International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:8, 2008

Dynamic Bayesian Networks Modeling for Inferring Genetic Regulatory Networks by Search Strategy: Comparison between Greedy Hill Climbing and MCMC Methods

Huihai Wu, Xiaohui Liu

***Abstract*—**Using Dynamic Bayesian Networks (DBN) to model genetic regulatory networks from gene expression data is one of the major paradigms for inferring the interactions among genes. Averaging a collection of models for predicting network is desired, rather than relying on a single high scoring model. In this paper, two kinds of model searching approaches are compared, which are Greedy hill-climbing Search with Restarts (GSR) and Markov Chain Monte Carlo (MCMC) methods. The GSR is preferred in many papers, but there is no such comparison study about which one is better for DBN models. Different types of experiments have been carried out to try to give a benchmark test to these approaches. Our experimental results demonstrated that on average the MCMC methods outperform the GSR in accuracy of predicted network, and having the comparable performance in time efficiency. By proposing the different variations of MCMC and employing simulated annealing strategy, the MCMC methods become more efficient and stable. Apart from comparisons between these approaches, another objective of this study is to investigate the feasibility of using DBN modeling approaches for inferring gene networks from few snapshots of high dimensional gene profiles. Through synthetic data experiments as well as systematic data experiments, the experimental results revealed how the performances of these approaches can be influenced as the target gene network varies in the network size, data size, as well as system complexity.

***Keywords*—**Genetic regulatory network, Dynamic Bayesian network, GSR, MCMC.

I. Introduction

THe main task of molecular biology is to decipher the mechanism of diverse cellular processes which involve interactions among a range of biomolecules inside the cell. One of such processes is about how to the gene transcription be regulated [22]. The well-known central dogma of molecular biology tell us that DNA is transcribed into mRNA and then translated into protein, but people yet do not know the complete interactions among genes and proteins, which form a genetic regulatory network(GRN). In such network one gene can control (inhibit or activate) another genes' expression through its product proteins called transcriptional factors. Recently, microarray technology allow us to detect the gene expression level. This provides a snapshot of thousands of genes expression profiles simultaneously, and recently it is possible to make several consecutive such snapshots of microarray in a shot time scale, resulting in a gene expression time series data. From the analysis of such data, we are able to infer the causal interactions between the genes, and contribute to the reconstruction of GRNs.

Huihai Wu and Xiaohui Liu are with the School of Information Systems, Computing and Mathematics, Brunel University, London, UK, email: huihai.wu@brunel.ac.uk, xiaohui.liu@brunel.ac.uk

There are many methods have been developed to analyze the genetic regulatory network by time series data. Clustering techniques are well used for analyzing the gene interactions, their methods are focused on classify the genes that have similar expression patterns from a set of microarray data [2], [3], [10]. Clustering is proven to be very useful in discovering co-regulated relationship in genes, but do not be able to infer the underlying transcriptional networks. so some other systematic method have been proposed such as Boolean network [1], [24], differential equation models [20]. Dynamic Bayesian network is one of most popular method for modeling GRNs [25], [13], [18], which use probabilistic graphic model for modeling the interaction uncertainties between genes.

These modeling methods have different strength. Boolean network can logically model the function of regulations, and allow large regulatory networks to be analyzed in an efficient way by making strong simplifying assumptions on the structure and dynamics of a GRN system. Since the Boolean network is intrinsic rule based method, just has two states to represent the gene expression level, for this reason it is difficult to build model more precisely. While the Bayesian network can infer network structure statistically. Using dynamic Bayesian network structure learning algorithm, one can obtain a probabilistic graphical model capable of representing causal relationship among genes. The capability of handling uncertainty is the main strength for the DBN modeling, due to the inherent stochastic aspects of gene expression and the existing of measurement noise. Once having a model of Bayesian network, it is possible to infer either direct or indirect causation. Another main strength of DBN is the ability to model cyclic interactions among genes, since the control loop such as feedback loops are very important structures in biological networks, especially in GRNs.

Using DBN for modeling gene regulatory network is addressed by many papers, but one big problem in these papers is hard to estimate the real performance of algorithms, since it is impossible to predict a gene interaction that is not supported by the literature. In this paper we give a strict test for DBN modeling only using two kinds of artificial data: synthetic data from a predefined DBN model and systematic data from a predefined a linear dynamic system. Due to dimensionality

problem of gene expression time series data set, which means the number of genes is huge relative to the number of time slices. We show how limited the learning algorithms of DBN could be when the time series data size become smaller. two kinds of model searching strategies are used, one is greed hill-climbing search [9], [4], and the other is Markov Chain Monte Carlo (MCMC) methods [11], [17]. Some works [8], [4] argue that hill-climbing methods works better than MCMC methods upon Bayesian network framework, while it is not clear for DBN model. In [17], they applied MCMC methods to DBN successfully, and give rise to a basic approach for DBN model searching with MCMC methods. Based on their works, we further investigate overall performance for these two kinds of search strategies through the model averaging technique. The GRNs are characterized by highly looped dynamic systems and often exhibit modularity [31], where nodes are connected functionally or physically, which can be seen as relatively stable automatons. A linear dynamic system is used to simulate such scenario, by which we investigate how the performance of algorithms changes as the complexity of system increase.

The rest of the paper is organized as follows. In the next section, we give a detailed introduction on the DBN modeling approaches, including the principle of DBN modeling, scoring functions for DBN learning, model search strategies, and some different Markov chain mixing approaches have been proposed. In the section 3, we investigate the performances of DNB modeling methods, and present experiments results. Finally, we give a brief discussion and conclusion in Section 4.

II. Methodology

Bayesian Networks (BN) is a graphic model $G = (V, E)$, where nodes V represent random variables which correspond to gene expression levels, denoted as $\{X_1, X_2, \ldots, X_p\}$, and E indicate the dependence relationship between nodes which can be causal relationship of interactions among genes. The graphic structure should be Directed Acyclic Graph (DAG), such as $X_1 \rightarrow X_2$, here X_1 is called parents of X_1 and X_2 is called child of X_1. By the chain rule of probability and conditional independence between the nodes, the joint probability distribution of X can be calculated as: $P(X) = \prod_{i=1}^{p} P(X_i|\mathbf{Pa}(X_i))$, where $\mathbf{Pa}(X_i)$ is parents of node X_i, the $P(X_i|\mathbf{Pa}(X_i))$ for node i is called Conditional Probability Distribution (CPD), also called local distribution, this is a table for discrete nodes and density function for continuous nodes.

A. Dynamic Bayesian networks

Dynamic Bayesian Networks (DBN) can be regarded as an extension of BNs, which is designed for dynamic time series data. Consider a microarray time series variable $X \in \mathbf{R}^{n \times p}$, where n for number of time slices and p for number of genes, x_{ti} represent an observation for gene i at time t, then observation vector at the time t can be represent as vector $X_{(t)} = [X_{t1}, \ldots, X_{tp}]^T$, and ith gene at all time points can be represent as vector $X_i = [X_{1i}, \ldots, X_{ni}]^T$. DBN models assume time dependence in which directed arcs should flow forward in time. Normally, it is assumed to be first order Markov model, in which each gene is directly influenced only by the previous genes. Due to the time dependence, feedback loop networks can be easily modeled, Fig.1 shows a simple example where a three genes' cyclic network is unrolled to a acyclic DBN model. In DBN model every observation x_{ti} can be assigned as a individual random variable so that the joint probability distribution for DNB can be decomposed as follows:

$$P(X_{11}, \ldots, X_{np}) = \prod_{i=1}^{p} \prod_{t=1}^{n} P(X_{ti}|\mathbf{Pa}_{t-1,i}),$$

where $\mathbf{Pa}_{t-1,i}$ is parents of gene X_{ti}. The local distributions of genes at fist time slice with no parents can be seen as their prior distribution for genes. Also the CPD is called transition model which is assumed to be the same for every pair of neighboring time slices, representing the stationary dynamic process between genes. Since the causal relationship between genes always follows the time flow, hence there is no equivalent class problem, or different DBN models must represent different joint probability distributions.

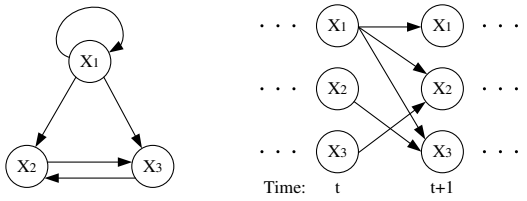

Fig. 1. Dynamic Bayesian network. Left figure is a simple gene network, where X_2 and X_3 form a multi-component loop control, and X_1 has a Auto-regulation. The left figure shows the corresponding Dynamic Bayesian network by unrolling the gene network followed by time points.

The main goal of modeling gene network is to find a model that best fits the data. To this end, one should learn the structure and parameters of DBNs from data. The task of learning a DBN can be stated as follows, Given a time series data set $D = \{X_{(1)}, \ldots, X_{(n)}\}$, which contain n instances of X in time, one should find a model $\mathcal{M} = (G, \Theta)$ that best fits D, where $\mathcal{M}$ is specified by the network structure G of DBN and corresponding parameters Θ of CPD families. By the bayes rule, the posterior distribution over model $\mathcal{M}$ is

$$P(\mathcal{M}|D) = \frac{P(\mathcal{M})P(D|\mathcal{M})}{P(D)}, \tag{1}$$

where the denominator $P(D) = \sum_G P(D|\mathcal{M})P(\mathcal{M})$ is a normalization factor which is independent of $\mathcal{M}$, therefore, taking logarithm, a scoring function for a model M can be built as follows,

$$S(\mathcal{M}) = \log P(D|\mathcal{M}) + \log P(\mathcal{M}), \tag{2}$$

where, $P(\mathcal{M})$ is prior for a model, which is used to panelize the complex models so as to avoid overfitting problem. $P(D|\mathcal{M})$ is called marginal likelihood over data D given $\mathcal{M}$, that is

$$P(D|\mathcal{M}) = \int P(D|\Theta, G)P(\Theta|G)d\Theta,$$

where $P(\Theta|G)$ is prior distribution for parameters. The objective for selecting the best model $\mathcal{M}$ can be reached by maximizing the marginal likelihood. The parameter prior also have the effect on penalizing the complex models. In order to obtain a closed form solution for the integration of marginal likelihood, normally one can choose the conjugate prior distribution, such as Dirichlet family for a discrete multinomial distribution, and Normal-Wishart family for a continual Gaussian distribution.

B. Scoring function

There are some assumptions that commonly are assumed [15], [14], [12], such as global parameter independence which says the parameters of each node in a BN are independent, and parameter modularity which says the same node having the same parents in distinct BNs has the same local distribution of parameters. By these assumptions the scoring function can be decomposed as $S(\mathcal{M}) = \sum_{i=1}^{p} S(\mathcal{M}_i)$, where $\mathcal{M}_i = (G_i, \theta_i)$ is the model for gene X_{ti}, in which G_i specify the parents $\mathbf{Pa}_i$ of X_i, $\theta_i \subset \Theta$. And the scoring function for $\mathcal{M}_i$ of a DBN can be expressed as

$$S(\mathcal{M}_i) = \log P(G_i) + \log \int \prod_{t=1}^{n} P(X_{ti}|\mathbf{Pa}_{t-1,i}, \theta_i) f(\theta_i|G_i) d\theta_i. \quad (3)$$

From the equation 2, as we can see, given the structure, the scoring of a model is determined by local distribution and its prior distribution of parameters. There are two types of transition model for DBNs, which are discrete model and continuous model. For the discrete model, the transition model is a CPT, in which the parameter at any time slices is set to be $P(X_i = x_k|\mathbf{Pa}_i = pa_j) = \theta_{ijk}$, and set r and q as total number of discrete state of X_i and $\mathbf{Pa}_i$ respectively. Hence the likelihood probability can be calculated as $P(X_i|\mathbf{Pa}_i, \theta_{ijk}) = \prod_{k=1}^{r} \theta_{ijk}^{N_{ijk}}$, which belongs to multinomial family, here N_{ijk} is sufficient statistics that is total count of observations when $X_i = x_k$ with $\mathbf{Pa}_i = pa_j$ occurs over all time slices. The conjugate prior distribution of parameters is Dirichlet distribution, for parameter θ_{ijk}, which can be expressed as $\Gamma(\alpha_{ij}) \prod_{k=1}^{r} (\theta_{ijk}^{\alpha_{ijk}-1} / \Gamma(\alpha_{ijk}))$, where α_{ijk} is its hyperparameters, $\alpha_{ij} = \sum_{k=1}^{r} \alpha_{ijk}$,and $\Gamma(\cdot)$ is Gamma function. Then, by the equation 3, the BDe criterion [7], [15] can be obtained as:

$$S_{BDe}(\mathcal{M}_i) = \log P(G_i) + \log \prod_{j=1}^{q} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})},$$

where sufficient staticstics $N_{ij} = \sum_{k=1}^{r} N_{ijk}$, together with hyperparameters α_{ij} determine the scoring of a DBN model.

For continuous model, we consider Gaussian networks, that is, the JPD is assumed multivariate Gaussian distribution, thereby the CPD are conditional Gaussian distributions. In [12], Dan and David developed a criterion, called BGe, here we give an concise interpretation. Suppose a Gaussian model $\mathcal{M} = (G_c, \Theta)$, where G_c is a complete DAG with no missing arcs, $\Theta = (\mu, B, V) = (\mu, \mathbf{W})$ is a set of parameters in which μ is unconditional mean of X, $B = \{b_1, \ldots, b_p\}$ is linear regression coefficients, such as $b_i = \{b_{i1}, \ldots, b_{ip}\}$ represent coefficient between gene X_i and its parents $\mathbf{Pa}_i$, $V = \{v_1, \ldots, v_p\}$ where v_i is conditional variance of node X_i given its parents $\mathbf{Pa}_i$, $\mathbf{W}$ equals inverse of covariance matrix of data, called precision matrix, that can be determined by parameters B and V. Then marginal likelihood for a complete Gaussian network can be expressed as:

$$f(X|G_c) = \int f(X|\mu, \mathbf{W}, G_c) f(\mu, \mathbf{W}|G_c) d\mu d\mathbf{W},$$

where, if $f(X|\mu, \mathbf{W}, G_c) = \mathcal{N}(\mu, \mathbf{W})$ is a normal distribution, and the conjugate prior distribution $f(\mu, \mathbf{W}|G_c)$ is a normal-Wishart distribution, then $f(X|G_c)$ must be an p dimensional multivariate t distribution. Based on the assumption of likelihood and prior modularity and global parameter independence, given any complete DAG G_c and data D, then the likelihood with arbitrary structure G can be computed as follows:

$$f(D|G) = \prod_{i=1}^{p} \frac{f(D(X_i, \mathbf{Pa}_i)|G_c)}{f(D(\mathbf{Pa}_i)|G_c)},$$

where $D(X_i, \mathbf{Pa}_i)$ represents the subset of data for gene X_i and its parents, and $D(\mathbf{Pa}_i)$ is in the same way, let a subset data of genes be D_Y, then the likelihood density have an closed form as follows:

$$\prod_{i=1}^{p} f(D_Y|G_c) = (2\pi)^{-np/2} \left(\frac{s_w}{s_w + n}\right)^{p/2} \frac{c(l, s_\mu)}{c(l, s_\mu + n)} |\mathbf{T}_0|^{s_\mu/2} |\mathbf{T}_n|^{-(s_\mu+n)/2}$$

where, s_μ is equivalent sample size for μ, s_w is equivalent sample size for $\mathbf{W}$, n is number of time slice, $c(\cdot)$ is a normalization constant given by $c(a, b) = [2^{ba/2} \pi^{a(a-1)/4} \prod_{i=1}^{a} \Gamma(\frac{b+1-i}{2})]^{-1}$. Given a network structure, parameter matrix $\mathbf{T}_0$ and $\mathbf{T}_n$ can be assessed by $\mathbf{W}$ and mean μ_0 from data which is assumed generating from a prior Gaussian network. For more details, please refer to [12].

In contrast to BDe criterion, the BGe is only able to modeling the linear relationship between genes. While the BGe criterion have the advantage that it use the low polynomial dimensionality of the parameter space of a multivariate normal distribution, whereas their discrete counterparts often require a parameter space that is exponential in the number of domain variables. In addition, discrete model is hugely affected by discretization methods.

C. Model selection

Having the decomposable scoring function, selecting a proper model for a gene network is a big challenge for learning DBNs. There are some reasons for that. First, the model space for searching is huge, even in DBNs, there is no cyclic graph problem which one should prevent in BNs. However the number of parents set for each node is still exponential in number of nodes n, that is, $\sum_{k=0}^{n} \binom{n}{k} = 2^n$, the optimization problem of identifying high scoring model is known to be NP-hard [6]. Second, the searching algorithm is not always

reach the best model, normally the model selected is local optimal one in terms of searching algorithm, such as greedy hill climbing approach. Third, due to the overfitting problem, the single high scoring model does not always agree with the best desired model, whereas such problem can be alleviated by penalizing the complex model and restricting the number of parents set, since gene networks prefer the sparse models.

For the above reasons, one can not rely on just one "best scoring" model, whereas the model averaging is a good strategy for model selecting. If two genes are highly correlated, then it is likely that the edge between them will appear in any high scoring models, such that the posterior probability of an edge between two genes can be evaluated. In this paper, we mainly focus on the search strategies by model averaging.

One traditional approach to search space for model searching is greedy hill-climbing search with restarts (GSR). At each restart, a random structure should be produced. Starting with this base structure, the mutation of structure will be made by a single edge addition or removal to a DBN model. The algorithm will check all the possible mutations of initial structure and choose the one with the highest scoring, and set it as new base structure, iterate this procedure until it reach a local maximum, and then store the structure and restart a new run. Finally, a set of models are sampled, the number of which is the same as the number of restarts. The algorithm can be depicted as in Algorithm 1.

Algorithm 1: Greedy Hill-Climbing Search with Restarts for DBN

Input: D (time series training data)
N_{res} (number of restarts)
Output: $\mathcal{M}_{out}$ (A collection of models with high scoring)
for $i = 1$ **to** N_{res} **do**
  Produce random structure $\mathcal{M}_0$
  repeat
    $\mathcal{M}_{best} \leftarrow \mathcal{M}_0$
    foreach *pair of nodes in DBN* **do**
      if *edge=0 (no connection between two nodes)* **then**
        $\mathcal{M}' \leftarrow AddEdge(\mathcal{M}_0)$
      else
        $\mathcal{M}' \leftarrow RemoveEdge(\mathcal{M}_0)$
      end
      if $Score(\mathcal{M}') > Score(\mathcal{M}_{best})$ **then**
        $\mathcal{M}_{best} \leftarrow \mathcal{M}'$
      end
    end
  until $\mathcal{M}_{best} == \mathcal{M}_0$ *(reach local maximum)*
  return $\mathcal{M}_{out} \leftarrow \mathcal{M}_{best}$
end

Another class of sampler we focus is Markov Chain Monte Carlo (MCMC) method [5], [17], a approach for generating samples from some high dimensional complex distribution of interest. The distribution we want draw here for DBN models is posterior $P(\mathcal{M}|D)$ from equation 1, and more importantly, for MCMC methods there is no need to compute the normalization constant $P(D)$. The mechanism of MCMC method is to construct a Markov chain process in which a new model $\mathcal{M}^*$ is generated only in terms of previous one $\mathcal{M}$, eventually a chain of models will be produced, that will convergence to the target distribution. A sufficient condition for such convergence is that the detailed balance equation holds for all models, that is $P(\mathcal{M}_i|\mathcal{M}_k)P(\mathcal{M}_k|D) = P(\mathcal{M}_k|\mathcal{M}_i)P(\mathcal{M}_i|D)$, where $P(\mathcal{M}_i|\mathcal{M}_k)$ is transition probability from $\mathcal{M}_k$ to $\mathcal{M}_i$.

One of most important MCMC methods is Metropolis-Hastings (MH) algorithm, which is based on acceptance-rejection sampling algorithm. For each run, the algorithm will sample a new candidate model from the jumping distribution $Q(\mathcal{M}^*, \mathcal{M})$, which is the probability of returning a new model $\mathcal{M}^*$ given a current model $\mathcal{M}$. Given the candidate model $\mathcal{M}^*$, the acceptance probability can be computed as $\alpha(\mathcal{M}, \mathcal{M}^*) = \min\left\{1, \frac{P(\mathcal{M}^*|D)Q(\mathcal{M}|\mathcal{M}^*)}{P(\mathcal{M}|D)Q(\mathcal{M}^*|\mathcal{M})}\right\}$, note that, since the ratio of two target distributions of models, the normalizing constant cancels out. Then the Markov chain will choose the current model with this acceptance probability. The algorithm is described as in Algorithm 2.

Algorithm 2: Metropolis-Hastings sampling algorithm for DBN

Input: D (time series training data)
N_{sam} (number of sampling)
Output: $\mathcal{M}_{out}$ (a sampling chain of models)
Produce initial model $\mathcal{M}_0$
for $i = 0$ **to** N_{sam} **do**
  sample a new model $\mathcal{M}^*$ from $Q(\mathcal{M}*|\mathcal{M})$
  compute
  $\alpha(\mathcal{M}_i, \mathcal{M}^*) \leftarrow \min\left\{1, \frac{P(\mathcal{M}^*|D)Q(\mathcal{M}_i|\mathcal{M}^*)}{P(\mathcal{M}_i|D)Q(\mathcal{M}^*|\mathcal{M}_i)}\right\}$
  sample u from $U_{(0,1)}$ (uniform distribution on $(0, 1)$)
  if $\alpha(\mathcal{M}_i, \mathcal{M}^*) > u$ **then**
    $\mathcal{M}_{i+1} \leftarrow \mathcal{M}^*$
  else
    $\mathcal{M}_{i+1} \leftarrow \mathcal{M}_i$
  end
  return $\mathcal{M}_{out} \leftarrow \mathcal{M}_{i+1}$
end

Due to the model space of a BN grows super-exponentially with the number of nodes, applying MCMC methods to learning BNs is always expensive computationally. In [11], they apply MCMC methods to get the posterior distribution given a order which is a sequence of nodes for a BN, then use a feature average technique to compute a posterior of this feature such as a particular choice of parents for a node. There are total number $n!$ of orders, and for each order, they need to sum over all possible structures consistent to this order to compute the scoring of the posterior, therefore cost of computation is very high even they restrict the number of fan-in parents of nodes and the scope in which the parents can be choose. But for DBN, as discussed above, edges are only allowed between neighboring time slices, there is no acyclic and equivalence problems, further the transition model is same for all intra-slices. Plus the restriction on the number of fan-ins, the cost of computation for DBN can be considerably alleviated.

Recall that the acceptance probability is determined by ratio of target distribution $P(\mathcal{M}|D)$ and the jumping distribution $Q(\mathcal{M}^*, \mathcal{M})$. The ratio of target distribution can calculated

from scoring function of equation 2. The jumping distribution for DNB is probability of jumping from one model to another one. The MH algorithm has no restriction on the jumping distribution, but different choice of jumping distribution do affect efficiency of the algorithm, such as how good a Markov chain to be mixed, a well mixing chain is the one that attempts to explore the whole model space equally, vice versa a chain is said to be poorly mixing. One approach for model jumping is random walks. As in GSR method, the new model is mutated based on the previous one by an elementary edge operation. In [17], they give the Hastings ratio as $Q(\mathcal{M}|\mathcal{M}^*)/Q(\mathcal{M}^*|\mathcal{M}) = N(\mathcal{M})/N(\mathcal{M}^*)$, where $N(\mathcal{M})$ is the size of the neighborhoods of $\mathcal{M}$, namely, the number of mutations that can be obtained from $\mathcal{M}$ by applying one of the elementary operations. Since adding or removing an edge in DBN is always acyclic networks, as a results we have $N(\mathcal{M}) = N(\mathcal{M}^*)$, so Hasting ratio always equals 1. Another approach of model jumping is independent chain sampling, in which the probability of jumping to model $\mathcal{M}^*$ is independent of the current model $\mathcal{M}$, so that the jumping distribution reduced as $Q(\mathcal{M}^*, \mathcal{M}) = R(\mathcal{M}^*)$. Thus the candidate model is simply drawn from a distribution of interest. In this paper uniform distribution is used for it, such that a model can can be chosen randomly and equally, so its Hasting ratio also is one.

The property of posterior distribution of DBN models is highly multimodaled and discontinuous as structure changes. For such target distribution, it is easy to get poorly mixing chain of samples, in which case the rejection rate could be very high. As discussed above, a suitable jumping distribution is one way to avoid this. On the other hand, the ratio of target distribution is also important for chain mixing, in our case, that is ratio of scoring of models. The higher scoring, then the higher acceptance probability a model will be. But the BDe scoring function is not well consistent with the model posterior distribution. The BDe criterion is sensitive to the number of parents of a node, here called fan-in size, different fan-in size has different scale of scoring. It is observed that for BDe criterion, the larger fan-in size a node has then the higher scoring could be, as a results the chain mixing is not stable as shown later in experiments. To avoid this, a simple way is to compensate fan-in size so as to make the nodes with lesser fan-ins have the same fan-in size as the one with lager fan-ins. We refer the compensated parents as to pseudo parents which have the while-noise data distribution, this approach making the ratio of scoring more effective for chain mixing process. In contrast to the BDe, the BGe criterion has no such problem, which can treat the scoring in the same scale.

Since the model distribution always has the multiple modes, it is possible that the MH sampling could be trapped in one mode, so that the algorithm will get highly rejection rate, and resulting the poorly mixing chain. To tackle this problem, we resort to simulated annealing (SA) method which is always used to find the global maximum of a complex function. The key ideal of SA is that a cooling schedule is used for Markov chain process so that the Markov chain will start with high acceptance rate then cool down in a rate tuned by a function $T(t)$ called temperature, the acceptance probability with temperature can be expressed as,

$$\alpha_{SA}(\mathcal{M}, \mathcal{M}^*) = \min\left\{1, \left(\frac{P(\mathcal{M}^*|D)}{P(\mathcal{M}|D)}\right)^{1/T(t)}\right\}, \quad (4)$$

from equation 4, it can be seen that the higher temperature, the higher acceptance probability could be, under this strategy, the model space can be well explored. In this work a simple temperature function is used as, $T(t) = \max\left(T_0^{1-t/n}, 1\right)$, by which, the Markov process start out at temperature T_0, and cool down to origin Markov chain process over n steps, and keep it for the rest of process.

III. Experiments

We compare the GSR and the MCMC approaches by experimental evaluation. We use two kinds of data set for simulations, one is synthetic data generated by certain predefined DBNs, and another one is generated by certain linear systems, we called it systematic data. For synthetic data, BDe criterion will be used, and BGe criterion is used for systematic data. The experiments are also done on the different network sizes, which is the number of nodes in a network.

After obtaining a collection of model samples from GSR or MCMC algorithms, we can evaluate the posterior of the edges that is proportional to the number of the edge in model samples. By set different threshold for edge posterior, we get a set of edges. Comparing this set of edges with true networks, then we can count the number of true edges (True Positive) denoted as TP, and the number of false edges (False Positive) denoted as FP. From the true network, the total number of edges NTP and the total number of true negative edges NTN can be calculated. Then we define true-sensitivity as TP/NTP which indicate how many true edges can be learned, and define false-sensitivity as FP/NTN which indicate how many edges are learned wrongly. For each set of edges with a threshold, we can get its true-sensitivity and false-sensitivity. Plotting true-sensitivities against the false-sensitivity, we got receiver operator characteristics (ROC) curves which give an overall evaluation for the algorithms. By integrating the ROC curve, we obtain the area under the curve (AUC), with larger scores indicating a better overall performance. It is desired that algorithms have higher true-sensitivity with lower false-sensitivity, and the ROC curves rightly indicate such performance. The ROC curves also qualitatively show how much cost should be paid for a desired recovery rate of true edges. It should be noticed that a recovered true edge with a high posterior probability can not simply ensure a real true edge when the false sensitivity is as high as the true sensitivity.

A. Experiments on synthetic data

We carried out the different types of experiments in terms of different network size and data size. The synthetic data is sampled from a predefined DBN, which is also randomly selected under certain structure and parameters. We use three types of DBN structures for data sampling, in which the network size is set as 10 nodes and 20 nodes respectively, and each node have the fan-in size of 3 or 4. The local distribution for DBNs is set as discrete trinomial distribution with

three discrete states representing the gene expression level of underexpressed, baseline, and overexpressed. The parameters of local distribution were generated randomly with a parameter value distribution that is close to being deterministic. The data size we choose is 100 and 20 respectively, so that the dimensionality problem can be examined, since so far, the microarray time course experiment is normally less than 20 points of time. The purpose of experimental settings is to try to reveal the strengths and weaknesses of approaches.

For synthetic datasets, BDe criterion is used for DBN learning, in which the model priors is set as uninformative, that is uniform distribution. We mainly test two types of model searching approaches which are GSR and MCMC. In GSR, a 50 restarts are used for model selecting. For MCMC methods, as mentioned above, there are three different variations by different jumping approaches, that is, random jumping approach denoted as MCMC-1, random jumping with fan-in compensation denoted as MCMC-2, and independent jumping denoted as MCMC-3. In Markov chain, at burning phase, a 2,000 runs of chain is thrown out, then 20,000 runs are used for sampling phase in which we choose a model with interval of 10 samples. We test these methods within 3 types of predefined DBN models, and demonstrate their performance using model averaging method. The results are averaged by repeating 3 times for each method. The ROCs are shown in Fig.2, Fig.3, and Fig.4, where the dashed diagonal line is the expected ROC curve of a random model selecting which used as a reference line. To effectively compared these approaches, here the values of AUC are computed as ratio: $AUCr = 100 \times (AUC - 0.5)/0.5$. The Table I,II,III show the performances: AUCr and time spent by algorithms which measured by seconds.

As Fig.2 shows, all methods perform perfect when data size is 100, and all dropped nearly the same amount of AUC when data size is reduced to 20, where all method reached about 0.8 true sensitivity against 0.3 false sensitivity. MCMC-1 and MCMC-3 seams to be slightly better than others from ROC curves. We observed that the performance of GSR improve very little when the number of restarts increase, since the models searched by GSR always fell into the same vicinity of a local optimal one.

The results is changed much just when adding one more fan-in parent for all nodes. From Fig.3, we can see that the ROC of MCMC-1 occurs abnormality, it nearly can not distinguish the true edges for data size of 100, but return normal for data size of 20. Due to the reason discussed in section II, the MCMC-1 always trapped in some structures especially when fan-in size is larger than 3. Since the correlation between nodes learned from data of 20 cases is more uncertain than that from data of 100 cases. The data with more cases can decrease the influence of such effect. Using a pseudo parent to compensate the fan-in size can fix this problem. As shown in Fig.3, for data size of 100, the ROC of GSR and MCMC-3 is dropped slightly, while MCMC-2 is still can recover all true edges without a false edge.

When the network size enlarged to 20 nodes, as shown in Fig.4, the performance is further degraded. In particular, they almost loose the discrimination for data size of 20. While, for

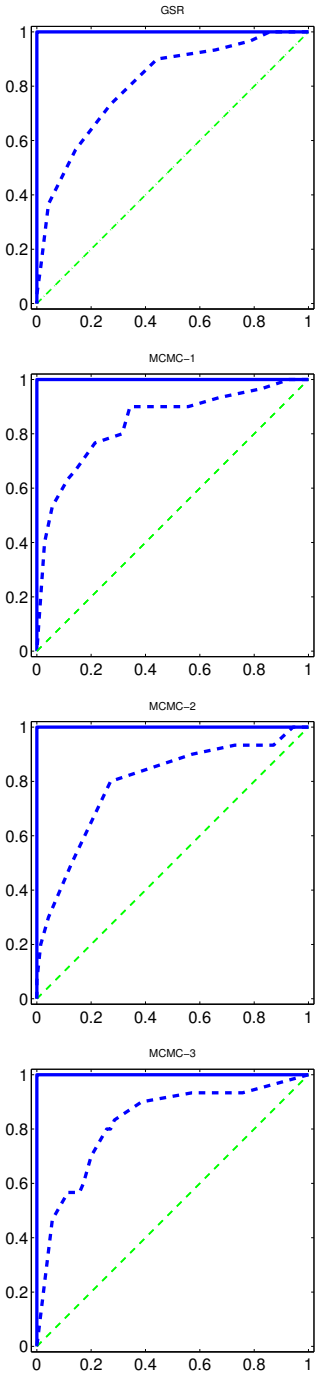

Fig. 2. ROC curves for synthetic data with network size 10 and fan-in size 3 of all nodes. Each subfigure of ROC is associated with a method denoted in title, where the solid line is correspond to the data size of 100, and dashed line correspond to the data size of 20.

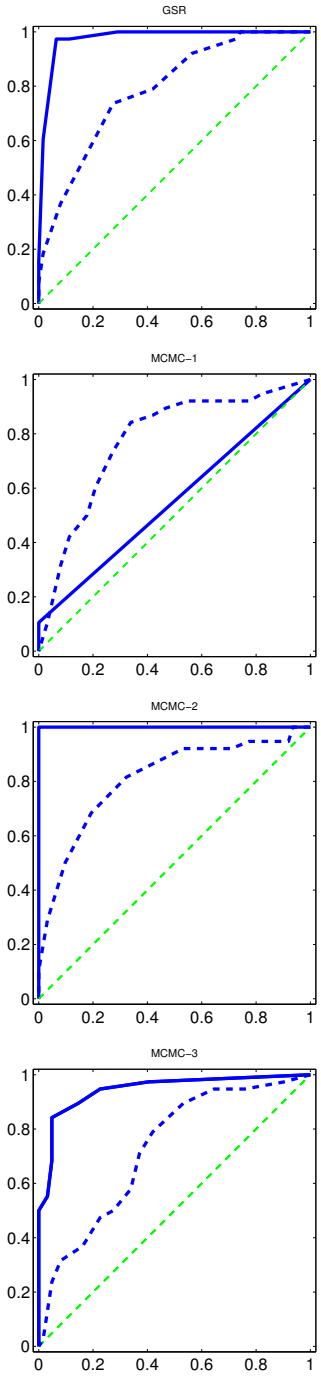

Fig. 3. ROC curves for synthetic data with network size 10 and fan-in size 4 of all nodes. Each subfigure of ROC is associated with a method denoted in title, where the solid line is correspond to the data size of 100, and dashed line correspond to the data size of 20.

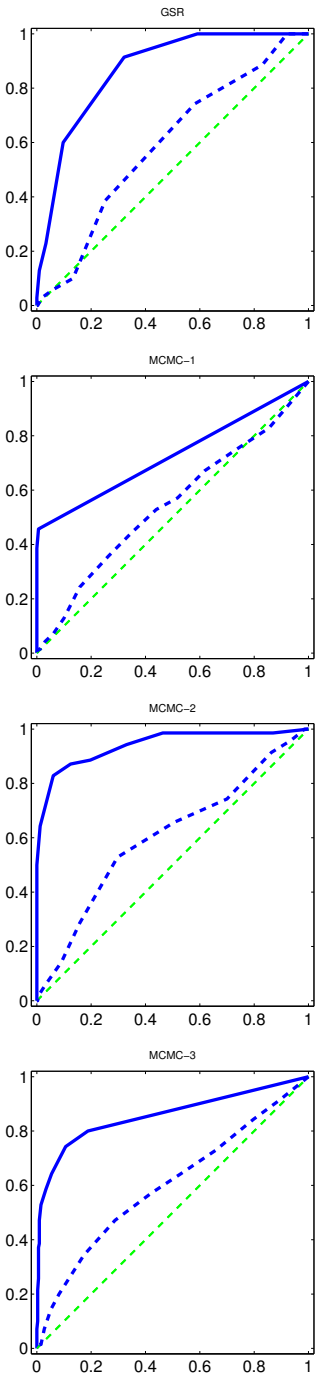

Fig. 4. ROC curves for synthetic data with network size 20, and half in fan-in size 3 and half in fan-in size 4. Each subfigure of ROC is associated with a method denoted in title, where the solid line is correspond to the data size of 100.

Data size	100		20	
	AUCr	Time(s)	AUCr	Time(s)
GSR	100	152	62.1	75
MCMC-1	100	263	68.3	158
MCMC-2	100	263	59	158
MCMC-3	100	274	65	201

TABLE I
RESULTS FOR SYNTHETIC DATA SET WITH NETWORK SIZE OF 10 AND FAN-IN SIZE OF 3.

Data size	100		20	
	AUCr	Time(s)	AUCr	Time(s)
GSR	95.25	124	57.7	93
MCMC-1	10.53	240	55.3	118
MCMC-2	100	240	61.9	156
MCMC-3	88.92	269	44.9	206

TABLE II
RESULTS FOR SYNTHETIC DATA SET WITH NETWORK SIZE OF 10 AND FAN-IN SIZE OF 4.

Data size	100		20	
	AUCr	Time(s)	AUCr	Time(s)
GSR	73.68	675	18	666
MCMC-1	45.34	347	8.6	118
MCMC-2	88.14	258	20.8	137
MCMC-3	71.6	381	19.9	307

TABLE III
RESULTS FOR SYNTHETIC DATA SET WITH NETWORK SIZE OF 20 AND FAN-IN SIZE OF 3 IN HALF AND 4 IN HALF.

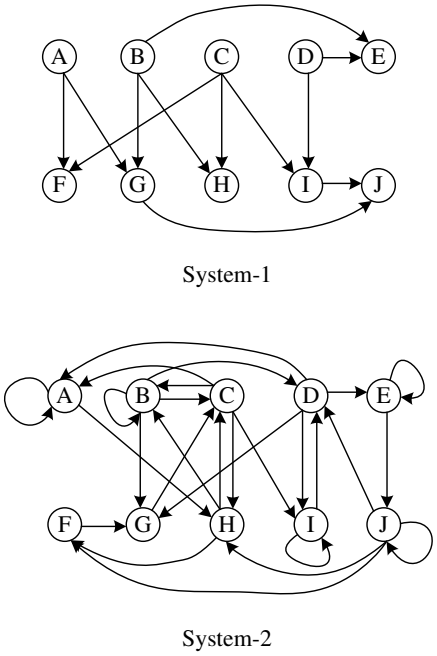

Fig. 5. Linear dynamic system. System-1 shows a simple system with no loops. System-2 shows a highly looped system.

data size of 100, apart from the MCMC-1, other three methods dropped not so much, in which the MCMC-2 has the best ROC. The time spent by MCMC methods are nearly identical, GSR has the half time of MCMC methods in network size of 10, while in network size of 20, the time spent by GSR is nearly twice more than that by MCMC methods.

B. Experiments on systematic data

From the synthetic data experiments, we can have an overview of performance of different approaches, especially how the performance changes when varying the scenarios with the network size and data size. However, the microarray time series data is generated from gene networks which is associated to a complex dynamic system process, while the synthetic data sets sampled from a predefined DBN that strictly are not time series of a system. So it is necessary to know how system complexity influences the performance of modeling approaches. In this group of experiments, we focus on the linear system, in which the expression of a gene is linearly regulated by others. The linear dynamic system model can be represent as $X_{t+1} = AX_t + BY_t + \omega_t$, where A is system transition matrix which represent the strength of linear correlation among genes X_t. Y_t is input and B is input to state matrix, and ω_t is system noise, here we do not consider noise model. We test two systems with different degree of complexity, as shown in Fig.5, where system-1 has no any loops with just fan-in size of 2, while system-2 is a highly looped stable autonomous system.

For the system-1, the nodes $A \sim D$ have random discrete signal as their input signals, where we choose 20 and 10 consecutive time points for training datasets respectively. Since system-2 is an autonomous system, given an initial perturbation value, then a continuous time series data can be generated by the evolution of system process. We sample 100 and 20 time points respectively for two training data. It is observed that such time point sampling should be within the time scale between the system stimulated by external perturbation and system falling into equilibrium. And the sampling interval should not be too short or too long, in principle, the sampling interval should be proportional to the data size so as to extract as much information as this data size could contain. In this group of experiments, we use BGe criterion for DBN learning. Apart from GSR, another three variations of MCMC methods are used for model searching, the first one use random jumping approach denoted as MCMC-A; the second one utilize simulated annealing strategy on MCMC-A called MCMC-B; the third one combined independent jumping approach with simulated annealing strategy denoted as MCMC-C. The number of runs for GSR and MCMC methods is the same as synthetic data experiments. The results are demonstrated as Fig.6, Fig.7, and Table IV,IV.

As we can see From the Fig.6, the MCMC-A works very well for such simple scenario, and evidently outperforms the GSR method. But for highly looped system, the situations are changed dramatically. As shown in Fig.7, the performance of GSR and MCMC-A dropped considerably, while by the SA strategy MCMC-B and MCMC-C give an substantial increase

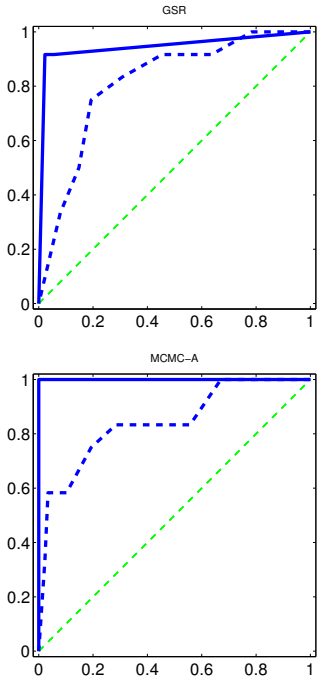

Fig. 6. ROC curves for systematic data with network size 10 and fan-in size 2 as shown in fig5. Each subfigure of ROC is associated with a method denoted in title, where the solid line is correspond to the data size of 20, and dashed line correspond to the data size of 10.

Data size	20		10	
	AUCr	Time(s)	AUCr	Time(s)
GSR	89.11	107	62.7	70
MCMC-A	100	177	68.5	101

TABLE IV
RESULTS FOR SYSTEMATIC DATA SET COMES FROM SYSTEM-1 WITH NETWORK SIZE OF 10 AND FAN-IN SIZE OF 2.

in performance. Especially the MCMC-C have the best overall performance among them, for data size of 100, it recovered the 75 percent true edges by paying the price of 5 percent false edges. Here, the time spent by MCMC methods is not much more than by GSR, since the BGe scoring is much faster than BDe scoring.

IV. Discussion and Conclusion

In this study, based on the DBN statistical modeling approach, we investigated the different model searching ap-

Data size	100		20	
	AUCr	Time(s)	AUCr	Time(s)
GSR	54.29	128	30.7	74
MCMC-A	53.58	147	23.7	115
MCMC-B	59.61	162	41.4	136
MCMC-C	63.52	256	40.8	209

TABLE V
RESULTS FOR SYSTEMATIC DATA SET COMES FROM SYSTEM-2 WITH NETWORK SIZE OF 10 AND FAN-IN SIZE OF 2 AND 3.

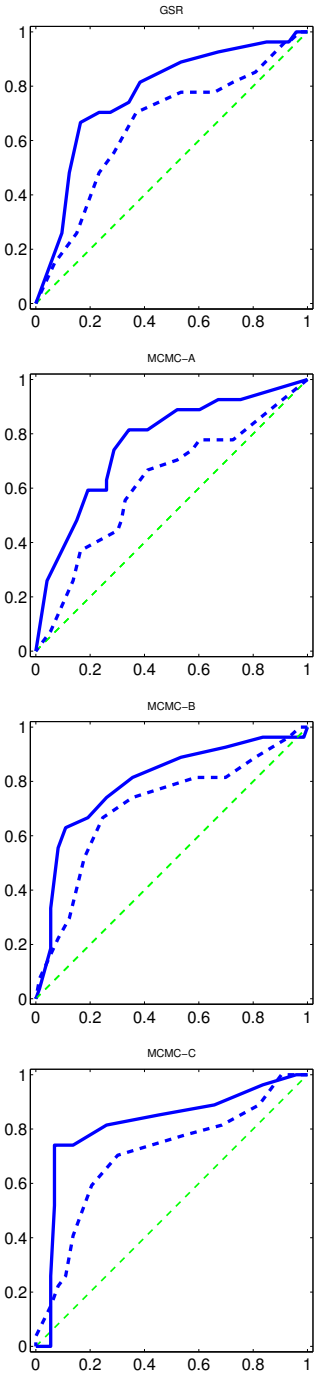

Fig. 7. ROC curves for systematic data with network size 10 and fan-in size 2 or 3 as shown in fig5. Each subfigure of ROC is associated with a method denoted in title, where the solid line is correspond to the data size of 20, and dashed line correspond to the data size of 10.

proaches involving GSR and MCMC methods. To give a meaningful evaluation, we test them under different scenarios using model averaging technique. On average, the results demonstrated that the MCMC methods basically outperform the GSR. Due to the shortcoming of BDe criterion, the MCMC-1, which is used in some papers, shows unstable performance, but through compensating a pseudo parent, surprisingly, it reached the best performance in synthetic data experiments. The performance of MCMC-3 is not as good as MCMC-2, but comparable to GSR. As second group of experiments shows, the SA strategy makes the MCMC methods more effective. It is because that SA spreads the searching scope of model space, thus avoiding model trapping, and eventually resulting in a well mixed chain for model averaging. That is why the MCMC-B and MCMC-C got a considerable gain in performance, and with independent jumping approach the MCMC-C made further improvement in performance. The time spent by all methods is sensitive to the restricted number of parents of a node, with lager this number, more time will be needed. It is observed that the GSR would need even more time to compete the MCMC methods, when the number of nodes increases. MCMC methods could get better time efficiency, as long as the Markov chain mixing well. In addition, the BGe criterion is much time efficient than the BDe criterion, since the lesser parameters are needed in BGe than that in BDe.

The dimensionality problem of microarray time series plays a fundamental role in genetic network modeling. The basic question is how many microarray experiments must be taken so as to insure the recovered gene networks is reliable. For learning a Bayesian network with binary nodes, a sufficient number of samples should be $O(n^2 \log n^2 \log n^{k+1})$ [16], where n is the total number of genes in the true network and k is the maximum fan-in size of the model. Unfortunately, our experiments revealed such serious consequences of dimensionality problem. From our experiments of synthetic data, take an example, for a gene network of 20 genes with up to 4 parents, given 20 time points data, all of approaches we test here almost cannot distinguish any interaction reliably. Moreover, our all experiments were noiseless, while real microarray data is full of noise, which need to be properly preprocessed involving primary expression signal filtering, normalization, and discretization. Also the discretization has strong influence on the performance of modeling approaches, since loss of information can be varying as using different discretization methods.

More importantly, the actual gene networks are characteristic of highly looped and highly modular dynamic systems, so another key question is how the system complexity influences the DBN modeling approaches. Unfortunately again, as our experiments of systematic data reveals, such influence could be dramatic. One of reasons for that we observed is that time series generated from a looped system often become more periodic and similar. Such phenomena is often found in real microarray time series. As a consequence the scoring function of DBNs can not distinguish them well, since essentially linear correlation is not necessary imply causation, such as a common regulator for two genes. In addition, in real GRN system, there are lots of other factors could affect the gene expression levels, such as levels of regulatory proteins, and the effects of mRNA and protein degradation. A linear state-space modeling can be used for inferring interactions among genes with hidden states [28], [27]. Another real issue is that the response between two genes actually has some time delay and not uniform[29], such delays could further reduced the accuracy of predicted gene networks [21].

As these serious issues mentioned above, it is almost impossible for DBN modeling approaches to reliably recover the gene networks by microarray data alone. However, it is proved that incorporating the relevant biological prior knowledge to the modeling approaches could effectively increase the accuracy of predicting [19], [23], [26], [30]. For example, in [19], biological knowledge they used include protein-protein and protein-DNA interactions, sequences of the binding site of the genes controlled by transcription regulators, and even biological literature. By adding biological knowledge into their Bayesian network approaches, they succeeded in extracting more information from microarray data and estimating the gene network more accurately. Recently, much attention has been paid in how to incorporate different kinds of biological knowledge more efficiently upon Bayesian network modeling framework [30], which is also our main concerns for our future works.

REFERENCES

[1] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. *Pac. Symp. Biocomput*, page 17C28, 1999.

[2] U. Alon, N. Barkai, D. Notterman, Ybarra S.and Mack D. Gish, K., and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Nat. Acad. Sci. USA*, 96:6745C6750, 1999.

[3] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6:281C297, 1999.

[4] Xue-wen Chen, Gopalakrishna Anantha, and Xinkun Wang. An effective structure learning method for constructing gene networks. *Bioinformatics*, 22(11):1367–1374, 2006.

[5] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *Amer. Statist.*, 49:327–335, 1995.

[6] David Maxwell Chickering, David Heckerman, and Christopher Meek. Large-sample learning of bayesian networks is np-hard. *J. Mach. Learn. Res.*, 5:1287–1330, 2004.

[7] F. Cooper and E. H. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[8] Honghua Dai, Gang Li, and Yiqing Tu. An empirical study of encoding schemes and search strategies in discovering causal networks. In *Machine Learning: ECML 2002: 13th European Conference on Machine Learning, Helsinki, Finland, August 19-23, 2002. Proceedings*.

[9] LAWRENCE A. DAVID and CHRIS H. WIGGINS. Benchmarking of Dynamic Bayesian Networks Inferred from Stochastic Time-Series Data. *Ann NY Acad Sci*, 1115(1):90–101, 2007.

[10] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genomewide expression patterns. *Proc. Nat. Acad. Sci. USA*, 95(14863C14868), 1998.

[11] N. Friedman and D. Koller. Being bayesian about network structure: A bayesian approach to structure discovery in bayesian networks. 2001.

[12] Dan Geiger and David Heckerman. Learning gaussian networks. (MSR-TR-94-10), 1994.

[13] 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. *Pac. Symp. Biocomput.*, 6:422–433, 2001.

[14] David Heckerman. Bayesian networks for data mining. *Data Min. Knowl. Discov.*, 1(1):79–119, 1997.

[15] David Heckerman, Dan Geiger, and David M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.*, 20(3):197–243, 1995.
[16] K. Hoffgen. Learning and robust learning of product distributions. *in:Sixth Annual Workshop on Computational Learning Theory*, pages 77–83, 1993.
[17] Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–2282, 2003.
[18] S. Imoto, T. Goto, and S. Miyano. Estimation of genetic networks and functional structures between genes by using bayesian networks and nonparametric regression. *Pac. Symp. Biocomput.*, 7:175–186, 2002.
[19] Seiya Imoto, Tomoyuki Higuchi, Takao Goto, Kousuke Tashiro, Satoru Kuhara, and Satoru Miyano. Combining microarrays and biological knowledge for estimating gene networks via bayesian networks. In *CSB '03: Proceedings of the IEEE Computer Society Conference on Bioinformatics*, page 104, 2003.
[20] B. Kholodenko, A. Kiyatkin, F. Bruggeman, E. Sontag, H. Westerhoff, and J. Hoek. Untangling the wires: a strategy to trace functional interactions in signaling and gene networks. *Proc. Natl. Acad. Sci.*, 99:12841C12846, 2002.
[21] Tie-Fei Liu, Wing-Kin Sung, and Ankush Mittal. Learning multi-time delay gene network using bayesian network framework. In *ICTAI '04: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04)*, pages 640–645, 2004.
[22] D.J. Lockhart and E.A. Winzeler. Genomics, gene expression and dna arrays. *Nature*, 405:827–836, 2000.
[23] N. Nariai, S. Kim, S. Imoto, and S. Miyano. Using protein-protein interactions for refining gene networks estimated from microarray data by bayesian networks. *Pac Symp Biocomput*, pages 336–347, 2004.
[24] D. Peer, A. Regev, G. Elidan, and Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17:S215CS224, 2001.
[25] D. Peer, A. Regev, G. Elidan, and Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(S215CS224), 2001.
[26] a Phillip P. Le, a Amit Bahl, and Lyle H. Ungar. Using prior knowledge to improve genetic network reconstruction from microarray data. *In Silico Biology*, 4(0027), 2004.
[27] Claudia Rangel, John Angus, Zoubin Ghahramani, Maria Lioumi, Elizabeth Sotheran, Alessia Gaiba, David L. Wild, and Francesco Falciani. Modeling T-cell activation using gene expression profiling and state-space models. *Bioinformatics*, 20(9):1361–1372, 2004.
[28] Claudia Rangel, David L. Wild, and Francesco Falciani. Modelling biological responses using gene expression profiling and linear dynamical systems. 2001.
[29] Nitzan Rosenfeld and Uri Alon. Response delays and the structure of transcription networks. *Journal of Molecular Biology*, 329.
[30] Adriano V. Werhli and Dirk 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*, 6.
[31] Z.Bar-Joseph, G.K.Gerber, T.I.Lee, N.J.Rinaldi, J.Y.Yoo, F.Robert, D.B.Gordon, E.Fraenkel, T.S.Jaakkola, and R.A.Young. Computational discovery of gene modules and regulatory networks. *Nat. Biotechnol*, 21:1337–1342, 2002.