Hindawi Publishing Corporation
EURASIP Journal on Bioinformatics and Systems Biology
Volume 2007, Article ID 71312, 14 pages
doi:10.1155/2007/71312

Research Article

Uncovering Gene Regulatory Networks from Time-Series Microarray Data with Variational Bayesian Structural Expectation Maximization

Isabel Tienda Luna,[1] Yufei Huang,[2] Yufang Yin,[2] Diego P. Ruiz Padillo,[1] and M. Carmen Carrion Perez[1]

[1] *Department of Applied Physics, University of Granada, 18071 Granada, Spain*
[2] *Department of Electrical and Computer Engineering, University of Texas at San Antonio (UTSA), San Antonio, TX 78249-0669, USA*

Received 1 July 2006; Revised 4 December 2006; Accepted 11 May 2007

Recommended by Ahmed H. Tewfik

We investigate in this paper reverse engineering of gene regulatory networks from time-series microarray data. We apply dynamic Bayesian networks (DBNs) for modeling cell cycle regulations. In developing a network inference algorithm, we focus on soft solutions that can provide a posteriori probability (APP) of network topology. In particular, we propose a variational Bayesian structural expectation maximization algorithm that can learn the posterior distribution of the network model parameters and topology jointly. We also show how the obtained APPs of the network topology can be used in a Bayesian data integration strategy to integrate two different microarray data sets. The proposed VBSEM algorithm has been tested on yeast cell cycle data sets. To evaluate the confidence of the inferred networks, we apply a moving block bootstrap method. The inferred network is validated by comparing it to the KEGG pathway map.

1. INTRODUCTION

With the completion of the human genome project and successful sequencing genomes of many other organisms, emphasis of postgenomic research has been shifted to the understanding of functions of genes [1]. We investigate in this paper reverse engineering gene regulatory networks (GRNs) based on time-series microarray data. GRNs are the functioning circuitry in living organisms at the gene level. They display the regulatory relationships among genes in a cellular system. These regulatory relationships are involved directly and indirectly in controlling the production of protein and in mediating metabolic processes. Understanding GRNs can provide new ideas for treating complex diseases and breakthroughs for designing new drugs.

GRNs cannot be measured directly but can be inferred based on their inputs and outputs. This process of recovering GRNs from their inputs and outputs is referred to as reverse engineering GRNs [2]. The inputs of GRNs are a sequence of signals and the outputs are gene expressions at either the mRNA level or the protein level. One popular technology that measures expressions of a large amount of gene at the mRNA levels is microarray. It is not surprising that microarray data have been a popular source for uncovering GRNs [3, 4]. Of particular interest to this paper are time-series microarray data, which are generated from a cell cycle process. Using the time-series microarray data, we aim to uncover the underlying GRNs that govern the process of cell cycles.

Mathematically, reverse engineering GRNs are a traditional inverse problem, whose solutions require proper modeling and learning from data. Despite many existing methods for solving inverse problems, solutions to the GRNs problem are however not trivial. Special attention must be paid to the enormously large scale of the unknowns and the difficulty from the small sample size, not to mention the inherent experimental defects, noisy readings, and so forth. These call for powerful mathematic modeling together with reliable inference. At the same time, approaches for integrating different types of relevant data are desirable. In the literature, many different models have been proposed for both static, cell cycle networks including probabilistic Boolean networks [5, 6], (dynamic) Bayesian networks [7–9], differential

equations [10], and others [11, 12]. Unlike in the case of static experiments, extra effort is needed to model temporal dependency between samples for the time-series experiments. Such time-series models can in turn complicate the inference, thus making the task of reverse engineering even tougher than it already is.

In this paper, we apply dynamic Bayesian networks (DBNs) to model time-series microarray data. DBNs have been applied to reverse engineering GRNs in the past [13–18]. Differences among the existing work are the specific models used for gene regulations and the detailed inference objectives and algorithms. These existing models include discrete binomial models [14, 17], linear Gaussian models [16, 17], and spline function with Gaussian noise [18]. We choose to use the linear Gaussian regulatory model in this paper. Linear Gaussian models model the continuous gene expression level directly, thus preventing loss of information in using discrete models. Even though linear Gaussian models could be less realistic, network inference over linear Gaussian models is relatively easier than that for nonlinear and/or non Gaussian models, therefore leading to more robust results. It has been shown in [19] that if taking both computational complexity and inference accuracy into consideration, linear Gaussian models are favored over nonlinear regulatory models. In addition, this model actually models the joint effect of gene regulation and microarray experiments and the model validity is better evaluated from the data directly. In this paper, we provide the statistical test of the validity of the linear Gaussian model.

To learn the proposed DBNs from time-series data, we aim at soft Bayesian solutions, that is, the solutions that provide the *a posteriori probabilities* (APPs) of the network topology. This requirement separates the proposed solutions with most of the existing approaches such as greedy search and simulated-annealing-based algorithms, all of which produce only point estimates of the networks and are considered as "hard" solutions. The advantage of soft solutions has been demonstrated in digital communications [20]. In the context of GRNs, the APPs from the soft solutions provide valuable measurements of confidence on inference, which is difficult with hard solutions. Moreover, the obtained APPs can be used for Bayesian data integration, which will be demonstrated in the paper. Soft solutions including Markov chain Monte Carlo (MCMC) sampling [21, 22] and variational Bayesian expectation maximization (VBEM) [16] have been proposed for learning the GRNs. However, MCMC sampling is only feasible for small networks due to its high complexity. In contrast, VBEM has been shown to be much more efficient. However, the VBEM algorithm in [16] was developed only for parameter learning. It therefore cannot provide the desired APPs of topology. In this paper, we propose a new variational Bayesian structural EM (VBSEM) algorithm that can learn both parameters and topology of a network. The algorithm still maintains the general feature of VBEM for having low complexity, thus it is appropriate for learning large networks. In addition, it estimates the APPs of topology directly and is suitable for Bayesian data integration. To this end, we discuss a simple Bayesian strategy for integrating two microarray data sets by using the APPs obtained from VBSEM.

We apply the VBSEM algorithm to uncover the yeast cell cycle networks. To obtain the statistics of the VBSEM inference results and to overcome the difficulty of the small sample size, we apply a moving block bootstrap method. Unlike conventional bootstrap strategy, this method is specifically designed for time-series data. In particular, we propose a practical strategy for determining the block length. Also, to serve our objective of obtaining soft solutions, we apply the bootstrap samples for estimating the desired APPs. Instead of making a decision of the network from each bootstrapped data set, we make a decision based on the bootstrapped APPs. This practice relieves the problem of small sample size, making the solution more robust.

The rest of the paper is organized as follows. In Section 2, DBNs modeling of the time-series data is discussed. The detailed linear Gaussian model for gene regulation is also provided. In Section 3, objectives on learning the networks are discussed and the VBSEM algorithm is developed. In Section 4, a Bayesian integration strategy is illustrated. In Section 5, the test results of the proposed VBEM on the simulated networks and yeast cell cycle data are provided. A bootstrap method for estimating the APPs is also discussed. The paper concludes in Section 6.

2. MODELING WITH DYNAMIC BAYESIAN NETWORKS

Like all graphical models, a DBN is a marriage of graphical and probabilistic theories. In particular, DBNs are a class of directed acyclic graphs (DAGs) that model probabilistic distributions of stochastic dynamic processes. DBNs enable easy factorization on joint distributions of dynamic processes into products of simpler conditional distributions according to the inherent Markov properties, and thus greatly facilitate the task of inference. DBNs are shown to be a generalization of a wide range of popular models, which include hidden Markov models (HMMs) and Kalman filtering models, or state-space models. They have been successfully applied in computer vision, speech processing, target tracking, and wireless communications. Refer to [23] for a comprehensive discussion on DBNs.

A DBN consists of nodes and directed edges. Each node represents a variable in the problem while a directed edge indicates the direct association between the two connected nodes. In a DBN, the direction of an edge can carry the temporal information. To model the gene regulation from cell cycle using DBNs, we assume to have a microarray that measures the expression levels of G genes at $N+1$ evenly sampled consecutive time instances. We then define a random variable matrix $\mathbf{Y} \in \mathcal{R}^{G\times(N+1)}$ with the (i,n)th element $y_i(n-1)$ denoting the expression level of gene i measured at time $n-1$ (see Figure 1). We further assume that the gene regulation follows a first-order time-homogeneous Markov process. As a result, we need only to consider regulatory relationships between two consecutive time instances and this relationship remains unchanged over the course of the microarray experiment. This assumption may be insufficient, but it will

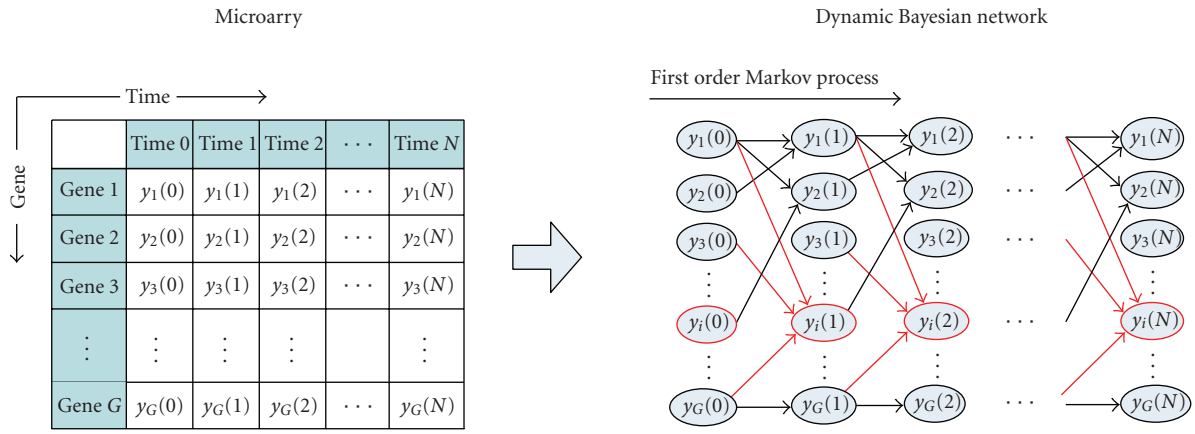

Figure 1: A dynamic Bayesian network modeling of time-series expression data.

facilitate the modeling and inference. Also, we call the regulating genes the "parent genes," or "parents" for short.

Based on these definitions and assumptions, the joint probability $p(\mathbf{Y})$ can be factorized as $p(\mathbf{Y}) = \prod_{1\le n\le N} p(\mathbf{y}(n) \mid \mathbf{y}(n-1))$, where $\mathbf{y}(n)$ is the vector of expression levels of all genes at time n. In addition, we assume that given $\mathbf{y}(n-1)$, the expression levels at n become independent. As a result, $p(\mathbf{y}(n) \mid \mathbf{y}(n-1))$, for all n, can be further factorized as $p(\mathbf{y}(n) \mid \mathbf{y}(n-1)) = \prod_{1\le i\le G} p(y_i(n) \mid \mathbf{y}(n-1))$. These factorizations suggest the structure of the proposed DBNs illustrated in Figure 1 for modeling the cell cycle regulations. In this DBN, each node denotes a random variable in $\mathbf{Y}$ and all the nodes are arranged the same way as the corresponding variables in the matrix $\mathbf{Y}$. An edge between two nodes denotes the regulatory relationship between the two associated genes and the arrow indicates the direction of regulation. For example, we see from Figure 1 that genes 1, 3, and G regulate gene i. Even though, like all Bayesian networks, DBNs do not allow circles in the graph, they, however, are capable of modeling circular regulatory relationship, an important property that is not possessed by regular Bayesian networks. As an example, a circular regulation can be seen in Figure 1 between genes 1 and 2 even though no circular loops are used in the graph.

To complete modeling with DBNs, we need to define the conditional distributions of each child node over the graph. Then the desired joint distribution can be represented as a product of these conditional distributions. To define the conditional distributions, we let $\mathbf{pa}_i(n)$ denote a column vector of the expression levels of all the parent genes that regulate gene i measured at time n. As an example in Figure 1, $\mathbf{pa}_i(n)^T = [y_1(n), y_3(n), y_G(n)]$. Then, the conditional distribution of each child node over the DBNs can be expressed as $p(y_i(n) \mid \mathbf{pa}_i(n-1))$, for all i. To determine the expression of the distributions, we assume linear regulatory relationship, that is, the expression level of gene i is the result of linear combination of the expression levels of the regulating genes at previous sample time. To make further simplification, we assume the regulation is a time-homogeneous process. Mathematically, we have the following expression:

$$y_i(n) = \mathbf{w}_i^T \mathbf{pa}_i(n-1) + e_i(n), \quad n = 1, 2, \ldots, N, \tag{1}$$

where $\mathbf{w}_i \in \mathcal{R}$ is the weight vector independent of time n and $e_i(n)$ is assumed to be white Gaussian noise with variance σ_i^2. We provide in Section 5 the statistical test of the validity of white Gaussian noise. The weight vector is indicative of the degree and the types of the regulation [16]. A gene is upregulated if the weight is positive and is down-regulated otherwise. The magnitude (absolute value) of the weight indicates the degree of regulation. The noise variable is introduced to account for modeling and experimental errors. From (1), we obtain that the conditional distribution is a Gaussian distribution, that is,

$$p(y_i(n) \mid \mathbf{pa}_i(n-1)) = \mathcal{N}(\mathbf{w}_i^T \mathbf{pa}_i(n-1), \sigma_i^2). \tag{2}$$

In (1), the weight vector $\mathbf{w}_i$ and the noise variance σ_i^2 are the unknown parameters to be determined.

2.1. Objectives

Based on the above dynamic Bayesian networks formulation, our work has two objectives. First, given a set of time-series data from a single experiment, we aim at uncovering the underlying gene regulatory networks. This is equivalent to learning the structure of the DBNs. In specific, if we can determine that genes 2 and 3 are the parents of gene 1 in the DBNs, there will be directed links going from gene 2 and 3 to gene 1 in the uncovered GRNs. Second, we are also concerned with integrating two data sets of the same network from different experiments. Through integrating the two data sets, we expect to improve the confidence of the inferred networks obtained from a single experiment. To achieve these two objectives, we propose in the following an efficient variational Bayesian structural EM algorithm to learn the network and a Bayesian approach for data integration.

3. LEARNING THE DBN WITH VBSEM

Given a set of microarray measurements on the expression levels in cell cycles, the task of learning the above DBN consists of two parts: structure learning and parameter learning. The objective of structure learning is to determine the topology of the network or the parents of each gene. This is essentially a problem of model or variable selection. Under a given structure, parameter learning involves the estimation of the unknown model coefficients of each gene: the weight vector $\mathbf{w}_i$ and the noise variance σ_i^2, for all i. Since the network is fully observed and, given parent genes, the gene expression levels at any given time are independent, we can learn the parents and the associated model parameters of each gene separately. Thus we only discuss in the following the learning process on gene i.

3.1. A Bayesian criterion for network structural learning

Let $\mathcal{S}_i = \{S_i^{(1)}, S_i^{(2)}, \ldots, S_i^{(K)}\}$ denote a set of K possible network topologies for gene i, where each element represents a topology derived from a possible combination of the parents of gene i. The problem of structure learning is to select the topology from $\mathcal{S}_i$ that is best supported by the microarray data.

For a particular topology $S_i^{(k)}$, we use $\mathbf{w}_i^{(k)}$, $\mathbf{pa}_i^{(k)}$, $\mathbf{e}_i^{(k)}$ and σ_{ik}^2 to denote the associated model variables. We can then express (1) for $S_i^{(k)}$ in a more compact matrix-vector form

$$\mathbf{y}_i = \mathbf{Pa}_i^{(k)}\mathbf{w}_i^{(k)} + \mathbf{e}_i^{(k)}, \tag{3}$$

where $\mathbf{y}_i = [y_i(1), \ldots, y_i(N)]^T$, $\mathbf{Pa}_i^{(k)} = [\mathbf{pa}_i^{(k)}(0), \mathbf{pa}_i^{(k)}(1), \ldots, \mathbf{pa}_i^{(k)}(N-1)]^\top$, $\mathbf{e}_i^{(k)} = [e_i^{(k)}(1), e_i^{(k)}(2), \ldots, e_i^{(k)}(N)]^\top$, and $\mathbf{w}_i^{(k)}$ is independent of time n.

The structural learning can be performed under the Bayesian paradigm. In particular, we are interested in calculating the *a posteriori* probabilities of the network topology $p(S_i^{(k)} \mid \mathbf{Y})$, for all k. The APPs will be important for the data integration tasks. They also provide a measurement of confidence on inferred networks. Once we obtain the APPs, we can select the most probable topology $\overline{S}_i$ according to the maximum *a posteriori* (MAP) criterion [24], that is,

$$\overline{S}_i = \arg\max_{S_i^{(k)} \in \mathcal{S}_i} p(S_i^{(k)} \mid \mathbf{Y}). \tag{4}$$

The APPs are calculated according to the Bayes theorem,

$$\begin{aligned} p(S_i^{(k)} \mid \mathbf{Y}) &= \frac{p(\mathbf{y}_i \mid S_i^{(k)}, \mathbf{Y}_{-i})\, p(S_i^{(k)} \mid \mathbf{Y}_{-i})}{p(\mathbf{y}_i \mid \mathbf{Y}_{-i})} \\ &= \frac{p(\mathbf{y}_i \mid \mathbf{Pa}_i^{(k)})\, p(S_i^{(k)})}{p(\mathbf{y}_i \mid \mathbf{Y}_{-i})}, \end{aligned} \tag{5}$$

where $\mathbf{Y}_{-i}$ represents a matrix obtained by removing $\mathbf{y}_i$ from $\mathbf{Y}$, the second equality is arrived at from the fact that given $S_i^{(k)}$, $\mathbf{y}_i$ depends on $\mathbf{Y}_{-i}$ only through $\mathbf{Pa}_i^{(k)}$, and the last equation is due to that given $\mathbf{Pa}_i^{(k)}$, $S_i^{(k)}$ is known automatically but $S_i^{(k)}$ cannot be determined from $\mathbf{Y}_{-i}$. Note also that there is a slight abuse of notation in (4). $\mathbf{Y}$ in $p(S_i^{(k)} \mid \mathbf{Y})$ denotes a realization of expression levels measured from a microarray experiment.

To calculate the APPs according to (5), the marginal likelihood $p(\mathbf{y}_i \mid \mathbf{Pa}_i^{(k)})$ and the marginalization constant $p(\mathbf{y}_i \mid \mathbf{Y}_{-i})$ need to be determined. It has been shown that with conjugate priors on the parameters, we can obtain $p(\mathbf{y}_i \mid \mathbf{Pa}_i^{(k)})$ analytically [21]. However, $p(\mathbf{y}_i \mid \mathbf{Y}_{-i})$ becomes computationally prohibited for large networks because computing $p(\mathbf{y}_i \mid \mathbf{Y}_{-i})$ involves summation over 2^G terms. This difficulty with $p(\mathbf{y}_i \mid \mathbf{Y}_{-i})$ makes the exact calculation of the APPs infeasible. Numerical approximation must be therefore employed to estimate the APPs instead. Monte Carlo sampling-based algorithms have been reported in the literature for this approximation [21]. They are however computationally very expensive and do not scale well with the size of networks. In what follows, we propose a much more efficient solution based on variational Bayesian EM.

3.2. Variational Bayesian structural expectation maximization

To develop the VBSEM algorithm, we define a G-dimensional binary vector $\mathbf{b}_i \in \{0,1\}^G$, where $b_i(j) = 1$ if gene j is a parent of gene i in the topology S_i and $b_i(j) = 0$ otherwise. We can actually consider $\mathbf{b}_i$ as an equivalent representation of S_i and finding the structure S_i can thus equate to determining the values of $\mathbf{b}_i$. Consequently, we can replace S_i in all the above expressions by $\mathbf{b}_i$ and turn our attention to estimate the equivalent APPs $p(\mathbf{b}_i \mid \mathbf{Y})$.

The basic idea behind VBSEM is to approximate the intractable APPs of topology with a tractable distribution $q(\mathbf{b}_i)$. To do so, we start with a lower bound on the normalizing constant $p(\mathbf{y}_i \mid \mathbf{Y}_{-i})$ based on Jensen's inequality

$$\begin{aligned} &\ln p(\mathbf{y}_i \mid \mathbf{Y}_{-i}) \\ &\quad = \ln \sum_{\mathbf{b}_i} \int d\boldsymbol{\theta}_i\, p(\mathbf{y}_i \mid \mathbf{b}_i, \boldsymbol{\theta}_i)\, p(\mathbf{b}_i)\, p(\boldsymbol{\theta}_i) \end{aligned} \tag{6}$$

$$\geq \int d\boldsymbol{\theta}_i q(\boldsymbol{\theta}_i) \left[\sum_{\mathbf{b}_i} q(\mathbf{b}_i) \ln \frac{p(\mathbf{b}_i, \mathbf{y}_i \mid \boldsymbol{\theta}_i)}{q(\mathbf{b}_i)} + \ln \frac{p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)} \right], \tag{7}$$

where $\boldsymbol{\theta}_i = \{\mathbf{w}_i, \sigma_i^2\}$ and $q(\boldsymbol{\theta}_i)$ is a distribution introduced for approximating the also intractable marginal posterior distribution of parameters $p(\boldsymbol{\theta}_i \mid \mathbf{Y})$. The lower bound in (7) can serve as a cost function for determining the approximate distributions $q(\mathbf{b}_i)$ and $q(\boldsymbol{\theta}_i)$, that is, we choose $q(\mathbf{b}_i)$ and $q(\boldsymbol{\theta}_i)$ such that the lower bound in (7) is maximized. The solution can be obtained by variational derivatives and a coordinate ascent iterative procedure and is shown to include the following two steps in each iteration:

VBE step:

$$q^{(t+1)}(\mathbf{b}_i) = \frac{1}{Z_{\mathbf{b}_i}} \exp\left[\int d\boldsymbol{\theta}_i q^{(t)}(\boldsymbol{\theta}_i) \ln p(\mathbf{b}_i, \mathbf{y}_i \mid \boldsymbol{\theta}_i) \right], \tag{8}$$

The VBSEM algorithm
(1) Initialization
Initialize the mean and the covariance matrices of the approximate distributions as described in Appendix A.
(2) VBE step: structural learning
Calculate the approximate posterior distributions of topology $q(\mathbf{b}_i)$ using (B.1)
(3) VBM step: parameter learning
Calculate the approximate parameter posterior distributions $q(\boldsymbol{\theta}_i)$ using (B.5)
(4) Compute $\mathcal{F}$
Compute the lower bound as described in Appendix A. If $\mathcal{F}$ increases, go to (2). Otherwise, terminate the algorithm.

Algorithm 1: The summary of VBSEM algorithm

VBM step:

$$q^{(t+1)}(\boldsymbol{\theta}_i) = \frac{1}{Z_{\theta_i}} p(\boldsymbol{\theta}_i) \exp\left[\sum_{\mathbf{b}_i} q^{(t+1)}(\mathbf{b}_i) \ln p(\mathbf{b}_i, \mathbf{y}_i \mid \boldsymbol{\theta}_i)\right], \tag{9}$$

where t and $t+1$ are iteration numbers and $Z_{\mathbf{b}_i}$ and Z_{θ_i} are the normalizing constants to be determined. The above procedure is commonly referred to as variational Bayesian expectation maximization algorithm [25]. The VBEM can be considered as a probabilistic version of the popular EM algorithm in the sense that it learns the distribution instead of finding a point solution as in EM. Apparently, to carry out this iterative approximation, analytical expressions must exist in both VBE and VBM steps. However, it is difficult to come up with an analytical expression at least in the VBM step since the summation is NP hard. To overcome this problem, we enforce the approximation $q(\mathbf{b}_i)$ to be a multivariate Gaussian distribution. The Gaussian assumption on the discrete variable $\mathbf{b}_i$ facilitates the computation in the VBEM algorithm, circumventing the 2^G summations. Although $p(\mathbf{b}_i \mid \mathbf{Y})$ is a high-dimensional discrete distribution, the defined Gaussian approximation will guarantee the approximations to fall in the exponential family, and as a result the subsequent computations in the VBEM iterations can be carried out exactly [25]. In specific, by choosing conjugate priors for both $\boldsymbol{\theta}_i$ and $\mathbf{b}_i$ as described in Appendix A, we can show that the calculations in both VBE and VBM steps can be performed analytically. The detailed derivations are included in Appendix B. Unlike the common VBEM algorithm, which learns only the distributions of parameters, the proposed VBEM learns the distributions of both structure and parameters. We, therefore, call the algorithm VB structural EM (VBSEM). The algorithm of VBSEM for learning the DBNs under study is summarized in Algorithm 1.

When the algorithm converges, we obtain $q(\mathbf{b}_i)$, a multivariate Gaussian distribution and $q(\boldsymbol{\theta}_i)$. Based on $q(\mathbf{b}_i)$, we need then to produce a discrete distribution as a final estimate of $p(\mathbf{b}_i)$. Direct discretization in the variable space is computationally difficult. Instead, we propose to work with the marginal APPs from model averaging. To this end, we first obtain $q(b_i(l))$, for all l from $q(\mathbf{b}_i)$ and then approximate the marginal APPs $p(b_i(l) \mid \mathbf{Y})$, for all l, by

$$p(b_i(l) = 1 \mid \mathbf{Y}) = \frac{q(b_i(l) = 1)}{q(b_i(l) = 1) + q(b_i(l) = 0)}. \tag{10}$$

Instead of the MAP criterion, decisions on $\mathbf{b}_i$ can be then made in a bitwise fashion based on the marginal APPs. In specific, we have

$$\hat{b}_i(l) = \begin{cases} 1 & \text{if } p(b_i(l) \mid \mathbf{Y}) \geq \rho, \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where ρ is a threshold. When $\hat{b}_i(l) = 1$, it implies that gene l is a regulator of gene i in the topology of gene i. Meanwhile, parameters can be learned from $q(\boldsymbol{\theta}_i)$ easily based on the minimum mean-squared-error criterion (MMSE) and they are

$$\hat{\mathbf{w}}_i = \mathbf{m}_{\mathbf{w}_i}, \qquad \hat{\sigma}_i^2 = \frac{\beta}{\alpha - 2}, \tag{12}$$

where $\mathbf{m}_{\mathbf{w}_i}$, β, and α are defined in Appendix B according to (B.5).

4. BAYESIAN INTEGRATION OF TWO DATA SETS

A major task of the gene network research is to integrate all prevalent data sets about the same network from different sources so as to improve the confidence of inference. As indicated before, the values of $\mathbf{b}_i$ define the parent sets of gene i, and thus the topology of the network. The APPs obtained from the VBSEM algorithm provide us with an avenue to pursue Bayesian data integration.

We illustrate here an approach for integrating two microarray data sets $\mathbf{Y}^1$ and $\mathbf{Y}^2$, each produced from an experiment under possibly different conditions. The premise for combining the two data sets is that they are the experimental outcomes of the same underlying gene network, that is, the topologies S_i or $\mathbf{b}_i$, for all i are the same in the respective data models. Direct combination of the two data sets at the data level requires many preprocesses including scaling, alignment, and so forth. The preprocessing steps introduce noise and potential errors to the original data sets. Instead, we propose to perform data integration at the topology level. The objective of topology-level data integration is to obtain the APPs of $\mathbf{b}_i$ from the combined data sets $p(\mathbf{b}_i \mid \mathbf{Y}^1, \mathbf{Y}^2)$ and then make inference on the gene network structures accordingly.

To obtain $p(\mathbf{b}_i \mid \mathbf{Y}^1, \mathbf{Y}^2)$, we factor it according to the Bayes rule as

$$\begin{aligned} p(\mathbf{b}_i \mid \mathbf{Y}^1, \mathbf{Y}^2) &= \frac{p(\mathbf{Y}^2 \mid \mathbf{b}_i) p(\mathbf{Y}^1 \mid \mathbf{b}_i) p(\mathbf{b}_i)}{p(\mathbf{Y}^1) p(\mathbf{Y}^2)} \\ &= \frac{p(\mathbf{Y}^2 \mid \mathbf{b}_i) p(\mathbf{b}_i \mid \mathbf{Y}^1)}{p(\mathbf{Y}^2)}, \end{aligned} \tag{13}$$

where $p(\mathbf{Y}^2 \mid \mathbf{b}_i)$ is the marginalized likelihood functions of data set 2 and $p(\mathbf{b}_i \mid \mathbf{Y}^1)$ is the APPs obtained from data set 1.

The above equation suggests a simple scheme to integrate the two data sets: we start with a data set, say $\mathbf{Y}_1$, and calculate the APPs $p(\mathbf{b}_i \mid \mathbf{Y}^1)$; then by considering $p(\mathbf{b}_i \mid \mathbf{Y}^1)$ as the prior distribution, the data set $\mathbf{Y}^1$ is integrated with $\mathbf{Y}_i^2$ according to (13). By this way, we obtain the desired APPs $p(\mathbf{b}_i \mid \mathbf{Y}^1, \mathbf{Y}^2)$ from the combined data sets. To implement this scheme, the APPs of the topology must be computed and the proposed VBSEM can be applied for the task. This new scheme provides a viable and efficient framework for Bayesian data integration.

5. RESULTS

5.1. Test on simulated systems

5.1.1. Study based on precision-recall curves

In this section, we validate the performance of the proposed VBSEM algorithm using synthetic networks whose characteristics are as realistic as possible. This study was accomplished through the calculation of the precision-recall curves. Among the scientific community in this field, it is common to employ the ROC analysis to study the performance of a proposed algorithm. However, since genetic networks are sparse, the number of false positives far exceeds the number of true positives. Thus, the specificity is inappropriate as even small deviation from a value of 1 will result in a large number of false positives. Therefore, we choose the precision-recall curves in evaluating the performance. *Precision* corresponds to the expected success rate in the experimental validation of the predicted interactions and it is calculated as $T_P/(T_P+F_P)$, where T_P is the number of true positives and F_P is the number of false positives. *Recall*, on the other hand, indicates the probability of correctly detecting a true positive and it is calculated as $T_P/(T_P+F_N)$, where F_N is the number of false negatives. In a good system, precision decreases as recall increases and the higher the area under the curve is the better the system is.

To accomplish our objective, we simulated 4 networks with 30, 100, 150, and 200 genes, respectively. For each tested network, we collected only 30 time samples for each gene, which mimics the realistic small sample scenario. Regarding the regulation process, each gene had either none, one, two, or three parents. Besides, the number of parents was selected randomly for each gene. The weights associated to each regulation process were also chosen randomly from an interval that contains the typical estimated values when working with the real microarray data. As for the nature of regulation, the signs of the weights were selected randomly as well. Finally, the data values of the network outputs were calculated using the linear Gaussian model proposed in (1). These data values were taken after the system had reached stationarity and they were in the range of the observations corresponding to real microarray data.

In Figure 2, the precision-recall curves are plotted for different settings. In order to construct these curves, we started by setting a threshold ρ for the APPs. This threshold ρ is between 0 to 1 and it was used as in (11): for each possible regulation relationship between two genes, if its APP is greater than ρ, then the link is considered to exist, whereas if the APP is lower than ρ, the link is not considered. We calculated the precision and the recall for each selected threshold between 0 and 1. We plotted the results in blue for the case with $G = 30$, black for $G = 100$, red for $G = 150$, and green for $G = 200$. As expected, the performance got worse as the number of genes increases. One measure of this degradation is shown in Table 1 where we calculated the area under each curve (AUC).

Table 1: Area under each curve.

Setting	$G = 30$	$G = 100$	$G = 150$	$G = 200$
AUC	0.8007	0.7253	0.6315	0.5872

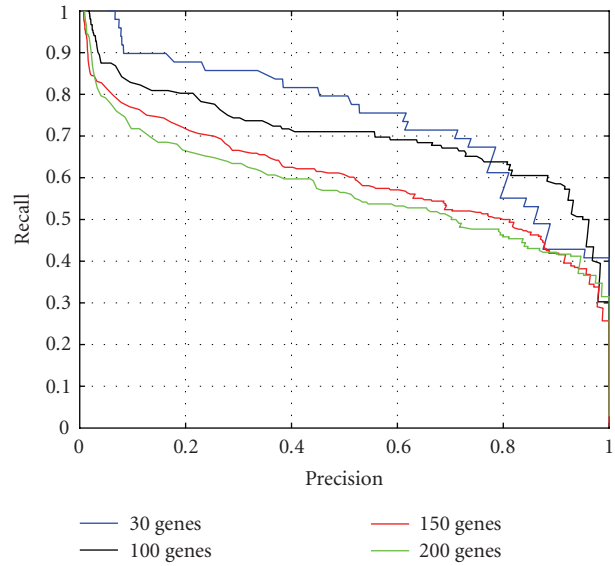

Figure 2: Precision-recall curve.

To further quantify the performance of the algorithms, we calculated the F-score. F-score constitutes an evaluation measure that combines precision and recall and it can be calculated as

$$F_\alpha = \frac{1}{\alpha(1/\text{precision}) + (1-\alpha)(1/\text{recall})}, \tag{14}$$

where α is a weighting factor and a large α means that the recall is more important, whereas a small α means that precision is more important. In general, $\alpha = 0.5$ is used, where the importance of precision and the importance of recall are even and F_α is called *harmonic mean*. This value is equal to 1 when both precision and recall are 100%, and 0 when one of them is close to 0. Figure 3 depicts the value of the harmonic mean as a function of the APP threshold ρ for the VBSEM algorithm. As it can be seen, the performance of the algorithm for $G = 30$ is better than the performance for any other setting. However, we can also see that there is almost no performance degradation between the curve corresponding to $G = 30$ and the one for $G = 100$ in the APP threshold interval from 0.5 to 0.7. The same observation can be obtained for

Table 2: Computation time for different sizes of networks.

Setting	$G_1 = 100$	$G_3 = 200$	$G_2 = 500$	$G_4 = 1000$
Computation time (s)	19.2871	206.5132	889.8120	12891.8732

Table 3: Number of errors in 100 Monte Carlo trials.

No. of errors	$G = 5, N = 5$	$G = 5, N = 10$
VBEM (no. of iterations = 10)	62	1
Gibbs sampling (500 samples)	126	5

curves $G = 150$ and $G = 200$ in the interval from 0.5 to 0.6. In general, in the interval from 0.5 to 0.7, the degradation of the algorithm performance is small for reasonable harmonic mean values (i.e., > 0.5).

To demonstrate the scalability of the VBSEM algorithm, we have studied the harmonic mean for simulated networks characterized by the following settings: ($G_1 = 1000$, $N_1 = 400$), ($G_2 = 500$, $N_2 = 200$), ($G_3 = 200$, $N_3 = 80$), and ($G_4 = 100$, $N_4 = 40$). As it can be noticed, the ratio G_i/N_i has been kept constant in order to maintain the proportion between the amount of nodes in the network and the amount of information (samples). The results were plotted in Figure 4 where we have represented the harmonic mean as a function of the APP threshold. The closeness of the curves at APP threshold equal to 0.5 supports the good scalability of the proposed algorithm. We have also recorded the computation time of VBSEM for each network and listed them in Table 2. The results were obtained with a standard PC with 3.4 GHz and 2 GB RAM.

5.1.2. Comparison with the Gibbs sampling

We tested in this subsection the VBSEM algorithm on a simulated network in order to compare it with the Gibbs sampling [26]. We simulated a network of 20 genes and generated their expressions based on the proposed DBNs and the linear Gaussian regulatory model with Gaussian distributed weights. We focused on a particular gene in the simulated networks. The gene was assumed to have two parents. We compared the performance of VBSEM and Gibbs sampling in recovering the true networks. In Table 3, we present the number of errors in 100 Monte Carlo tests. For the Gibbs sampling, 500 Monte Carlo samples were used. We tested the algorithms under different settings. In the table, N stands for the number of time samples and G is the number of genes. As it can be seen, the VBSEM outperforms Gibbs sampling even in an underdetermined system. Since the VBSEM has much lower complexity than Gibbs sampling, the proposed VBSEM algorithm is better suited for uncovering large networks.

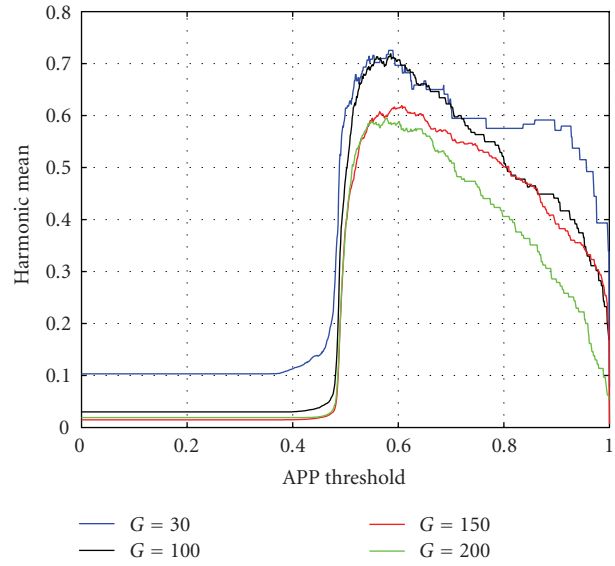

Figure 3: Harmonic mean as a function of the APP threshold.

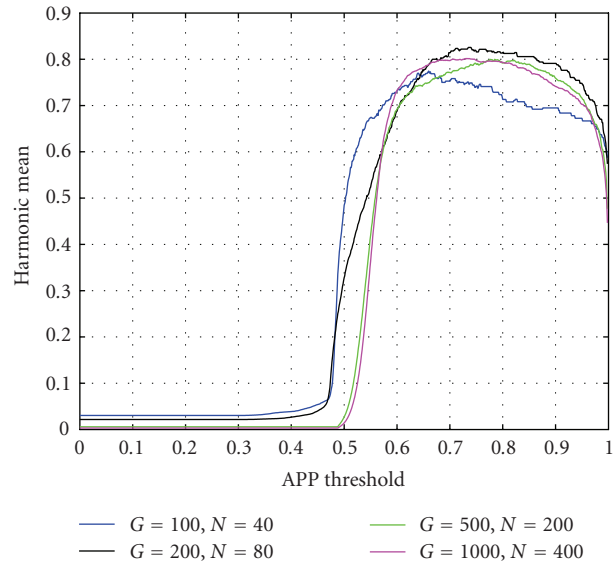

Figure 4: Harmonic mean as a function of the APP threshold to proof scalability.

5.2. Test on real data

We applied the proposed VBSEM algorithm on cDNA microarray data sets of 62 genes in the yeast cell cycle reported in [27, 28]. The data set 1 [27] contains 18 samples evenly measured over a period of 119 minutes where a synchronization treatment based on α mating factor was used. On the other hand, the data set 2 [28] contains 17 samples evenly measured over 160 minutes and a temperature-sensitive CDC15 mutant was used for synchronization. For each gene, the data is represented as the $\log_2\{$(expression at time t)/(expression in mixture of control cells)$\}$. Missing

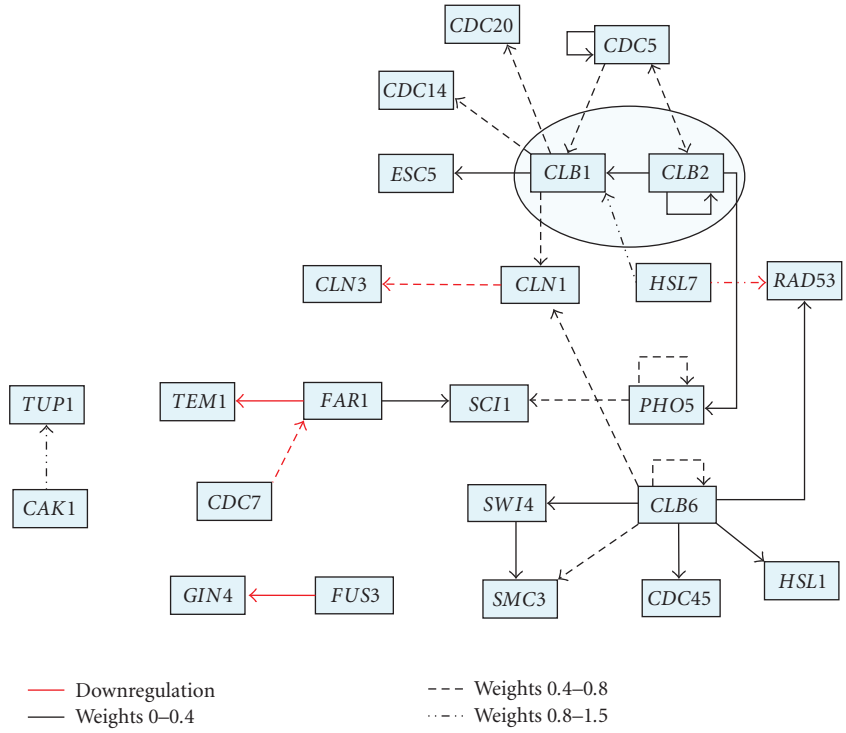

Figure 5: Inferred network using the α data set of [27].

values exist in both data sets, which indicate that there was no strong enough signal in the spot. In this case, simple spline interpolation was used to fill in the missing data. Note the time step that differs in each data set can be neglected since we assume a time-homogeneous regulating process.

When validating the results, the main objective is to determine the level of confidence of the connections in the inferred network. The underlying intuition is that we should be more confident on features that would still be inferred when we perturb the data. Intuitively, this can be performed on multiple independent data sets generated from repeated experiments. However, in this case and many other practical scenarios, only one or very limited data replicates are available and the sample size in each data set is small. The question is then how to produce the perturbed data from the limited available data sets and at the same time maintain the underlying statistical features of the data set. One way to achieve it is to apply the bootstrap method [29]. Through bootstrapping the data set, we can generate multiple pseudoindependent data sets, each of which still maintains the statistics of the original data. The bootstrap methods have been used extensively for static data sets. When applied to time-series data, an additional requirement is to maintain as much as possible the inherent time dependency between samples in the bootstrapped data sets. This is important since the proposed DBNs modeling and VBSEM algorithm exploit this time dependency. Approaches have been studied in the bootstrap literatures to handle time-dependent samples and we adopt the popular moving block bootstrap method [30]. In moving block bootstrap, we created pseudo-data sets from the original data set by first randomly sampling blocks of sub-data sets and then putting them together to generate a new data set. The detailed steps can be summarized as follows.

(1) Select the length of the block L.
(2) Create the set of possible $n = N - L + 1$ blocks from data. These blocks are created in the followingway:

$$\mathbf{Z}_i = \mathbf{Y}(:, i : i + L - 1). \tag{15}$$

(3) Randomly sample with replacement $\lceil N/L \rceil$ blocks from the set of blocks $\{\mathbf{Z}_i\}_{i=1}^{N-L+1}$.
(4) Create the pseudo-data set by putting all the sampled blocks together and trim the size to N by removing the extra data samples.

A key issue in moving block bootstrap is to determine the block length L. The idea is to choose a large enough block length L so that observations more than L time units apart will be nearly independent. Many theoretical and applicable results have been developed on choosing the block length. However, they rely on large size of data samples and are computationally intensive. Here, we develop an easy and practical approach to determine the block length. We compute the autocorrelation function on data and choose the block length as the delay, at which the ACF becomes the smallest. The ACF in this case may not be reliable but it provides at least some measures of independence.

In Figure 5, we show the inferred network when the data set from [27] was considered and the moving block bootstrap

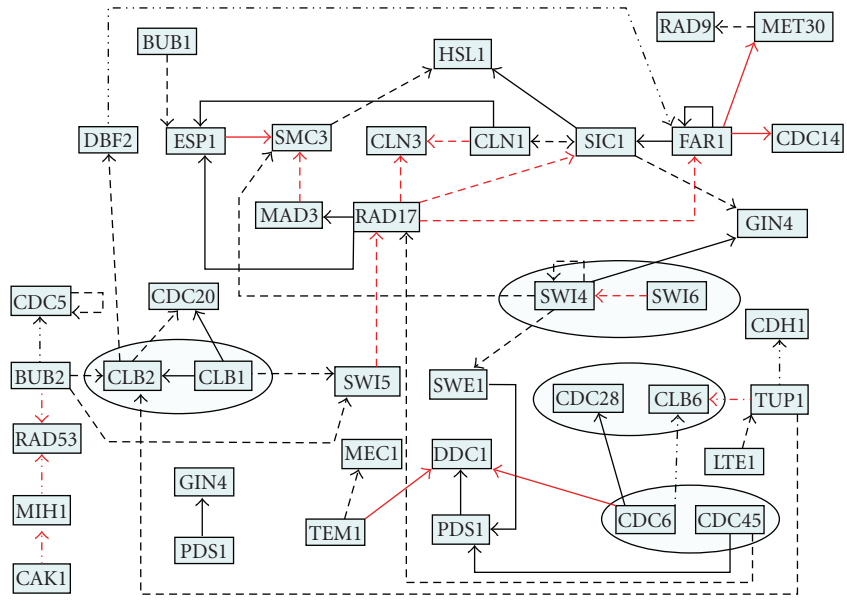

Figure 6: Inferred network using the CDC28 data set of [28].

was used to resample the observations. The total number of re-sample data sets was 500. In this plot, we only drew those links with the estimated APP higher than 0.6. Weused the solid lines to represent those links with weights between 0 and 0.4, the dotted lines for the links with weights between 0.4 and 0.8, and the lines with dashes and dots for those with weights higher than 0.8. The red color was used to represent downregulation. A circle enclosing some genes means that those corresponding proteins compose a complex. The edges inside these circles are considered as correct edges since genes inside the same circle will coexpress with some delay. In Table 4, we show the connections with some of the highest APPs found from the α data set of [27]. We compared them with the links in the KEGG pathway [31], and some of the links inferred by the proposed algorithm are predicted in it. We considered a connection as predicted when the parent is in the upper stream of the child in the KEGG. Furthermore, the proposed algorithm is also capable of predicting the nature of the relationship represented by the link through the weight. For example, the connection between CDC5 and CLB1 has a weight equal to 0.6568, positive, so it represents an upregulation as predicted in the KEGG pathway. Another example is the connection from CLB1 to CDC20; its APP is 0.6069 and its weight is 0.4505, again positive, so it stands for an up-regulation as predicted by the KEGG pathway.

In Figure 6, we depict the inferred network when the CDC28 data set of [28] was used. A moving block bootstrap was also used with the number of the bootstrap data sets equal to 500 again. Still, the links presented in this plots are those with the APP higher than 0.6. In Table 5, we show some of the connections with some of the highest APPs. We also compared them with the links in the KEGG pathway, and some of the links inferred by the proposed algorithm are also predicted in it. Furthermore, the proposed algorithm is

Table 4: Links with higher APPs obtained from the α data set of [27].

From	To	APPs	Comparison with KEGG
CDC5	*CLB1*	0.6558	Predicted
CLB6	*CDC45*	0.6562	Predicted
CLB6	*SMC3*	0.7991	Not predicted
SWI4	*SMC3*	0.6738	Not predicted
CLB6	*HSL1*	0.6989	Not predicted
CLB6	*CLN1*	0.7044	Predicted the other way round
CLN1	*CLN3*	0.6989	Predicted the other way round
PH05	*SIC1*	0.6735	Not predicted
CLB6	*RAD53*	0.6974	Not predicted
CDC5	*CLB2*	0.6566	Predicted
FUS3	*GIN4*	0.6495	Not predicted
PH05	*PHO5*	0.6441	Not predicted
CLB2	*CDC5*	0.6390	Predicted the other way round
CLB6	*SWI4*	0.6336	Predicted the other way round

also capable of predicting the nature of the relationship represented by the link through the weight. For example, the connection between TEM1 and *DDC1* has a weight equal to -0.3034; the negative sign represents a downregulation as predicted in the KEGG pathway. Another example is the connection from *CLB2* to *CDC20*, its APP is 0.6069 and its weight is 0.7763, this time positive, so it stands for an up-regulation as predicted by the KEGG pathway.

Model validation

To validate the proposed linear Gaussian model, we tested the normality of the prediction errors. If the prediction errors

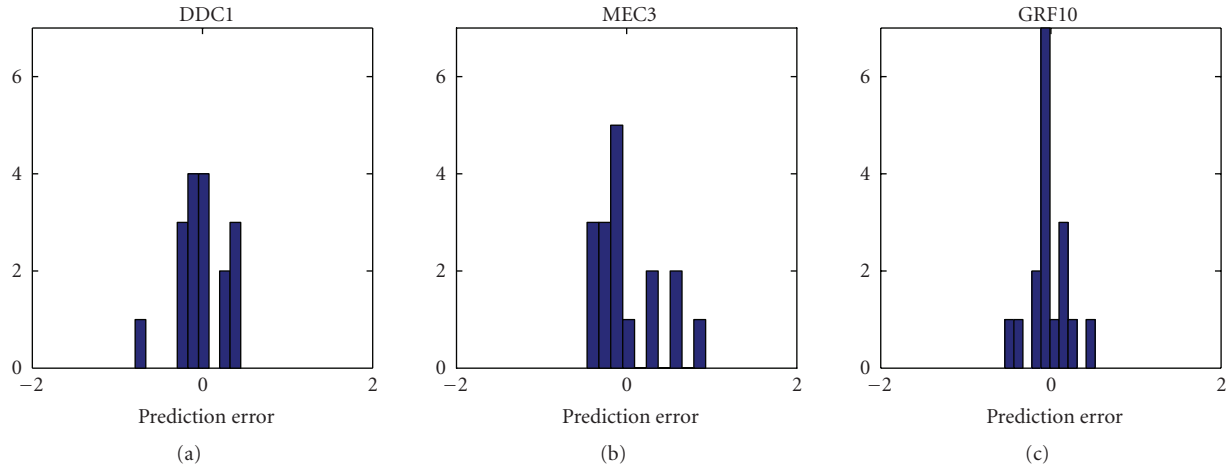

Figure 7: Histogram of prediction error in the α data set.

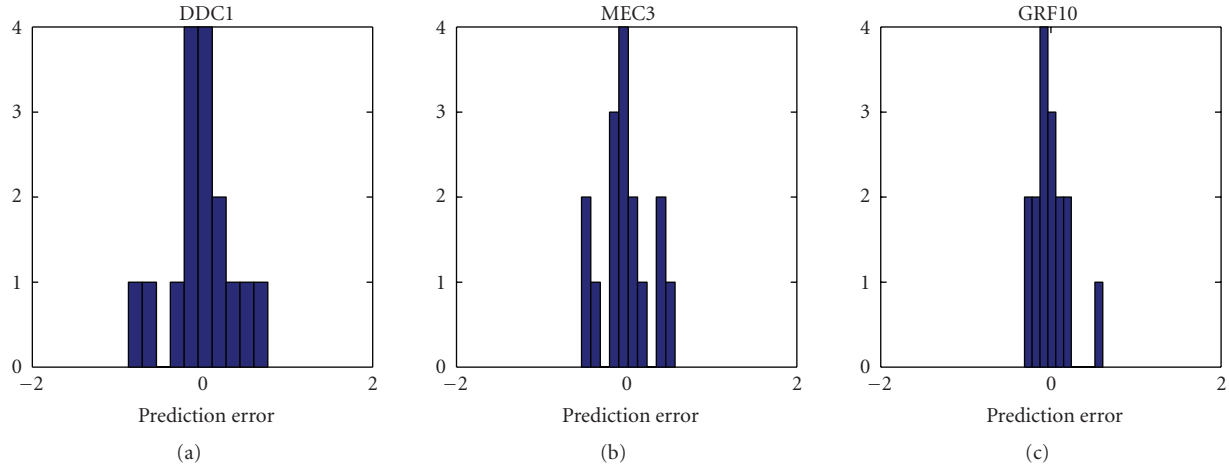

Figure 8: Histogram of prediction error in the CDC28 data set.

yield Gaussian distributions as in the linear model (1), it then proves the feasibility of linear Gaussian assumption on data.

Given the estimated $\hat{\mathbf{b}}_i$ and $\hat{\mathbf{w}}_i$ of gene i, the prediction error $\hat{\mathbf{e}}_i$ is obtained as

$$\hat{\mathbf{e}}_i = \mathbf{R}\widehat{\mathbf{W}}_i\hat{\mathbf{b}}_i - \mathbf{y}_i, \tag{16}$$

where $\widehat{\mathbf{W}}_i = \text{diag}(\hat{\mathbf{w}}_i)$ and $\mathbf{R} = \mathbf{T}\mathbf{Y}^\top$, with

$$\mathbf{T} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \vdots \\ & 1 & 0 \end{pmatrix}. \tag{17}$$

We show in Figures 7 and 8 examples of the histograms of the prediction errors for genes *DDC*1, *MEC*3, and *GRF*10 in the α and CDC28 data sets.

Those histograms exhibit the bell shape for the distribution of the prediction errors and such pattern is constant over all the genes. To examine the normality, we performed Kolmogorov-Smirnov goodness-of-fit hypothesis test (KSTEST) of the prediction errors for each gene. All the prediction errors pass the normality test at the significance level of 0.05, and therefore it demonstrates the validity of the proposed linear Gaussian assumption.

Results validation

To systematically present the results, we treated the KEGG map as the ground truth and calculated the statistics of the results. Even though there are still uncertainties, the KEGG map represents up-to-date knowledge about the dynamics of gene interaction and it should be reasonable to serve as a benchmark of results validation. In Tables 6 and 7, we enlisted the number of true positives (tp), true negatives (tn), false positives (fp), and false negative (fn) for the α and CDC28 data sets, respectively. We also varied the

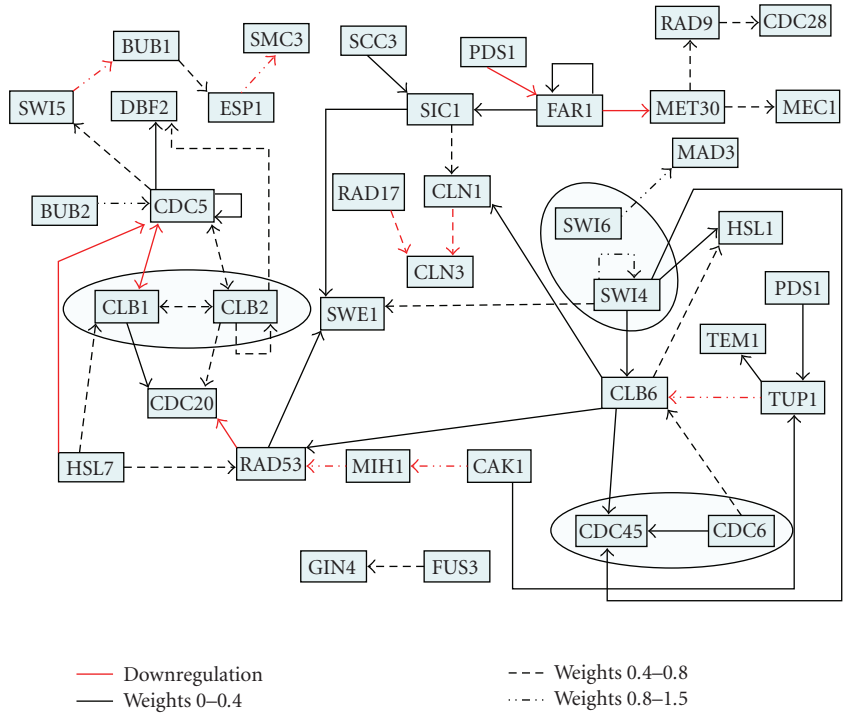

Figure 9: Inferred network by integrating the α and CDC28 data sets.

Table 5: Links with higher APPs obtained from the CDC28 data set of [28].

From	To	APPs	Comparison with KEGG
CLB1	*CDC20*	0.7876	Predicted the other way round
BUB1	*ESP1*	0.6678	Predicted
BUB2	*CDC5*	0.7145	Predicted
SIC1	*GIN4*	0.6700	Not predicted
SMC3	*HSL1*	0.6689	Not predicted
CLN1	*CLN3*	0.7723	Predicted the other way round
FAR1	*SIC1*	0.6763	Predicted
CLN1	*SIC1*	0.6640	Predicted
CDC5	*PCL1*	0.7094	Not predicted
DBF2	*FAR1*	0.7003	Not predicted
SIC1	*CLN1*	0.8174	Predicted the other way round
PBS1	*MBP1*	0.7219	Not predicted
FAR1	*MET30*	0.873	Not predicted
CLB2	*DBF2*	0.7172	Predicted the other way round

APP threshold for decision. (The thresholds are listed in the threshold column of the tables.) A general observation is that we do not have high confidence about the inference results since high tp cannot be achieved at low fp. Since the VB-SEM algorithm has been tested with acceptable performance on simulated networks and the model has also been validated, this can very well indicate that the two data sets were not quite informative about the causal relationship between genes.

Table 6: The α data set.

APPs threshold	tp	tn	fp	fn
0.4	411	177	3247	9
0.5	58	3116	308	362
0.6	8	3405	19	412

Table 7: The CDC28 data set.

APPs threshold	tp	tn	fp	fn
0.4	405	163	3261	15
0.5	74	3019	405	346
0.6	14	3363	61	406

Data integration

In order to improvethe accuracy of the inference, we applied the Bayesian integration scheme described in Section 4 to combine the two data sets, trying to use information provided from both data sets to improve the inference confidence. The Bayesian integration includes two stages. In the first stage, the proposed VBSEM algorithm is run on the data set 1 that contains larger number of samples. In the second

Table 8: Links with higher APPs obtained based on the integrated data set.

From	To	APPs	Comparison with KEGG
*CLB*1	*CDC*20	0.7969	Predicted the other way round
*CLB*2	*CDC*5	0.6898	Predicted the other way round
*CDC*6	*CLB*6	0.7486	Predicted the other way round
*HSL*7	*CLB*1	0.6878	Predicted
*CDC*5	*CLB*1	0.7454	Predicted
*CLB*2	*CLB*1	0.6795	Predicted
*CLB*6	*HSL*1	0.7564	Not predicted
*RAD*17	*CLN*3	0.7324	Not predicted
*FAR*1	*SIC*1	0.7329	Predicted
*FAR*1	*MET*30	0.7742	Not predicted
*MET*30	*RAD*9	0.7534	Not predicted
*CDC*5	*CLB*2	0.7033	Predicted
*CLB*6	*CDC*45	0.6299	Predicted
*CLB*6	*Cln*1	0.6912	Predicted the other way round
*CLN*1	*CLN*3	0.8680	Predicted the other way round
*BUB*1	*ESP*1	0.6394	Predicted
*BUB*2	*CDC*5	0.6142	Predicted
*SIC*1	*CLN*1	0.6793	Predicted the other way round

stage, the APPs of the latent variables $\mathbf{b}_i$ obtained in the first stage are used as the priors in the VBSEM algorithm run on the second data set from [28]. In Figure 9, we plot the inferred network obtained from the integration process. We also performed bootstrap resampling in the integration process: we first obtained a sampled data set from the data set 1 and then we use its calculated APPs as the prior to integrate a bootstrap sampled data from set 2.

In Table 8, we present the links with the higher APPs inferred performing integration of the data sets. We made a comparison between these links and the ones shown in the KEGG pathway map again. As it can be seen, the proposed algorithm is able to predict many relationships. For instance, the link between CDC5 and CLB1 is predicted correctly by our algorithm with a posteriori probability of 0.7454. The weight associated to this connection is -0.1245, which is negative, and so there is a downregulation relationship confirmed in the KEGG pathway. We also observed improvements from integrating the two data sets. Regarding the link between CDC5 and CLB1, if we compare the result obtained from the integrated data set, with that shown in Table 4, we see that this relationship was not predicted when using the CDC28 data set 2. Even though this link was predicted by the α data set its APP is however lower and the weight is positive indicating an inconsistency with the KEGG map. The inconsistency has been fixed by data integration. As another example, the relationship between *HSL*7 and *CLB*1 was predicted based on the integrated data sets but it was not predicted from the CDC28 data set. This link was predicted when only the α data set was used but its APP is 0.6108, lower than the APP obtained performing integration. Similar phenomenon can be observed for the link between *FAR*1 to *SIC*1 again.

Table 9: Integrated data set.

APPs threshold	tp	tn	fp	fn
0.4	252	1655	1769	168
0.5	50	3175	249	370
0.6	17	3374	50	403

We also listed the statistics of the results when compared with the KEGG map in Table 9. We can see that when compared with Tables 6 and 7, data integration almost halved the fp at the thresholds 0.4 and 0.5 and also reduced the fp at 0.6. Meanwhile, tp increased. This implies the increased confidence on the results after data integration, which demonstrates the advantages of the Bayesian data integration.

Another way of looking at the benefits of the integration process is by examining the lower bound of the VBSEM. If the data integration process benefits the performance of the algorithm, we must see higher lower bound values than those of single data set. This happens because if the data contains more information after integration, the lower bound should be closer to the value it is approximating. In Figure 10, we plot the evolution of lower bound over the VBSEM iterations for each gene from the α data set, the CDC28 data set, and the integrated data sets. The increase of the lower bound, when the integrated data sets were used, supports the advantages of Bayesian data integration.

6. CONCLUSION

We investigated the DBNs modeling of cell cycle GRNs and the VBSEM learning of network topology. The proposed VBSEM solution is able to estimate the APPs of topology. We showed how the estimated APPs can be used in a Bayesian data integration strategy. The low complexity of the VBSEM algorithm shows its potential to work with large networks. We also showed how the bootstrap method can be used to obtain the confidence of the inferred networks. This approach has been approved very useful in the case of small data size, a common case in computational biology research.

APPENDICES

A. CONJUGATE PRIORS OF TOPOLOGY AND PARAMETERS

We choose the conjugate priors for topology and the parameters and they are

$$p(\mathbf{b}_i) = \mathcal{N}(\mathbf{b}_i \mid \boldsymbol{\mu}_0, \mathbf{C}_0), \tag{A.1}$$

$$p(\boldsymbol{\theta}_i) = p(\mathbf{w}_i, \sigma_i^2) = p(\mathbf{w}_i \mid \sigma_i^2)\, p(\sigma_i^2) = \mathcal{N}(\mathbf{w}_i \mid \boldsymbol{\mu}_{\mathbf{w}_i}, \sigma_i^2 \mathbf{I}_G)\, \mathcal{IG}\left(\frac{\gamma_0}{2}, \frac{\nu_0}{2}\right), \tag{A.2}$$

where $\boldsymbol{\mu}_0$ and $\mathbf{C}_0$ are the mean and the covariance of the prior probability density $p(\mathbf{b}_i)$. In general, $\boldsymbol{\mu}_{\mathbf{w}_i}$ and $\boldsymbol{\mu}_0$ are simply set as zero vectors, and meanwhile ν_0 and γ_0 are set equal to small positive real values. Moreover, covariance matrix $\mathbf{C}_0$

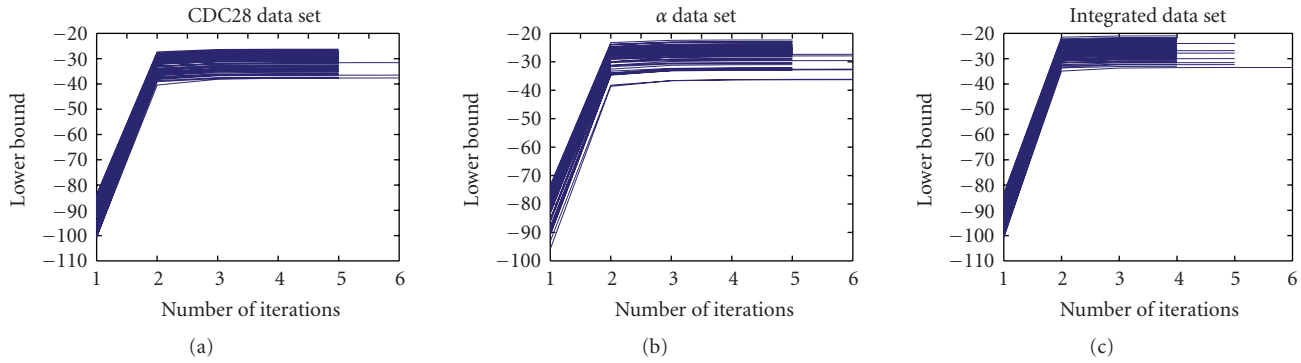

Figure 10: Evolution of the VBSEM lower bound.

needs to be checked carefully and is usually set as a diagonal matrix with a relatively large constant at each diagonal element.

These priors satisfy the conditions for conjugate exponential (CE) models [25]. For conjugate exponential models, formulae exist in [25] for solving analytically the integrals in the VBE and VBM steps.

B. DERIVATION OF VBE AND VBM STEPS

Let us first start with VBE step. Suppose that $q(\boldsymbol{\theta}_i)$ obtained in the previous VBM step follows a Gaussian-inverse-gamma distribution and has the expression (B.5). The VBE step calculates the approximation on the APPs of topology $p(\mathbf{b}_i)$. By applying the theorems of the CE model [25], $q(\mathbf{b}_i)$ can be shown to have the following expression:

$$q(\mathbf{b}_i) = \mathcal{N}(\mathbf{b}_i \mid \mathbf{m}_{\mathbf{b}_i}, \Sigma_{\mathbf{b}_i}), \tag{B.1}$$

where

$$\mathbf{m}_{\mathbf{b}_i} = \Sigma_{\mathbf{b}_i}(\mathbf{C}_0^{-1} + \mathbf{f}) \qquad \Sigma_{\mathbf{b}_i} = (\mathbf{C}_0^{-1} + \mathbf{D})^{-1}, \tag{B.2}$$

with

$$\begin{aligned} \mathbf{D} &= \mathbf{B} \otimes [(\mathbf{m}_{\mathbf{w}_i}\mathbf{m}_{\mathbf{w}_i})^T \langle \sigma_i^{-2} \rangle_{q(\theta_i)} + \mathbf{A}^{-1}], \\ \mathbf{f}^\top &= \mathbf{y}_i{}^\top \mathbf{R} \operatorname{diag}(\mathbf{m}_{\mathbf{w}_i}) \langle \sigma_i^{-2} \rangle_{q(\theta_i)}, \\ \mathbf{B} &= \mathbf{R}^\top \mathbf{R}, \\ \mathbf{A} &= \mathbf{I}_G + \mathbf{K}, \\ \mathbf{K} &= \mathbf{B} \otimes (\Sigma_{\mathbf{b}_i} + \mathbf{m}_{\mathbf{b}_i}\mathbf{m}_{\mathbf{b}_i}^T), \end{aligned} \tag{B.3}$$

and $\mathbf{R} = \mathbf{T}\mathbf{Y}^\top$, with

$$\mathbf{T} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \vdots \\ & 1 & 0 \end{pmatrix} \tag{B.4}$$

being an $N \times (N+1)$ matrix.

We now turn to the VBM step in which we compute $q(\boldsymbol{\theta}_i)$. Again, from the CE model and $q(\mathbf{b}_i)$ obtained in (B.1), we have

$$q(\boldsymbol{\theta}_i) = \mathcal{N}(\mathbf{w}_i \mid \mathbf{m}_{\mathbf{w}_i}, \Sigma_{\mathbf{w}_i}) \mathcal{IG}\left(\frac{\alpha}{2}, \frac{\beta}{2}\right), \tag{B.5}$$

where

$$\begin{aligned} \mathbf{m}_{\mathbf{w}_i} &= (\mathbf{I}_G + \mathbf{K})^{-1} (\mathbf{y}_i{}^T \mathbf{R} \mathbf{M}_x)^T, \\ \Sigma_{\mathbf{w}_i} &= \sigma_i^2 (\mathbf{I}_G + \mathbf{K})^{-1}, \\ \alpha &= N(\eta + 1) - G - 2, \\ \beta &= -c, \end{aligned} \tag{B.6}$$

with

$$\begin{aligned} \mathbf{M}_x &= \operatorname{diag}(\mathbf{m}_{\mathbf{b}_i}), \\ c &= \mathbf{y}_i{}^\top \mathbf{R} \mathbf{M}_x \mathbf{m}_{\mathbf{w}_i} - \mathbf{y}_i{}^\top \mathbf{y}_i - \nu_0, \end{aligned} \tag{B.7}$$

and η is a hyperparameter of the parameter prior $p(\boldsymbol{\theta}_i)$ based on CE models (A.2).

1. Computation of the lower bound $\mathcal{F}$

The convergence of the VBEM algorithm is tested using a lower bound of $\ln p(\mathbf{y}_i)$. In this paper, we use $\mathcal{F}$ to denote this lower bound and we calculate it using the newest $q(\mathbf{b}_i)$ and $q(\boldsymbol{\theta}_i)$ obtained in the iterative process. $\mathcal{F}$ can be written more succinctly using the definition of the KL divergence. Let us first review the definition of the KL divergence and then derive an analytical expression for $\mathcal{F}$.

The KL divergence measures the difference between two probability distributions and it is also termed relative entropy. Thus, using this definition we can write the difference between the real and the approximate distributions in the following way:

$$\begin{aligned} \mathrm{KL}[q(\mathbf{b}_i) || p(\mathbf{b}_i, \mathbf{y}_i \mid \boldsymbol{\theta}_i)] &= -\int d\mathbf{b}_i q(\mathbf{b}_i) \ln \frac{p(\mathbf{b}_i, \mathbf{y}_i \mid \boldsymbol{\theta}_i)}{q(\mathbf{b}_i)}, \\ \mathrm{KL}[q(\boldsymbol{\theta}_i) || p(\boldsymbol{\theta}_i)] &= -\int d\boldsymbol{\theta}_i \ln \frac{p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)}. \end{aligned} \tag{C.1}$$

And finally, the lower bound $\mathcal{F}$ can be written in terms of the previous definitions as

$$\begin{aligned}\mathcal{F} &= \int d\boldsymbol{\theta}_i q(\boldsymbol{\theta}_i)\left[\int d\mathbf{b}_i q(\mathbf{b}_i)\ln\frac{p(\mathbf{b}_i,\mathbf{y}_i\mid\boldsymbol{\theta}_i)}{q(\mathbf{b}_i)}+\ln\frac{p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)}\right]\\ &= -\int d\boldsymbol{\theta}_i q(\boldsymbol{\theta}_i)\,\mathrm{KL}\left[q(\mathbf{b}_i)||p(\mathbf{b}_i,\mathbf{y}_i\mid\boldsymbol{\theta}_i)\right]\\ &\quad -\mathrm{KL}\left[q(\boldsymbol{\theta}_i)||p(\boldsymbol{\theta}_i)\right].\end{aligned} \tag{C.2}$$

ACKNOWLEDGMENTS

Yufei Huang is supported by an NSF Grant CCF-0546345. Also M. Carmen Carrion Perez thanks MCyT under project TEC 2004-06096-C03-02/TCM forfunding.

REFERENCES

[1] H. Kitano, "Looking beyond that details: a rise in system-oriented approaches in genetics and molecular biology," *Current Genetics*, vol. 41, no. 1, pp. 1–10, 2002.

[2] P. D'haeseleer, S. Liang, and R. Somogyi, "Genetic network inference: from co-expression clustering to reverse engineering," *Bioinformatics*, vol. 16, no. 8, pp. 707–726, 2000.

[3] P. Brazhnik, A. de la Fuente, and P. Mendes, "Gene networks: how to put the function in genomics," *Trends in Biotechnology*, vol. 20, no. 11, pp. 467–472, 2002.

[4] N. Friedman, "Inferring cellular networks using probabilistic graphical models," *Science*, vol. 303, no. 5659, pp. 799–805, 2004.

[5] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, vol. 18, no. 2, pp. 261–274, 2002.

[6] X. Zhou, X. Wang, and E. R. Dougherty, "Construction of genomic networks using mutual-information clustering and reversible-jump Markov-chain-Monte-Carlo predictor design," *Signal Processing*, vol. 83, no. 4, pp. 745–761, 2003.

[7] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young, "Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks," in *Proceedings of the 6th Pacific Symposium on Biocomputing (PSB '01)*, pp. 422–433, The Big Island of Hawaii, Hawaii, USA, January 2001.

[8] E. J. Moler, D. C. Radisky, and I. S. Mian, "Integrating naive Bayes models and external knowledge to examine copper and iron homeostasis in S. cerevisiae," *Physiol Genomics*, vol. 4, no. 2, pp. 127–135, 2000.

[9] E. Segal, *Rich probabilistic models for genomic data*, Ph.D. thesis, Stanford University, Stanford, Calif, USA, 2004.

[10] H. de Jong, "Modeling and simulation of genetic regulatory systems: a literature review," *Journal of Computational Biology*, vol. 9, no. 1, pp. 67–103, 2002.

[11] Z. Bar-Joseph, "Analyzing time series gene expression data," *Bioinformatics*, vol. 20, no. 16, pp. 2493–2503, 2004.

[12] N. Simonis, S. J. Wodak, G. N. Cohen, and J. van Helden, "Combining pattern discovery and discriminant analysis to predict gene co-regulation," *Bioinformatics*, vol. 20, no. 15, pp. 2370–2379, 2004.

[13] K. Murphy and S. Mian, "Modelling gene expression data using dynamic Bayesian networks," Tech. Rep., Computer Science Division, University of California, Berkeley, Calif, USA, 1999.

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

[15] R. J. P. van Berlo, E. P. van Someren, and M. J. T. Reinders, "Studying the conditions for learning dynamic Bayesian networks to discover genetic regulatory networks," *Simulation*, vol. 79, no. 12, pp. 689–702, 2003.

[16] M. J. Beal, F. Falciani, Z. Ghahramani, C. Rangel, and D. L. Wild, "A Bayesian approach to reconstructing genetic regulatory networks with hidden factors," *Bioinformatics*, vol. 21, no. 3, pp. 349–356, 2005.

[17] B.-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. d'Alché-Buc, "Gene networks inference using dynamic Bayesian networks," *Bioinformatics*, vol. 19, supplement 2, pp. ii138–ii148, 2003.

[18] S. Y. Kim, S. Imoto, and S. Miyano, "Inferring gene networks from time series microarray data using dynamic Bayesian networks," *Briefings in Bioinformatics*, vol. 4, no. 3, pp. 228–235, 2003.

[19] F. Ferrazzi, R. Amici, P. Sebastiani, I. S. Kohane, M. F. Ramoni, and R. Bellazzi, "Can we use linear Gaussian networks to model dynamic interactions among genes? Results from a simulation study," in *Proceedings of IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS '06)*, pp. 13–14, College Station, Tex, USA, May 2006.

[20] X. Wang and H. V. Poor, *Wireless Communication Systems: Advanced Techniques for Signal Reception*, Prentice Hall PTR, Englewood Cliffs, NJ, USA, 2004.

[21] J. Wang, Y. Huang, M. Sanchez, Y. Wang, and J. Zhang, "Reverse engineering yeast gene regulatory networks using graphical models," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '06)*, vol. 2, pp. 1088–1091, Toulouse, France, May 2006.

[22] D. Husmeier, "Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks," *Bioinformatics*, vol. 19, no. 17, pp. 2271–2282, 2003.

[23] K. P. Murphy, *Dynamic Bayesian networks: representation, inference and learning*, Ph.D. thesis, University of California, Berkeley, Calif, USA, 2004.

[24] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1997.

[25] M. J. Beal, *Variational algorithms for approximate Bayesian inference*, Ph.D. thesis, The Gatsby Computational Neuroscience Unit, University College London, London, UK, May 2003.

[26] S. P. Brooks, "Markov chain Monte Carlo method and its application," *Journal of the Royal Statistical Society: Series D, The Statistician*, vol. 47, no. 1, pp. 69–100, 1998.

[27] P. T. Spellman, G. Sherlock, M. Q. Zhang, et al., "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization," *Molecular Biology of the Cell*, vol. 9, no. 12, pp. 3273–3297, 1998.

[28] R. J. Cho, M. J. Campbell, E. A. Winzeler, et al., "A genome-wide transcriptional analysis of the mitotic cell cycle," *Molecular Cell*, vol. 2, no. 1, pp. 65–73, 1998.

[29] B. Efron and R. Tibshirani, *An Introduction to Bootstrap*, Monographs on Statistics and Applied Probability, no. 57, Chapman & Hall, New York, NY, USA, 1993.

[30] S. N. Lahiri, *Resampling Methods for Dependent Data*, Springer, New York, NY, USA, 2003.

[31] "Kegg: Kyoto encyclopedia of genes and genomes," http://www.genome.jp/kegg/.