyces cerevisiae. All parameters of SynTReN were set to default values. We generated 12 simulated data sets with the above configurations and each data set consisted of 60 microarray samples for 20 experimental conditions.

In each synthetic data set, we ran 10 independent Gibbs samplers using the list of true regulators as potential regulators. Staring from a randomly generated regression tree, each Gibbs sampler used 50 iterations for burn in steps, had a sampling step of 10 iterations, and totally consisted of 100 iterations. This configuration is applied in all Gibbs samplers in this paper. Then, we calculated the expected value of the regulatory score of each regulator [see Equation (5)] based on the regression trees sampled by the 10 Gibbs samplers. Lastly, we ranked the regulators according to their expected values. To investigate the convergence property of this sampling procedure, we calculated the correlation measure between the expected values from two sets of 10 Gibbs samplers [see Equation (6)], which is 0.99, a value indicating that the Gibbs sampling procedure reached convergence.

In this subsection, we compare the ranks produced by the regression tree-based Gibbs sampling with those produced by the deterministic algorithm [8] and LeMoNe [12]. In a good rank of regulators, the more genes a regulator regulates, the higher the rank of this regulator. We use the F-measure [6] to evaluate the ranks produced by different algorithms. Given a rank of regulators in a synthetic gene network, starting from an empty predicted regulator set, regulators are sequentially added into the predicted regulator set according to their order in the rank. As one regulator is included into the predicted regulator set, the regulator is predicted to regulate all genes in the network. Then, based on the true gene network and predicted regulatory relationships, we calculate the corresponding true positive (tp), false positive (fp), false negative (fn) and F-measure for the current predicted regulator set. The F-measure is defined as:

$$F = \frac{2PR}{P+R},$$

where

$$P = \frac{tp}{tp+fp} \quad and \quad R = \frac{tp}{tp+fn}.$$

Note that regulatory relationships via intermediate regulators are also considered as true positives.

1) Regression Tree-Based Gibbs Sampling versus Deterministic Algorithm: The deterministic algorithm (see Section II-B) can rank regulators based on the order that they are selected to split leaf nodes. Figure 1 shows the comparison of the F-measure between the regression tree-based Gibbs sampling and the deterministic algorithm. Generally, the regression tree-based Gibbs sampling gives better F-measure. Furthermore, we compared the ranks produced by the two algorithms in each data set (see Table I). In 9 out of 12 data sets, they gave comparable results, but the regression tree-based Gibbs sampling algorithm generated the better results in the remaining 3 data sets.

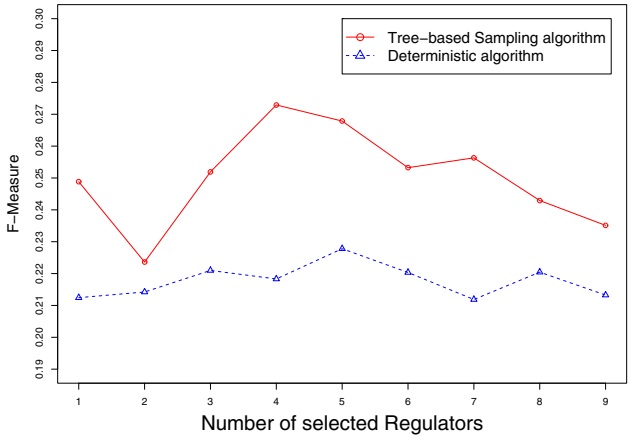


Fig. 1. Plot of the average of the F-measure produced by the regression tree-based Gibbs sampling and the deterministic algorithm in 12 synthetic data sets.

Figure 2 shows the regulatory network in the data set 4 where the tree-based sampling algorithm outperformed the deterministic algorithm. In the data set, the tree-based sampling algorithm ranked MBP1_SWI6, SWI4_SWI6, and SPT16 as the top 3 regulators and they regulate 10, 3 and 12 genes, respectively, but the deterministic algorithm selected SWI4_SWI6, ACE2 and TUP1 and they all only regulate 3 genes. Since MBP1_SWI6 and SPT16 are the most important regulators, the tree-based sampling algorithm detects the true network structure in the data set. Generally, we observed that when the structures of networks are relatively complex, such as having many intermediate regulators, the regression tree-based Gibbs sampling algorithm is more likely to outperform the deterministic algorithm.

2) Regression Tree-Based Gibbs Sampling Algorithm versus LeMoNe: LeMoNe [12] is an ensemble method, which samples the clustering of experimental conditions and learns fuzzy decision trees as regulation programs. Given a module and a list of regulators, LeMoNe is also able to rank the regulators according to the probabilities that they regulate the module.

In most data sets the regression tree-based Gibbs sampling algorithm and LeMoNe achieved comparable results (see Table I). In the data set 4, the tree-based sampling algorithm outperformed LeMoNe and LeMone's result is similar to that of the deterministic algorithm. However, LeMoNe gave the better result in the data set 12. The regulatory network of this data set is shown in Figure 3. The tree-based sampling algorithm ranked ALPHA1 and REB1 as the top two regulators, but they both only regulated two genes. LeMoNe assigned

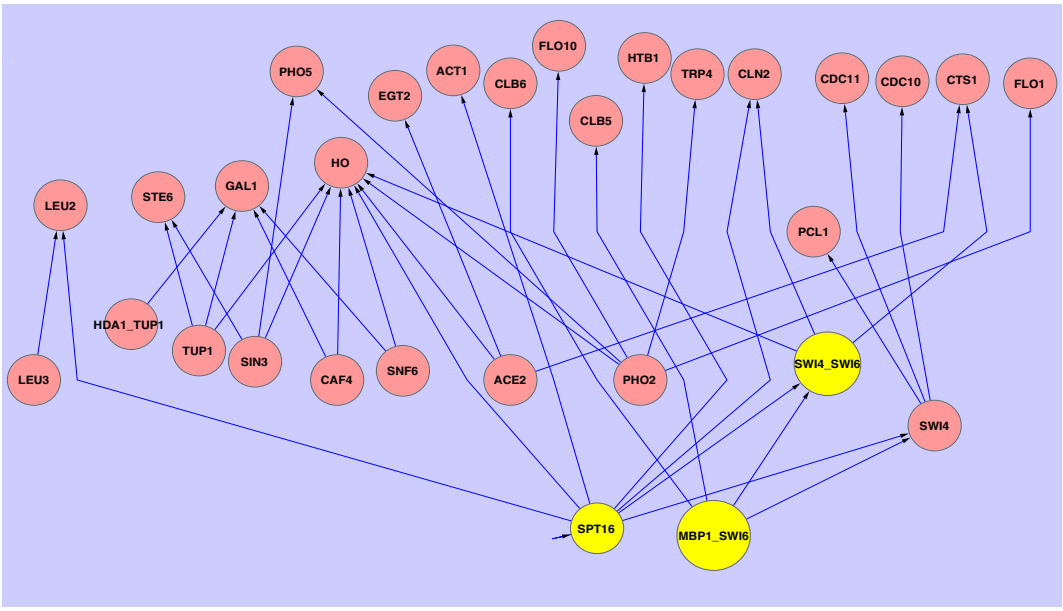


Fig. 2. Regulatory network in the data set 4.

TABLE I

F-MEASURE OF THE REGRESSION TREE-BASED SAMPLING ALGORITHM, DETERMINISTIC ALGORITHM AND LEMONE IN SYNTHETIC DATA SETS

Algorithms	#Regulators Selected	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6	Dataset 7	Dataset 8	Dataset 9	Dataset 10	Dataset 11	Dataset 12
Regression	1	0.33	0.38	0.41	**0.26**	0.17	0.14	0.15	0.17	**0.25**	**0.26**	0.41	0.05
Tree-based	2	0.27	0.31	0.33	**0.24**	0.14	0.17	0.12	0.14	**0.21**	**0.32**	0.35	0.08
Sampling Algorithm	3	0.22	0.28	0.33	**0.36**	0.15	0.27	0.16	0.22	**0.35**	**0.28**	0.31	0.08
The Deterministic Algorithm	1	0.33	0.38	0.41	0.08	0.17	0.14	0.15	0.17	0.02	0.26	0.41	0.03
	2	0.27	0.38	0.33	0.11	0.18	0.28	0.12	0.14	0.09	0.26	0.35	0.06
	3	0.22	0.33	0.33	0.13	0.15	0.26	0.16	0.14	0.26	0.22	0.35	0.09
LeMoNe	1	0.03	0.38	0.41	0.08	0.17	0.14	0.15	0.17	0.02	0.26	0.41	**0.03**
	2	0.04	0.29	0.37	0.11	0.14	0.28	0.12	0.27	0.26	0.32	0.35	**0.19**
	3	0.21	0.24	0.32	0.12	0.15	0.33	0.13	0.22	0.35	N/A	0.52	**0.18**

A1_ALPHA2 and GAL11 as the top two regulators and they regulated one and nine genes, respectively.

B. Biological Data

In this subsection, we tested the regression tree-based Gibbs sampling algorithm on two real biological modules, module 11 and 7 learned in [13], from the yeast stress data set [17] consisting of 2355 genes and 173 experimental conditions. The data set contains a large number of regulators (466) and experimental conditions (173), so the Gibbs sampling procedure may have a slow convergence rate. Hence, we applied the method proposed in [13] to identify how many independent Gibbs sampler runs are required for reaching convergence.

We ran 150 independent Gibbs samplers[1] in module 7. Then, we calculated the expected value of the regulatory score of each regulator based on the regression trees sampled by k ($k = 1, ..., 50$) samplers. Figure 4 shows the correlation measure between the expected values of two non-overlapping sets of k ($k = 1, ..., 50$) Gibbs samplers. We observed that all samplers achieved comparable Bayesian scores, but the correlation measure between two individual Gibbs samplers is only around 0.37. This indicates that there are multiple local maxima in the posterior distribution and each sampler

[1]We used the discretized expression values of regulators in the replacing operation when constructing the neighborhood of a regression tree and set the maximum number of leaf nodes in the tree to the square root of the number of the experimental conditions in the data set, i.e., 14.

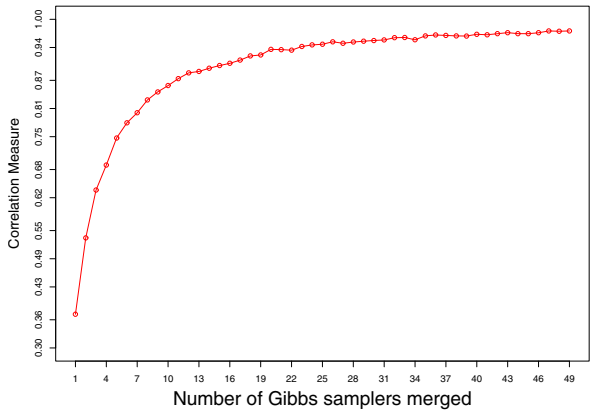


Fig. 4. Correlation measure between two sets of k Gibbs samplers ($k = 1, ..., 50$) in module 7.

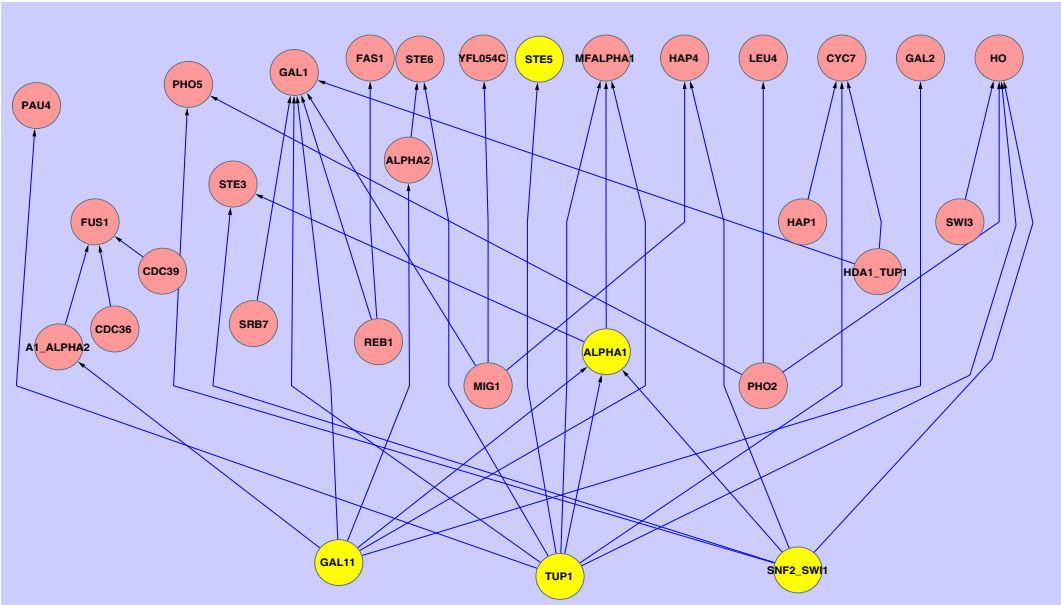


Fig. 3. Regulatory network in the data set 12.

can only visit a local maximum. However, the correlation measure between two sets of 20 Gibbs samplers is around 0.94, a value indicating that the Gibbs sampling procedure reached convergence. We observed a similar result of the convergence test in module 11. This implies that the regression tree-based Gibbs sampling has a well mixing rate even in this large data set.

Based on the above convergence tests, we ran 20 independent Gibbs samplers in module 7 and 11, respectively. Regulatory relationships recorded in the YEASTRACT [18] database were used to validate the results of the algorithm. Below is the experimental results.

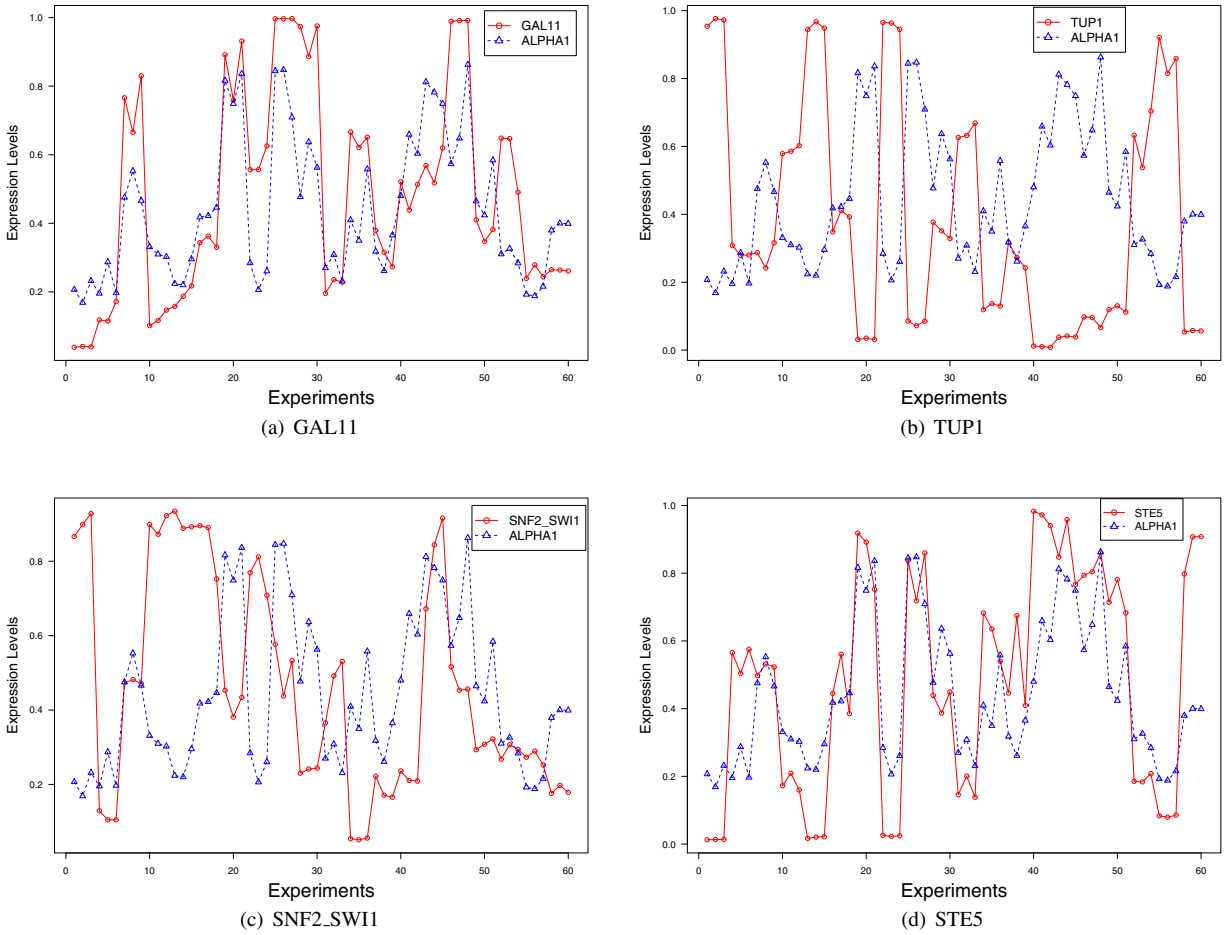
Module 11 consists of 47 genes. Most genes in the module participate in nitrogen catabolite repression or amino acid metabolism. The regression tree-based Gibbs sampling algorithm ranks UGA3, MET28 and GAT1 as the top three regulators, and they are all supported by YEASTRACT. Met28 is a member of the basic leucine zipper DNA binding factor family and encodes a transcription factor that participates in the regulation of sulfur amino acid metabolism. In the module, 8 of 47 genes are known to be regulated by Met28 in YEASTRACT. In addition, the Met28 binding motif, 5'-TCACGTG-3', is detected in upstream of 20 genes. Gat1 encodes a transcriptional activator that is involved in the regulation of genes participating in nitrogen catabolite repression. In the module, seven genes are regulated by Gat1 according to the YEASTRACT database and the Gat1 binding motif is found in upstream of 19 genes. Uga3 regulates the transcription of genes, which are required for the utilization of gamma-aminobutyrate as a nitrogen source. In the module, one gene is known to be regulated by Uga3 in YEASTRACT and the Uga3 binding motif, 5'-SGCGGNWTTT-3', is detected in upstream of four genes.

Module 7 consists of 30 genes. Most genes in the module participate in respiratory processes. Hap4, Gsm1 and Usv1 are the top three regulators predicted by the regression tree-based Gibbs sampling algorithm. Hap4 is a well known regulator of respiratory genes and 28 genes in the module are regulated by Hap4 according to the YEASTRACT database. YEASTRACT does not show that any gene in the module is regulated by Gsm1 or Usv1, but the functions of the two regulators recorded in SGD database [19] are relevant to respiratory process. GSM1 and Usv1 are suspected to regulate genes involved in energy metabolism and growth on non-fermentable carbon sources, respectively.

C. Discussion

The regression tree-based Gibbs sampling, deterministic algorithm and LeMoNe all show bad performance when detecting the regulatory network shown in Figure 3 (the data set 12) and have a common problem that they select an intermediate regulator, which is co-regulated with the module instead of regulating it, as the top regulator. In this subsection we analyze the result produced by the tree-based sampling algorithm to identify the reason.

ALPHA1 is selected as the top regulator by the regression tree-based Gibbs sampling algorithm, but it is actually an intermediate regulator regulated by GAL11, SNF2_SWI1 and TUP1, the most important regulators in the gene network. Due to the regulatory relationships, ALPHA1 co-expresses with the three regulators (see Figures 5(a) to 5(c)). Furthermore, we observed that ALPHA1 also co-expresses with genes regulated by the three regulators to some degree, because correlations between gene expression levels show transitivity. For example, Figure 5(d) shows that ALPHA1 co-expresses with STE5 that is regulated by TUP1.



(a) GAL11 (b) TUP1

(c) SNF2_SWI1 (d) STE5

Fig. 5. Correlation between the expression profiles of GAL11, TUP1, SNF2_SWI1, STE5 and ALPHA1 in the data set 12.

We consider the correlation between ALPHA1 and STE5 as a fake correlation in contrast to a true correlation between a gene and its regulator. Fake correlations cause ALPHA1 to co-express with most genes in the network, but the tree-based sampling algorithm cannot distinguish between true correlations and fake correlations, so it ranks ALPHA1 as the most significant regulator. This problem is very common. For example, in the network shown in Figure 2, SWI4_SWI6, which is regulated by SPT16 and MBPI_SWI6, is ranked as the second most significant regulator, but it only regulates 3 genes.

Solely gene expression value-based methods, including the regression tree-based Gibbs sampling, deterministic algorithm and LeMoNe, cannot distinguish between true correlations and fake correlations. One solution for the problem is to combine biological prior knowledge into the learning algorithms. The method designed in [20] can be used to model prior knowledge and add it into the Bayesian score of regression trees. In addition, another possible solution is to evaluate multiple regulators in each internal node in a regression tree, instead of only selecting a regulator and a splitting value. For example, we may build a classifier that uses several regulators as features to categorize experimental conditions into two groups in each internal node. We will further investigate the feasibility of the two solutions.

IV. Conclusion

In this paper, we extended the module network method by using a regression tree-based Gibbs sampling algorithm for learning regulation programs in a module network. We showed that in synthetic data sets the proposed sampling algorithm achieves better results than the deterministic algorithm. Moreover, most predictions made by the sampling algorithm in real biological data are supported by known regulatory relationships in biological databases. In addition, we identified a problem in solely gene expression value-based learning algorithms and we will work on the problem in our future work.

References

[1] N. Friedman, “Inferring Cellular Networks Using Probabilistic Graphical Models,” *Science*, vol. 303, no. 5659, pp. 799–805, 2004.

[2] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner, "Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles," *PLoS Biology*, vol. 5, no. 1, pp. 54–66, 2007.

[3] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 95, no. 25, pp. 14863–14868, 1998.

[4] N. Friedman, M. Linial, and I. Nachman, "Using Bayesian networks to analyze expression data," *Journal of Computational Biology*, vol. 7, no. 3-4, pp. 601–620, 2000.

[5] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller, "Rich probabilistic models for gene expression," *Bioinformatics*, vol. 17, no. suppl_1, pp. S243–252, 2001.

[6] T. Michoel, S. Maere, E. Bonnet, A. Joshi, Y. Saeys, T. Van den Bulcke, K. Van Leemput, P. van Remortel, M. Kuiper, K. Marchal, and Y. Van de Peer, "Validating module network learning algorithms using simulated data," *BMC Bioinformatics*, vol. 8, no. Suppl 2, p. S5, 2007.

[7] E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman, "Learning module networks," *Journal of Machine Learning Research*, vol. 6, pp. 557–588, 2005.

[8] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman, "Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data.," *Nature Genetics*, vol. 34, pp. 166–176, June 2003.

[9] J. Li, Z. J. Liu, Y. C. Pan, Q. Liu, X. Fu, N. G. Cooper, Y. Li, M. Qiu, and T. Shi, "Regulatory module network of basic/helix-loop-helix transcription factors in mouse brain," *Genome Biol*, vol. 8, no. 11, p. R244, 2007.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[11] D. Heckerman, *A tutorial on learning with Bayesian networks*. Cambridge, MA, USA: MIT Press, 1999.

[12] A. Joshi, R. De Smet, K. Marchal, Y. Van de Peer, and T. Michoel, "Module networks revisited: computational assessment and prioritization of model predictions," *Bioinformatics*, vol. 25, no. 4, pp. 490–496, 2009.

[13] A. Joshi, Y. Van de Peer, and T. Michoel, "Analysis of a Gibbs sampler method for model-based clustering of gene expression data," *Bioinformatics*, vol. 24, no. 2, pp. 176–183, 2008.

[14] D. Madigan, J. York, and D. Allard, "Bayesian graphical models for discrete data," *International Statistical Review*, vol. 63, no. 2, pp. 215–232, 1995.

[15] J. S. Liu, *Monte Carlo strategies in scientific computing*. Springer, 2004.

[16] T. Van den Bulcke, K. Van Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. De Moor, and K. Marchal, "Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms," *BMC Bioinformatics*, vol. 7, no. 1, p. 43, 2006.

[17] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown, "Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes," *Mol. Biol. Cell*, vol. 11, no. 12, pp. 4241–4257, 2000.

[18] P. T. Monteiro, N. D. Mendes, M. C. Teixeira, S. d'Orey, S. Tenreiro, N. P. Mira, H. Pais, A. P. Francisco, A. M. Carvalho, A. B. Lourenco, I. Sa-Correia, A. L. Oliveira, and A. T. Freitas, "YEASTRACT-DISCOVERER: new tools to improve the analysis of transcriptional regulatory associations in Saccharomyces cerevisiae," *Nucl. Acids Res.*, vol. 36, no. suppl_1, pp. D132–136, 2008.

[19] J. Cherry, C. Adler, C. Ball, S. Chervitz, S. Dwight, E. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, and D. Botstein, "SGD: Saccharomyces Genome Database," *Nucl. Acids Res.*, vol. 26, no. 1, pp. 73–79, 1998.

[20] A. V. Werhli and D. Husmeier, "Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge," *Statistical Applications in Genetics and Molecular Biology*, vol. 6, no. 1, 2007.