



# Construction of genomic networks using mutual-information clustering and reversible-jump Markov-chain-Monte-Carlo predictor design

Xiaobo Zhou<sup>a</sup>, Xiaodong Wang<sup>b</sup>, Edward R. Dougherty<sup>a,c,\*</sup>

<sup>a</sup>Department of Electrical Engineering, Texas A&M University, 3128 TAMU, College Station, TX 77843-3128, USA

<sup>b</sup>Department of Electrical Engineering, Columbia University, New York, NY 10027, USA

<sup>c</sup>Department of Pathology, University of Texas M.D. Anderson Cancer Center, Houston, TX 77030, USA

Received 10 March 2002; received in revised form 22 August 2002

## Abstract

We consider the problem of constructing gene regulatory networks from expression data using the probabilistic-Boolean-network paradigm. We propose to construct the networks according to the following stages. Firstly, we determine the number of possible parent gene sets and the input sets of gene variables corresponding to each gene, and this is done by using a novel clustering technique based on mutual information minimization. Simulated annealing is employed to solve the optimization problem. After such initial gene clustering, we restrict our attention to the class of different functions from the possible parent gene sets to each target gene. Secondly, each function is then modeled by a perceptron consisting of a linear term and a nonlinear term. A reversible-jump Markov-chain-Monte-Carlo (MCMC) technique is used to calculate the model order and the parameters. Finally, the coefficient of determination (CoD) is employed to compute the probability of selecting different predictors for each gene. To test this approach for constructing gene regulatory networks, we have carried out computational experiments using data from known gene response pathways including ionizing radiation and downstream targets of inactivating gene mutations.

© 2002 Elsevier Science B.V. All rights reserved.

*Keywords:* Gene microarray; Probabilistic Boolean network; Gene regulatory network; Clustering; Mutual information; Simulated annealing; Markov chain Monte Carlo technique; Coefficient of determination

## 1. Introduction

To understand the nature of cellular function, it is necessary to study the behavior of genes in a holistic rather than in an individual manner because

the expressions and activities of genes are not isolated or independent of each other [14]. Mathematical and computational methods are being developed in order to construct formal models of genetic interactions. There have been a number of attempts to model gene regulatory networks, including linear models [26,45], Bayesian networks [20,27,39,40], neural networks [46], differential equations [38], and models including stochastic components on the molecular level [2,11,28,43]. In general, gene expression time

\* Corresponding author. Department of Electrical Engineering, Texas A&M University, 3128 TAMU, College Station, TX 77843-3128, USA. Tel.: +1-409-845-7441; fax: +1-409-845-6259.  
E-mail address: [edward@ee.tamu.edu](mailto:edward@ee.tamu.edu) (E.R. Dougherty).

trajectories can be modelled as random functions of time [17]. The model system that has received the most attention may be the Boolean network originally introduced by Kauffman [22,34,35]. There are many references applying this kind of network to genomic analysis, such as [2,3,25,31,42]. The recent paper [41] studies probabilistic Boolean networks (PBNs), which provide insight and a conceptual framework for an integrative view of genetic function and regulation. How to build PBNs from gene microarray data remains an open problem. We consider the problem in a more general framework by not restricting network functions to be binary valued.

The advantage of PBNs is that they naturally allow one to incorporate prior biological knowledge [41]. Thus, if certain regulatory relationships are known to exist, then the Boolean-function classes corresponding to the genes in question can be constructed to reflect this prior knowledge. In this paper, we first investigate the prior biological knowledge via an unsupervised clustering algorithm by considering dependence and independence in gene microarray data. Clustering has been used in a number of studies to obtain a global, unsupervised perspective on the similarity of expression profiles [6,7,12,16,19,33]. Pertinent to our approach is a study based on computing the mutual information for all pairs of genes and then choosing a threshold of the mutual information to create clusters of genes encompassing those with mutual information higher than the threshold [8]. Similar treatments are found in [25,21,20]. These works are based on pair-wise mutual information, and thus essentially only explore the marginal distributions of the multi-dimensional data. Here our clustering strategy is based on minimizing the mutual information of the variables among clusters, and hence it fully explores the underlying *joint* probability distribution of the data. Upon completion of the clustering, we can restrict our attention to constructing network functions for a gene in terms of the other genes within its information-based cluster.

Since the current data revealing the structure of the transcriptional regulation process is both limited and noisy [10], the functions should not be overly complex. We use a perceptron model consisting of a linear and nonlinear component to estimate the predictors (functions) for each gene. A reversible-jump Markov-chain-Monte-Carlo (MCMC) method is adopted to perform the necessary computations. The

coefficient of determination (CoD) [18] is employed to select the different predictors probabilistically.

As the first step in constructing a gene regulatory network, the complexity of the expression data is reduced by thresholding changes in transcript level, for instance, into ternary expression data:  $-1$  (down-regulated),  $+1$  (up-regulated), or  $0$  (invariant). When using multiple microarrays, the absolute signal intensities vary extensively due to both the process of preparing and printing the EST elements [9], and the process of preparing and labelling the cDNA representations of the RNA pools. This problem is solved via internal standardization. An algorithm first calibrates the data internally to each microarray and statistically determines whether the data justify the conclusion that expression is down-regulated or up-regulated [9]. We then build a network using the clustering and prediction methodologies.

The paper is organized as follows. Section 2 describes the main problem and our proposed approach. In Section 3, clustering analysis via mutual information minimization is developed. In Section 4, predictors used for gene regulatory networks are estimated using a reversible jump MCMC method. In Section 5, the CoD is used for selecting predictors probabilistically. Section 6 provides experimental results and analysis. Section 7 gives conclusions.

## 2. Construction of probabilistic gene regulatory networks

### 2.1. Probabilistic gene regulatory networks

A deterministic gene regulatory network  $G(V, F)$  consists of a set  $V = \{x_1, \dots, x_n\}$  of nodes representing genes and a list  $F = (f_1, \dots, f_n)$  of functions, where a function  $f_i(x_{i_1}, \dots, x_{i_k})$  with inputs from specified nodes  $x_{i_1}, \dots, x_{i_k}$  is assigned to each node  $x_i$ . In general, each node takes values from a finite set of values (e.g., binary or ternary). The state or expression pattern  $\psi$  of the network is composed of the values of all nodes. The expression pattern  $\psi_{t+1}$  at time  $t + 1$  is determined by the functions in  $F$  from the expression pattern  $\psi_t$  at time  $t$ . Collectively, the states of individual genes in the genome form a gene activity profile (GAP). Although there are a large number of possible GAPs, most functions have only a few

essential variables [2,31,35]. To overcome the deterministic rigidity of the above networks, the network concept is extended to a probabilistic setting. The basic idea is to accommodate more than one possible function for each node. Thus, to every node  $x_i$ , there corresponds a set

$$F_i = \{f_k^{(i)}\}_{k=1,\dots,l(i)}, \quad i = 1, 2, \dots, n, \quad (1)$$

where each  $f_k^{(i)}$  is a possible function determining the value of gene  $x_i$  and  $l(i)$  is the number of possible functions for gene  $x_i$ . The functions  $\{f_k^{(i)}\}$  are also referred to as *predictors*. If there are  $N$  possible realizations, then there are  $N$  vector functions  $f_1, f_2, \dots, f_N$  of the form  $f_j = (f_{j_1}^{(1)}, f_{j_2}^{(2)}, \dots, f_{j_n}^{(n)})$ , for  $j = 1, 2, \dots, N$ ,  $1 \leq j_i \leq l(i)$  and where  $f_{j_i}^{(i)} \in F_i$  ( $i=1, \dots, n$ ). In other words, the vector function  $f_j$  acts as a transition function representing a possible realization of the entire network. Let  $f = (f^{(1)}, f^{(2)}, \dots, f^{(n)})$  be a random vector taking values in  $F_1 \times \dots \times F_n$ . That is,  $f$  can take on all possible realizations of the network. Then, the probability that predictor  $f_k^{(i)}$ ,  $1 \leq k \leq l(i)$ , is used to predict gene  $i$  is

$$c_k^{(i)} = P\{f^{(i)} = f_k^{(i)}\} = \sum_{j: f_{j_i}^{(i)} = f_k^{(i)}} P\{f = f_j\}. \quad (2)$$

Obviously,  $\sum_{k=1}^{l(i)} c_k^{(i)} = 1$ . Thus, a probabilistic gene regulatory network  $G(V, F)$  is defined by a set of nodes  $V = \{x_1, \dots, x_n\}$  and the list  $F = (F_1, \dots, F_n)$ , where  $F_i$  is defined in (1). A network is said to be *independent*

if  $f^{(1)}, f^{(2)}, \dots, f^{(n)}$  are probabilistically independent. Unless otherwise mentioned, we will assume independent networks. Fig. 1 illustrates a building block of a probabilistic gene regulatory network. Assuming independence,  $N = \prod_{i=1}^n l(i)$  is the number of possible network realizations. If  $l(i) = 1$  for all  $i = 1, \dots, n$ , then  $N = 1$  and it reduces to the standard deterministic network.

### 2.2. Problem statement

In a probabilistic gene regulatory network, for every node  $x_i$ , there are  $l(i)$  functions  $F_i = \{f_k^{(i)}\}_{k=1}^{l(i)}$ , where each  $f_k^{(i)}$  is a possible function determining the value of gene  $x_i$ , and each function is chosen to determine  $x_i$  with probability  $c_k^{(i)}$ ,  $k = 1, 2, \dots, l(i)$ . In order to build the network for gene  $x_i$ , we implement the following four steps:

- (1) Determine  $l(i)$  and the input set of variables corresponding to function  $f_k^{(i)}$ ,  $k = 1, 2, \dots, l(i)$ . This is done by using a novel clustering technique based on mutual information minimization. Simulated annealing is employed to solve the optimization problem.
- (2) Each  $f_k^{(i)}$  is then modelled by a perceptron consisting of a linear term and a nonlinear term. A reversible jump MCMC technique is used to calculate the model order and the parameters.
- (3) The coefficient of determination (CoD) [18] is employed to compute the probabilities  $c_k^{(i)}$ ,

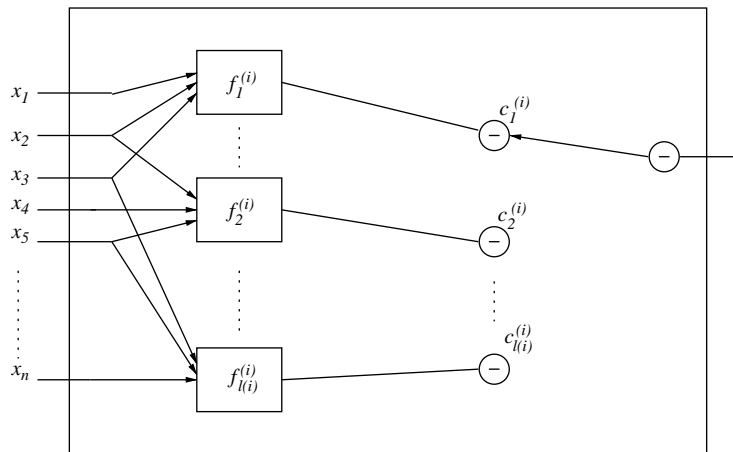


Fig. 1. A building block of a probabilistic gene regulatory network.

$k = 1, 2, \dots, l(i)$ , of selecting different predictors for gene  $i$ .

- (4) When the data sample size is small, we can adjust the clustering result and the corresponding predictors by a local search technique.

### 3. Gene clustering via mutual information minimization

In the case of gene-expression microarrays, the data are severely limited relative to the number of variables in the system. Consequently, it is necessary to restrict the number of variables over which a function is defined. Fortunately, most functions require only a few essential variables [35].

#### 3.1. Mutual information

The motivation for considering mutual information is its capacity to measure a general dependence among random variables. The main idea of this technique is to identify a relatively small number of candidate parents for each gene based on statistics such as correlation. We then restrict our search to networks in which only the candidate parents of a variable can be its parents, resulting in a smaller search space in which we can more efficiently find a good structure.

Shannon's information theory provides a suitable formalism for quantifying the above concepts. The *entropy* of a gene expression pattern is a measure of the uncertainty information content in that pattern. Given a random vector  $\mathbf{X}$  and its probability distribution  $P(\mathbf{X} = \mathbf{x}_i)$ ,  $i = 1, \dots, N_x$ , where  $N_x$  is the number of possible values  $\mathbf{X}$  can take, the *entropy* is defined as

$$H(\mathbf{X}) \triangleq - \sum_{i=1}^{N_x} P(\mathbf{X} = \mathbf{x}_i) \ln P(\mathbf{X} = \mathbf{x}_i). \quad (3)$$

Higher entropy for gene variables means that their expression levels are more uniformly distributed. The *joint entropy* of  $\mathbf{X}$  and  $\mathbf{Y}$  is a measure of the total uncertainty contained in  $\mathbf{X}$  and  $\mathbf{Y}$ , and is defined as

$$H(\mathbf{X}, \mathbf{Y}) \triangleq - \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} P(\mathbf{X} = \mathbf{x}_i, \mathbf{Y} = \mathbf{y}_j) \times \ln P(\mathbf{X} = \mathbf{x}_i, \mathbf{Y} = \mathbf{y}_j), \quad (4)$$

where  $N_y$  is the number of possible values  $\mathbf{Y}$  can take. The *mutual information* between  $\mathbf{X}$  and  $\mathbf{Y}$  is a measure of information about  $\mathbf{X}$  (or  $\mathbf{Y}$ ) contained in  $\mathbf{Y}$  (or  $\mathbf{X}$ ), and is given by

$$I(\mathbf{X}; \mathbf{Y}) \triangleq H(\mathbf{Y}) - H(\mathbf{X} | \mathbf{Y}) \\ = H(\mathbf{X}) - H(\mathbf{Y} | \mathbf{X}) \quad (5)$$

$$= H(\mathbf{X}) + H(\mathbf{Y}) - H(\mathbf{X}, \mathbf{Y}) \quad (6)$$

$$= \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} P(\mathbf{X} = \mathbf{x}_i, \mathbf{Y} = \mathbf{y}_j) \\ \times \ln \frac{P(\mathbf{X} = \mathbf{x}_i, \mathbf{Y} = \mathbf{y}_j)}{P(\mathbf{X} = \mathbf{x}_i)P(\mathbf{Y} = \mathbf{y}_j)}. \quad (7)$$

It is known that mutual information is always non-negative, i.e.,  $I(\mathbf{X}; \mathbf{Y}) \geq 0$  [13]. In practice, the probabilities in (7) are estimated by the corresponding histograms, i.e.,

$$P(\mathbf{X} = \mathbf{x}_i, \mathbf{Y} = \mathbf{y}_j) \cong \frac{\#(\mathbf{x}_i, \mathbf{y}_j)}{N}, \quad (8)$$

$$P(\mathbf{X} = \mathbf{x}_i) \cong \frac{\#(\mathbf{x}_i)}{N}, \quad (9)$$

$$P(\mathbf{Y} = \mathbf{y}_j) \cong \frac{\#(\mathbf{y}_j)}{N}, \quad (10)$$

where  $N$  is the total number of samples, and  $\#(\mathbf{x}_i)$  denotes the number of occurrences of the  $\mathbf{x}_i$ .

#### 3.2. Objective function

Suppose we are to partition the set of gene variables into  $k$  disjoint subsets as  $V = \mathbf{X}_1 \cup \mathbf{X}_2 \cup \dots \cup \mathbf{X}_k$ . The cost function is defined as the sum of pair-wise mutual information between any two subsets,

$$f(s) = \sum_{i \neq j} I(\mathbf{X}_i; \mathbf{X}_j), \quad (11)$$

where  $s$  denotes a particular partition scheme. The simulated annealing algorithm is employed to find an optimal partition scheme such that the cost function attains the minimum, i.e.,

$$s^* = \arg \min_{s \in S} f(s), \quad (12)$$

where  $S$  denotes the set of all possible  $k$ -partition schemes. Finally, by varying the value of  $k$ , we can

find the number of clusters with the minimal mutual information. Next we will give two algorithms to solve the above optimization problem.

### 3.3. Optimization

The original simulated annealing algorithm [1] is a computational modelling of the physical process of annealing from metallurgical engineering, where a molten metal is cooled in such a way as to reach the minimal energy state possible at the final temperature. The basic procedure involves a cooling procedure, in which a temperature parameter starts out high and is gradually lowered until the system is frozen. At each temperature, the state is perturbed many times, which avoids the limitation of being initialization-dependent. The algorithm moves to the next temperature in the schedule until the system reaches thermal equilibrium at that temperature on the basis of a decreasing energy cost function.

Let  $s_0, s_1$  denote two different  $k$ -partition schemes with cost values  $f(s_0)$  and  $f(s_1)$ , respectively. Then  $s_1$  is accepted from  $s_0$  according to the acceptance probability:

$$P\{\text{accept } s_1\} = \begin{cases} 1 & \text{if } f(s_1) \leq f(s_0), \\ \exp\left(\frac{-(f(s_1)-f(s_0))}{T}\right) & \text{if } f(s_1) > f(s_0), \end{cases} \quad (13)$$

where  $T \in \mathbb{R}^+$  denotes the control parameter (temperature factor). To generate a new partition  $s_1$  from  $s_0$ , we randomly select two clusters from  $s_0$  and then randomly pick a gene variable from one cluster and put it in the other cluster.

The basic simulated annealing procedure suffers from a very slow convergence, we therefore resort to the fast annealing techniques [32]. The basic idea is to run a set of  $\kappa$  ( $\kappa = k$  in our experiments) annealing procedures in parallel and to adjust the annealing schedules adaptively. The initial temperature is set as  $T = 10$  in our experiments.

### 3.4. Determine the number of clusters

We set an upper bound  $K$  on the number of clusters. For each  $k \leq K$ , we run the above simulated

annealing algorithm to find the optimal partition  $s_k$ . The final best gene clustering scheme is then given by  $s_{k^*}$ , where  $k^* = \arg \min_{2 \leq k \leq K} f(s_k)$ . The set of parent genes for each gene is obtained by the above mutual information based clustering procedure. However, in gene microarray analysis, the histogram-based probability estimates are imprecise because the sample size is small. Thus, it is not prudent to regard this optimal partition-order  $k^*$  as the final partition order. Moreover, if only one partition scheme  $s_{k^*}$  is regarded as the final partition scheme, then each gene only belongs to one set, meaning there is only one influential gene set to predict each gene. But according to Heyer et al. [29] and Shmulevich et al. [41], that may not be true. Therefore, we choose several partition orders around  $k^*$ , and in particular, from the partition  $s_{k^*-m_1}, \dots, s_{k^*}, \dots, s_{k^*+m_2}$  for some integers  $m_1$  and  $m_2$ . Now let the corresponding clusters containing gene  $x_i$  be  $C_1^{(i)}, C_2^{(i)}, \dots, C_{l(i)}^{(i)}$ ,  $l(i) \leq m_1 + m_2 + 1$ , then it is natural to define the  $l(i)$  sets  $\{\mathbf{X}_k^{(i)}\}_{k=1}^{l(i)}$  as influencing sets for gene  $x_i$  to be composed of the other genes in  $C_1, C_2, \dots, C_{l(i)}$ . For the moment, let us proceed under this assumption. Later on we will make an adjustment on this assumption.

In the next section, we adopt a new nonlinear model to estimate the predictors (functions) for each gene. A reversible jump Markov chain Monte Carlo (MCMC) annealing method is adopted to perform the necessary computations for parameter estimation.

## 4. Predictor construction using reversible jump MCMC annealing

### 4.1. Predictor model

Based on the possible  $l(i)$  sets  $\{\mathbf{X}_k^{(i)}\}_{k=1}^{l(i)}$ , we want to estimate the predictors  $f_k^{(i)} : \mathbf{X}_k^{(i)} \rightarrow y_i$ ,  $k = 1, \dots, l(i)$  from the observations (in fact,  $y_i$  should be  $x_i$ , but to describe the functions clearly we let  $y_i$  stand for  $x_i$ ). For notational convenience, we consider a general formulation,  $f : \mathbf{X} \rightarrow y$ . We postulate a multivariate-input, single-output mapping

$$y_t = f(\mathbf{x}_t) + n_t, \quad (14)$$

where  $\mathbf{x}_t \in \mathbb{R}^d$  is a set of input variables,  $y_t \in \mathbb{R}$  is a target variable,  $n_t$  is a noise term, and  $t \in \{1, 2, \dots\}$  is an index variable over the data. The learning problem

involves computing an approximation to the function  $f$  and estimating the characteristics of the noise process given a set of  $N$  input–output observations,

$$\mathcal{O} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N; y_1, y_2, \dots, y_N\}. \tag{15}$$

Typical examples include regression, prediction and classification.

We consider the following family of regression equations (the estimated predictor denoted by  $\hat{f}$ ):

$$\mathcal{M}_0 : y_t = b + \boldsymbol{\beta}^T \mathbf{x}_t + n_t, \quad k = 0, \tag{16}$$

$$\mathcal{M}_J : y_t = \sum_{j=1}^J a_j \phi(\|\mathbf{x}_t - \boldsymbol{\mu}_j\|) + b + \boldsymbol{\beta}^T \mathbf{x}_t + n_t, \tag{17}$$

$$1 \leq J \leq J_{\max},$$

where  $J_{\max}$  is an upper-bound on the number of non-linear terms in (17);  $\phi$  is a radial basis function (RBF);  $\|\cdot\|$  denotes a distance metric (usually Euclidean or Mahalanobis);  $\boldsymbol{\mu}_j \in \mathbb{R}^d$  denotes the  $j$ th RBF center;  $a_j \in \mathbb{R}$  is the  $j$ th RBF coefficient;  $b \in \mathbb{R}$ ;  $\boldsymbol{\beta} \in \mathbb{R}^d$  are the linear regression parameters; and  $n_t \in \mathbb{R}$  is assumed to be i.i.d. noise. Depending on our a priori knowledge about the smoothness of the mapping, we can choose different types of basis functions. Here the Gaussian basis function is used, given by  $\phi(\varrho) = \exp(-\lambda\varrho^2)$ , where  $\lambda = 1$  in our experiments. We express model (17) in vector-matrix form as

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\triangleq \mathbf{y}} = \underbrace{\begin{bmatrix} 1 & x_{1,1} \cdots x_{1,d} & \phi(\|\mathbf{x}_1 - \boldsymbol{\mu}_1\|) \cdots \phi(\|\mathbf{x}_1 - \boldsymbol{\mu}_J\|) \\ 1 & x_{2,1} \cdots x_{2,d} & \phi(\|\mathbf{x}_2 - \boldsymbol{\mu}_1\|) \cdots \phi(\|\mathbf{x}_2 - \boldsymbol{\mu}_J\|) \\ \vdots & \vdots & \vdots \\ 1 & x_{N,1} \cdots x_{N,d} & \phi(\|\mathbf{x}_N - \boldsymbol{\mu}_1\|) \cdots \phi(\|\mathbf{x}_N - \boldsymbol{\mu}_J\|) \end{bmatrix}}_{\triangleq \mathbf{D}} \underbrace{\begin{bmatrix} b \\ \beta_1 \\ \vdots \\ \beta_d \\ a_1 \\ \vdots \\ a_J \end{bmatrix}}_{\triangleq \boldsymbol{\alpha}} + \underbrace{\begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_N \end{bmatrix}}_{\triangleq \mathbf{n}}, \tag{18}$$

that is

$$\mathbf{y} = \mathbf{D}\boldsymbol{\alpha} + \mathbf{n}. \tag{19}$$

The noise variance is assumed to be  $\sigma^2$ . We assume here that the number  $J$  of RBFs and the parameters

$\boldsymbol{\Theta}_J \triangleq \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_J, \sigma^2, \boldsymbol{\alpha}\}$  are unknown. Given the data set  $\mathcal{O}$  in (15), our objective is to estimate  $J$  and  $\boldsymbol{\Theta}_J$ , where  $\boldsymbol{\Theta}_J \in \mathbb{R}^m \times \mathbb{R}^+ \times \Omega_J$  with  $m = 1 + d + J$ ; that is  $\boldsymbol{\alpha} \in \mathbb{R}^m$ ,  $\sigma^2 \in \mathbb{R}^+$ , and  $\boldsymbol{\mu} \in \Omega_J$ , where

$$\Omega_J = \{\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_J); \tag{20}$$

$$\mu_{j,i} \in \left[ \min_{1 \leq l \leq N} x_{l,i} - \varphi, \max_{1 \leq l \leq N} x_{l,i} + \varphi \right],$$

$$j = 1, \dots, J; i = 1, \dots, d\}.$$

Here we set  $\varphi = 0.5$  in our experiments. The Bayesian inference of  $J$  and  $\boldsymbol{\Theta}_J$  is based on the joint posterior distribution  $p(J, \boldsymbol{\Theta}_J | \mathcal{O})$ . Our aim is to estimate this joint distribution from which, by standard probability marginalization and transformation techniques, one theoretically expresses all posterior features of interest. It is not possible to obtain these quantities analytically, since this would require the evaluation of high dimensional integrals of nonlinear functions in the parameters. Consequently, we propose to use an MCMC method to perform Bayesian computation. The basic idea of MCMC is to build an ergodic Markov chain  $\{J^{(i)}, \boldsymbol{\Theta}_J^{(i)}, i \in \mathbb{N}\}$  whose equilibrium distribution is the desired joint posterior distribution  $p(J, \boldsymbol{\Theta}_J | \mathcal{O})$ .

The linear model (16) has been studied in [36]. Here, we extend the basic model to interpolate the nonlinear terms as in (17). It is known that the model (17) exhibits a universal approximation

property [5]. As noted in [36], a key problem with using the neural model for microarray prediction is its imprecision when data sets are limited. In our proposed approach, information-based clustering yields

an initial partition so that only a part of the full gene set is tied to a specific gene. We adaptively estimate the nonlinear model-order  $J$ , which is adjusted to zero if the data set is insufficient to estimate the predictors.

4.2. Reversible jump Markov chain Monte Carlo annealing

From model (19), given  $J, \mu_1, \dots, \mu_J$ , the least-squares estimate of  $\alpha$  is given by

$$\hat{\alpha} = [D^T D]^{-1} D^T y. \tag{21}$$

An estimate of  $\sigma^2$  is then given by

$$\hat{\sigma}^2 = \frac{1}{N} (y - D\hat{\alpha})^T (y - D\hat{\alpha}) = \frac{1}{N} y^T P_J^* y, \tag{22}$$

where

$$P_J^* \triangleq I_N - D[D^T D]^{-1} D^T. \tag{23}$$

Based on the minimum description length (MDL) criterion, we can impose the following a priori distribution on  $J$  [5],

$$P(J) \propto \exp \left[ - \left( J + \frac{d+1}{2} \right) \log N \right]. \tag{24}$$

Assuming the noise samples are i.i.d. Gaussian, it can then be shown that the joint posterior distribution of  $(J, \mu_1, \dots, \mu_J)$  is given by [5]

$$p(J, \mu_1, \dots, \mu_J) \propto [(y^T P_J^* y)^{-N/2}] P(J). \tag{25}$$

Hence the maximum a posteriori (MAP) estimate of these parameters is obtained by maximizing the right-hand side of (25). This can be done by using the reversible jump MCMC algorithm [24].

MCMC is a family of procedures for Bayesian computations. The basic idea is to draw random samples from a Markov chain, whose equilibrium distribution is the target distribution. These samples are then used to approximate the desired inference. The reversible jump MCMC algorithm is capable of jumping between parameter spaces of different dimensions [24]. Here we employ a reversible jump MCMC algorithm [5] to jointly estimate the model-order  $J$  ( $J \leq J_{\max}$ ) and the RBF centers  $\mu_1, \dots, \mu_J$  at each iteration. Note that  $J_{\max}$  should be set as a small number when the number of samples is very small. In this study,  $J_{\max} = \lfloor N/10 \rfloor$ .

The initial values of  $\mu_1, \dots, \mu_J$  are randomly chosen according to (20). At each iteration, this algorithm performs one of the following five possible operations:

- (1) Birth jump: Let  $J \leftarrow J + 1$ , and update  $\mu_1, \dots, \mu_{J+1}$ .
- (2) Death jump: Random drop one of the  $J$  RBF terms in (17); let  $J \leftarrow J - 1$ , and update  $\mu_1, \dots, \mu_{J-1}$ .
- (3) Merge jump: Randomly choose one of the  $J$  RBF terms (17), and merge it with its closest neighbor; let  $J \leftarrow J - 1$ , and update  $\mu_1, \dots, \mu_{J-1}$ .
- (4) Split jump: Randomly choose one of the  $J$  RBF terms (17), and split it into two neighbor RBF terms; let  $J \leftarrow J + 1$ , and update  $\mu_1, \dots, \mu_{J+1}$ .
- (5) No jump:  $J$  remains unchanged; update  $\mu_1, \dots, \mu_J$ .

Throughout the algorithm, the above five jumps are performed with probabilities  $b_J, d_J, m_J, s_J$ , and  $u_J$ , respectively. We have  $b_J + d_J + m_J + s_J + u_J = 1$  for all  $0 \leq J \leq J_{\max}$ . In our experiments, we set  $b_J = d_J = m_J = s_J = u_J = 0.2$ . At each iteration, after one of the above jumps is selected, a Metropolis–Hasting (MH) step is then followed to determine whether that jump is accepted. That is, let the acceptance ratio for the proposed jump be  $r_{\text{jump}}$ , then the above jump is accepted with probability  $\min\{1, r_{\text{jump}}\}$ . The detailed calculation of the acceptance ratios for the different jumps are given in Appendix A. For the annealing scheme, the initial temperature is set as  $T = 10$ , and it is reduced iteratively according to  $T \leftarrow 0.85T$  in our experiments.

**Example.** We illustrate the performance of the above prediction construction procedure for the following model

$$y_t = 1^T x_t + 2 \exp(-\|x_t\|^2) + n_t, \quad n_t \sim \mathcal{N}(0, 1),$$

where  $x_t \in \mathbb{R}^d$ . We consider four cases:  $d=3, 5, 10$ , and  $20$ . In each case, each element of  $x_t$  is independently generated from a uniform distribution on  $[0, 1]$ . The total numbers of training and testing data are both 50. The results are given in terms of variance unexplained (PVU) [5,30], defined by

$$\text{PVU} = \frac{\mathbb{E}[\hat{f}(X) - f(X)]^2}{\mathbb{E}[f(X) - \mathbb{E}f(X)]^2}.$$

Table 1  
The PVU values of perceptron predictor and the proposed predictor

	$d$	Reversible jump MCMC	Perceptron
Training	3	0.0269	0.9221
Testing	3	0.0617	0.9527
Training	5	0.4930	0.9514
Testing	5	0.7909	1.0569
Training	10	0.9061	0.9621
Testing	10	1.0120	1.2027
Training	20	1.0300	1.0300
Testing	20	1.4569	1.4569

Table 1 shows the PVU values of two methods—perceptron [36] and reversible jump MCMC annealing. The performance of the MCMC method is much better than the perceptron when the data dimension  $d$  is small. The performance of the two methods is similar for larger values of  $d$ , in which case the prediction performance is not good. A key problem with using a complex model for microarray prediction is that multiple-layer stochastic training can be very imprecise when data sets are limited. Hence, we do not fix the nonlinear term in our model, and adaptively estimate the model's order. Comparatively, the perceptron model is too simple to describe the complex functions of gene interactions from the simulated data.

## 5. Coefficients of determination for predictors

A natural way to select a set of predictors for a given gene is to employ the *coefficient of determination* [17,36,41]. The CoD measures the degree to which prediction of the transcription level of a target gene is improved relative to the best possible prediction in the absence of observations. Let  $x_i$  be the target gene;  $\mathbf{X}_1^{(i)}, \mathbf{X}_2^{(i)}, \dots, \mathbf{X}_{l(i)}^{(i)}$  be sets of genes; and  $f_1^{(i)}, f_2^{(i)}, \dots, f_{l(i)}^{(i)}$  be function rules such that  $f_1^{(i)}(\mathbf{X}_1^{(i)}), f_2^{(i)}(\mathbf{X}_2^{(i)}), \dots, f_{l(i)}^{(i)}(\mathbf{X}_{l(i)}^{(i)})$  are optimal predictors of  $x_i$  relative to some probabilistic error measure  $\varepsilon(x_i, f_k^{(i)}(\mathbf{X}_k^{(i)}))$ , which is defined

as

$$\varepsilon(x_i, f_k^{(i)}(\mathbf{X}_k^{(i)})) \triangleq \mathbb{E}[|g(\hat{f}_k^{(i)}(\mathbf{X}_k^{(i)})) - x_i|^2],$$

$$i = 1, \dots, n; k = 1, \dots, l(i), \quad (26)$$

where  $g$  is a  $\{-1, 0, 1\}$ -valued threshold function [ $g(z) = -1$  if  $z < -0.5$ ,  $g(z) = 0$  if  $-0.5 \leq z \leq 0.5$ , and  $g(z) = 1$  if  $z > 0.5$ ], and  $\hat{f}_k^{(i)}(\mathbf{X}_k^{(i)})$  is estimated by the reversible jump MCMC method developed in Section 4 or the perceptron [36]. For each  $k$ , the CoD for  $x_i$  relative to the conditioning set  $\mathbf{X}_k^{(i)}$  is defined by [41,44]

$$\theta_k^i = \frac{\varepsilon_i - \varepsilon(x_i, f_k^{(i)}(\mathbf{X}_k^{(i)}))}{\varepsilon_i},$$

$$i = 1, \dots, n; k = 1, \dots, l(i), \quad (27)$$

where  $\varepsilon_i$  is the error of the best (constant) estimate of  $x_i$  in the absence of any conditional variables. In the case of minimum mean-square error estimation,  $\varepsilon_i$  is the error of the mean of  $x_i$ , which is the best constant estimate, i.e.,  $\varepsilon_i = \mathbb{E}[|x_i - g(\mathbb{E}(x_i))|^2]$ .

In this study, we limited the number of parents of each gene to be less than 10. However, if it happens that there are more than 10 genes in some classes, then we use a local optimal clustering algorithm to re-cluster those sets or use the CoD technique directly. Once a class of gene sets  $\mathbf{X}_1^{(i)}, \mathbf{X}_2^{(i)}, \dots, \mathbf{X}_{l(i)}^{(i)}$  has been selected, we take the designed approximations of the optimal function rules  $f_1^{(i)}, f_2^{(i)}, \dots, f_{l(i)}^{(i)}$  as the rule for gene  $x_i$  with the probability of  $f_k^{(i)}$  being

$$c_k^{(i)} = \frac{\theta_k^{(i)}}{\sum_{j=1}^{l(i)} \theta_j^{(i)}}, \quad i = 1, \dots, n, k = 1, \dots, l(i), \quad (28)$$

where the CoDs are the estimates formed from the training data according to (27). Accordingly, those functions corresponding to the highest CoDs will be selected more often in the probabilistic network. The number of chosen predictors,  $l(i)$  is selected in the clustering stage.

After obtaining the possible  $l(i)$  sets  $\{\mathbf{X}_k^{(i)}\}_{k=1}^{l(i)}$  for each gene  $x_i$ , we can estimate the predictors  $f_k^{(i)} : \mathbf{X}_k^{(i)} \rightarrow x_i, k = 1, \dots, l(i)$ , using the reversible jump MCMC annealing algorithm. The CoDs and the probabilities  $\{c_k^{(i)}\}$  are estimated according to (27)

and (28). Probabilistic gene regulatory network construction is complete: for every node  $x_i$ , we have a set  $F_i = \{f_k^{(i)}\}_{k=1}^{l(i)}$  of  $l(i)$  functions, where each  $f_k^{(i)}$  is a possible function (with probability  $c_k^{(i)}$ ) determining the value of gene  $x_i$ .

The number  $l(i)$  of predictors is based on the optimization of the number of clusters. Suppose the clusters containing gene  $x_i$  are  $C_1, C_2, \dots, C_{l(i)}$ . Owing to the difficulty of estimating entropy with small sample sizes, there is potential imprecision in these clusters. Consequently, when sample sizes are small, as they are in the application we will show next, we will proceed in a more flexible manner in choosing the  $l(i)$  sets. Suppose  $C_k = \{x_i, g_{k1}, g_{k2}, \dots, g_{k,r(k)}\}$ , where  $r(k)$  is the number of genes in  $C_k$  besides  $x_i$ . For  $j = 1, 2, \dots, r(k)$ , form the sets  $\{C_k, C_{kj1}, C_{kj2}, \dots, C_{k,j,s(k)}\}$ , where  $s(k)$  is the number of genes not in  $C_k$ , by interchanging  $g_{kj}$  with each of the genes not in  $C_k$ . We then choose  $f_k^{(i)}$  to be the predictor with highest CoD from among the predictors formed from  $C_{kj1}, C_{kj2}, \dots, C_{k,j,s(k)}$ ,  $j = 1, 2, \dots, r(k)$ . While this increases the computational burden of finding the predictors, it remains far less than required by the full-search method.

## 6. Experimental results

The ability of the mutual-information clustering algorithm and MCMC predictors to build probabilistic gene regulatory networks based on the observations in transcription level has been tested in the context of responsiveness to genotoxic stresses. As a result of a microarray study surveying transcription of 1238 genes during the response of a myeloid line to ionizing radiation [4], thirty genes not previously known to participate in response to IR were found to be responsive. To further characterize the responsiveness of these genes to genotoxic stresses, the responsiveness of a subset of 9 of them was examined by blot assays in 12 cell lines stimulated with ionizing radiation (IR), a chemical mutagen methane sulfonate (MMS), or ultraviolet (UV). The cell lines were chosen so that a sampling of both  $p53$  proficient and  $p53$  deficient cells would be assayed. By using the same data set used for the initial studies concerning nonlinear prediction in the context of genomic microarray

data [36,37], we are able to compare the results of the proposed, information-MCMC approach with the previously obtained results. Here we relate some results to [36], which considered both unconstrained and perceptron predictors, and focused on design issues for predictors, whereas [37] considered only perceptron and focused on biological issues.

The ternary data of the survey (14 genes and 30 samples) are given in Table 2, where the conditions IR, MMS, and UV have the values 1 or 0, depending on whether the condition is or is not in effect. To maintain consistency with [36,37], we continue to employ the blind controls used in those studies, in which expression patterns for two fictitious genes were created. Rules were made for the fictitious genes dependent on other gene states in the set, and the degrees of concordance of the observations to the rules were varied. *AHA* has the following rule: up-regulated if  $p53$  up-regulated, down-regulated if *RCH1* and  $p53$  down-regulated. Full concordance with the rule would produce 15 instances of up regulation and five instances of down regulation. The data set generated for *AHA* includes 13 of the 15 up regulations and all 5 down regulations. The rule for *OHO* is: up-regulated if *MDM2* up-regulated and *RCH1* down-regulated, and down-regulated if  $p53$  down-regulated and *REL-B* up-regulated. Full concordance with this rule would produce four instances of up regulation and five instances of down regulation. The data set generated for *OHO* has the four expected up regulations plus seven unpredicted up regulations, and only two of the five predicted down regulations.

Tables 3–8 show the results of applying the clustering algorithm to the ternary expression data in Table 2. The optimal partition number is four, and we select the  $k$ -partition schemes for  $k = 2$  to 7. From these tables, we can find the possible parents for each gene. For example, the possible parents of gene  $x_1$  are  $\{x_2, x_9\}$ ,  $\{x_2, x_3, x_9\}$ ,  $\{x_6, x_8, x_9\}$ ,  $\{x_3, x_8, x_9, x_{14}\}$ ,  $\{x_2, x_3, x_4, x_6, x_7, x_8, x_9, x_{10}\}$  and  $\{x_2, x_3, x_4, x_6, x_7, x_8, x_9, x_{14}\}$ ; the possible parents of gene  $x_{10}$  are  $\{x_3, x_{14}\}$ ,  $\{x_2, x_4, x_6, x_7\}$ ,  $\{x_1, x_2, x_3, x_4, x_6, x_7, x_8, x_9\}$  and  $\{x_5, x_{11}, x_{12}, x_{13}\}$ ; the possible parents of gene  $x_{11}$  are  $\{x_5\}$ ,  $\{x_5, x_{14}\}$  and  $\{x_5, x_{10}, x_{12}, x_{13}\}$ ; and the possible parents of gene  $x_{12}$  are  $\{x_{13}\}$ , and  $\{x_5, x_{10}, x_{11}, x_{13}\}$ . Note that the artificial gene *AHA* ( $x_{13}$ ) has been defined so that it is tightly coupled to  $p53$  ( $x_{12}$ ). This is reflected in the fact that *AHA* and

Table 2  
Ternary expression data [36]

Gene index		Gene													
		$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$
Cell line	Condition	<i>RCH1</i>	<i>BCL3</i>	<i>FRA1</i>	<i>REL-B</i>	<i>ATF3</i>	<i>IAP-1</i>	<i>PC-1</i>	<i>MBP-1</i>	<i>SSAT</i>	<i>MDM2</i>	<i>p21</i>	<i>p53</i>	<i>AHA</i>	<i>OHO</i>
ML-1	IR	-1	1	1	1	1	1	1	1	1	1	1	1	1	1
ML-1	MMS	0	0	0	0	1	0	0	0	0	1	1	1	1	0
Molt4	IR	-1	0	0	1	1	0	1	0	0	1	1	1	1	1
Molt4	MMS	0	0	1	0	1	0	0	0	0	0	1	1	1	0
SR	IR	-1	0	0	1	1	1	1	1	0	1	1	1	1	1
SR	MMS	0	0	0	0	1	0	0	0	0	1	1	1	1	0
A549	IR	0	0	0	0	0	0	0	0	0	1	1	1	0	0
A549	MMS	0	0	0	0	1	0	0	0	0	0	1	1	1	0
A549	UV	0	0	0	0	1	0	0	0	0	0	1	1	1	0
MCF7	IR	-1	0	1	1	0	0	0	0	0	1	1	1	0	1
MCF7	MMS	0	0	1	0	1	0	0	0	0	1	1	1	1	0
MCF7	UV	0	0	1	1	1	0	0	0	0	1	1	1	1	0
RKO	IR	0	1	0	1	1	1	0	0	0	1	1	1	1	0
RKO	MMS	0	0	0	0	1	0	0	0	0	0	1	1	1	0
RKO	UV	0	0	0	0	1	0	0	0	0	0	1	1	1	0
CCRF-CEM	IR	-1	1	1	1	1	0	1	0	0	0	0	-1	-1	0
CCRF-CEM	MMS	0	0	0	0	1	0	0	0	0	0	0	-1	0	0
HL60	IR	-1	1	0	1	1	0	1	0	1	0	1	-1	-1	-1
HL60	MMS	0	0	1	0	1	0	0	0	0	1	1	-1	0	1
K562	IR	0	0	0	0	0	0	0	0	0	0	0	-1	0	0
K562	MMS	0	0	0	0	1	0	0	0	0	0	0	-1	0	0
H1299	IR	0	0	0	1	0	0	1	0	0	0	0	-1	0	0
H1299	MMS	0	0	0	0	1	0	0	0	0	0	1	-1	0	1
H1299	UV	0	0	0	0	1	0	1	0	0	0	1	-1	0	1
RKO/E6	IR	-1	1	0	1	0	1	1	0	0	0	0	-1	-1	0
RKO/E6	MMS	-1	0	0	0	1	0	0	0	0	0	1	-1	-1	1
RKO/E6	UV	-1	0	0	0	1	0	0	0	0	0	1	-1	-1	1
T46D	IR	0	0	0	1	0	0	0	0	0	0	1	-1	0	-1
T46D	MMS	0	0	0	0	1	0	0	0	0	0	1	-1	0	1
T46D	UV	0	0	0	0	1	0	0	0	0	0	1	-1	0	1

Table 3  
Results of two classes

Class-1	$x_5$	$x_{10}$	$x_{11}$	$x_{12}$	$x_{13}$				
	<i>ATF3</i>	<i>MDM2</i>	<i>p21</i>	<i>p53</i>	<i>AHA</i>				
Class-2	$x_1$	$x_2$	$x_3$	$x_4$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{14}$
	<i>RCH1</i>	<i>BCL3</i>	<i>FRA1</i>	<i>REL-B</i>	<i>IAP-1</i>	<i>PC-1</i>	<i>MBP-1</i>	<i>SSAT</i>	<i>OHO</i>

*p53* are in the same partition in each table. No corresponding phenomenon occurs for the artificial gene *OHO* ( $x_{14}$ ), whose definition depends on four genes and for which concordance with the rule is poor.

It is instructive to compare the results of the new information-MCMC approach with those obtained by using a full search restricted by a small number of predictor genes. To do so, we consider four target genes,

Table 4  
Results of three classes

Class-1	$x_{12}$ <i>p53</i>	$x_{13}$ <i>AHA</i>							
Class-2	$x_5$ <i>ATF3</i>	$x_{11}$ <i>p21</i>	$x_{14}$ <i>OHO</i>						
Class-3	$x_1$ <i>RCH1</i>	$x_2$ <i>BCL3</i>	$x_3$ <i>FRA1</i>	$x_4$ <i>REL-B</i>	$x_6$ <i>IAP-1</i>	$x_7$ <i>PC-1</i>	$x_8$ <i>MBP-1</i>	$x_9$ <i>SSAT</i>	$x_{10}$ <i>MDM2</i>

Table 5  
Results of four classes

Class-1	$x_{12}$ <i>p53</i>	$x_{13}$ <i>AHA</i>							
Class-2	$x_5$ <i>ATF3</i>	$x_{11}$ <i>p21</i>							
Class-3	$x_1$ <i>RCH1</i>	$x_3$ <i>FRA1</i>	$x_8$ <i>MBP-1</i>	$x_9$ <i>SSAT</i>	$x_{14}$ <i>OHO</i>				
Class-4	$x_2$ <i>BCL3</i>	$x_4$ <i>REL-B</i>	$x_6$ <i>IAP-1</i>	$x_7$ <i>PC-1</i>	$x_{10}$ <i>MDM2</i>				

Table 6  
Results of five classes

Class-1	$x_{12}$ <i>p53</i>	$x_{13}$ <i>AHA</i>		
Class-2	$x_5$ <i>ATF3</i>	$x_{11}$ <i>p21</i>		
Class-3	$x_2$ <i>BCL3</i>	$x_4$ <i>REL-B</i>	$x_7$ <i>PC-1</i>	
Class-4	$x_3$ <i>FRA1</i>	$x_{10}$ <i>MDM2</i>	$x_{14}$ <i>OHO</i>	
Class-5	$x_1$ <i>RCH1</i>	$x_6$ <i>IAP-1</i>	$x_8$ <i>MBP-1</i>	$x_9$ <i>SSAT</i>

Table 7  
Results of six classes

Class-1	$x_{10}$ <i>MDM2</i>			
Class-2	$x_{12}$ <i>p53</i>	$x_{13}$ <i>AHA</i>		
Class-3	$x_5$ <i>ATF3</i>	$x_{11}$ <i>p21</i>		
Class-4	$x_4$ <i>REL-B</i>	$x_7$ <i>PC-1</i>		
Class-5	$x_6$ <i>IAP-1</i>	$x_8$ <i>MBP-1</i>	$x_{14}$ <i>OHO</i>	
Class-6	$x_1$ <i>RCH1</i>	$x_2$ <i>BCL3</i>	$x_3$ <i>FRA1</i>	$x_9$ <i>SSAT</i>

*p53* ( $x_{12}$ ), *p21* ( $x_{11}$ ), *MDM2* ( $x_{10}$ ), and *BCL3* ( $x_2$ ), and we make some comparisons with results reported in [36]. We will not focus extensively on the exact numerical values obtained in [36] because there is a slight difference in the way in which CoD estimation is handled. In both studies, the CoD of a predictor is

estimated using cross-validation among the 30 samples: 20 samples are randomly chosen for training, the remaining 10 samples are used for test data, and this

Table 8  
Results of seven classes

Class-1	$x_{10}$ <i>MDM2</i>		
Class-2	$x_3$ <i>FRA1</i>		
Class-3	$x_{12}$ <i>p53</i>	$x_{13}$ <i>AHA</i>	
Class-4	$x_5$ <i>ATF3</i>	$x_{11}$ <i>p21</i>	
Class-5	$x_4$ <i>REL-B</i>	$x_7$ <i>PC-1</i>	
Class-6	$x_6$ <i>LAP-1</i>	$x_8$ <i>MBP-1</i>	$x_{14}$ <i>OHO</i>
Class-7	$x_1$ <i>RCH1</i>	$x_2$ <i>BCL3</i>	$x_9$ <i>SSAT</i>

is performed 1000 times to estimate the CoD. An issue arises if  $\varepsilon_i$ , the error of the best (constant) estimate of  $x_i$  in the absence of any conditional variables, has test error 0. This occurs when the random selection results in all test data having the value of the best constant estimate. It can happen quite often when the measurements for a specific gene change very little across the samples. There needs to be an appropriate re-definition of the CoD when it does. This is done differently in the present study than in [36], and is explained in Appendix B.

It is known that *p53* ( $x_{12}$ ) is influential, but not determinative, of the up regulation of both *p21* ( $x_{11}$ ) and *MDM2* ( $x_{10}$ ). Thus, some level of prediction of *p53* should be possible by a combination of these two genes. This expectation is met with *p21* and *MDM2* combining with *ATF3* ( $x_5$ ) and *AHA* ( $x_{13}$ ) as shown in Table 9.

To fully interpret the results, we must recognize that *AHA* is an artificial gene defined to couple with *p53*. This is evidenced in Table 9, which shows that, acting alone as  $\mathbf{X}_1^{(12)}$ , *AHA* predicts *p53* with a substantial CoD. Adjoining *p21*, *MDM2*, and *ATF3* increases the MCMC CoD, whereas the perceptron CoD is unchanged (the small variation in empirical error being insignificant). We conjecture that the in-

crement is due to *p21* and *MDM2*. One should not be surprised to see the inclusion of a noninfluential gene, such as we conjecture *ATF3* to be, in predictor sets. Not only is this inevitable when  $k$  is small, but it also reflects the twin fact that not all related genes are co-expressed and some unrelated genes have similar expression patterns [29]. The key point is that the set  $\{p21, MDM2, ATF3, OHO\}$  forms a good predictor set for the probabilistic gene regulatory network.

In [36], the following statement was made regarding *p21* ( $x_{11}$ ): “as *p21* shows both *p53* dependent and *p53* independent regulation in response to genomic damage [23], it was expected that the *p53* component would not be recognized by the algorithm.” And it was not. The algorithm chose  $\{MDM2, ATF3\}$  as the best predictor set with perceptron CoD 0.356. In the present study, although not having a high CoD itself, *p21* ( $x_{11}$ ) was recognized by the MCMC approach as contributing to strong determination when combined with other genes in Table 9. Specifically,  $\{p53, OHO\}$  has MCMC CoD 0.8044. The much lower CoD for the corresponding perceptron indicates that the relationship between  $\{p53, OHO\}$  and *p21* is highly nonlinear. In some nonlinear fashion, *OHO*, whose definition involves *p53* and *MDM2*, is working in conjunction with *p53* to provide strong prediction of *p21*. The Table 9 also shows the four-gene predictor set  $\{p53, OHO, MDM2, ATF3\}$ , for which the MCMC CoD appears lower due to error estimation based on such a small sample.

The biological expectation is that *MDM2* ( $x_{10}$ ) is incompletely, but significantly, predicted by *p53* ( $x_{12}$ ). As shown in Table 10, this expectation is met in all sets, where selected predictor sets include *p53*. It is useful to note the consistency of the result here with that in [36], where *p53* acting alone has CoD 0.473, and it was recognized that using more genes in conjunction with *p53* does not help. As discussed in [36] (see also [15]), in the case of small samples the addition of predictors can result in poor performance on the test data because larger numbers of predictors require larger samples to avoid overfitting of the training data. Thus, in [36], whenever the CoD of a predictor set is estimated to be less than the CoD of a subset, the CoD of the predictor set is assumed to the higher value (see [36] for relevant statistical remarks). Mutual-information clustering

Table 9  
(a) Regulatory rules for gene  $x_{12}$  ( $p53$ )

	Parent sets	Reversible jump MCMC		Perceptron	
		CoD	$c_{l(12)}^{(12)}$	CoD	$c_{l(12)}^{(12)}$
$X_1^{(12)}$	$\{x_{13}\}$	0.5532	0.4764	0.5532	0.5051
$X_2^{(12)}$	$\{x_5, x_{10}, x_{11}, x_{13}\}$	0.6080	0.5236	0.5421	0.4949

(b) Regulatory rules for gene  $x_{11}$  ( $p21$ )

	Parent sets	Reversible jump MCMC		Perceptron	
		CoD	$c_{l(11)}^{(11)}$	CoD	$c_{l(11)}^{(11)}$
$X_1^{(11)}$	$\{x_8\}$	0.1233	0.0752	0.1233	0.2813
$X_2^{(11)}$	$\{x_{12}, x_{14}\}$	0.8044	0.4904	0.1117	0.2548
$X_3^{(11)}$	$\{x_5, x_{10}, x_{12}, x_{14}\}$	0.7125	0.4344	0.2033	0.4639

Table 10  
Regulatory rules for gene  $x_{10}$  ( $MDM2$ )

	Parent sets	Reversible jump MCMC		Perceptron	
		CoD	$c_{l(10)}^{(10)}$	CoD	$c_{l(10)}^{(10)}$
$X_1^{(10)}$	$\{x_{12}, x_{14}\}$	0.3851	0.3463	0.3851	0.2326
$X_2^{(10)}$	$\{x_2, x_4, x_7, x_{12}\}$	0.2492	0.2241	0.2441	0.2441
$X_3^{(10)}$	$\{x_1, x_2, x_3, x_4, x_6, x_7, x_9, x_{12}\}$	0.1548	0.1392	0.1548	0.1548
$X_4^{(10)}$	$\{x_5, x_{11}, x_{12}, x_{13}\}$	0.3230	0.2904	0.3230	0.3230

greatly reduces the computational burden, thereby allowing larger predictor sets that potentially possess greater CoDs; however, it might result in underestimation of the CoD owing to small samples. Since computation of single-predictor CoDs is not burdensome, one can use those values as minima when making estimates using mutual-information clustering and MCMC. Moreover, since computation of two-predictor CoDs is not overly burdensome (even for several hundred genes), one might also separately compute all of those. If this is done using perceptron prediction, then one is assured that MCMC prediction with more genes has at least as high a CoD, even though that might not be reflected in a small sample.

Lastly, we consider  $BCL3$  ( $x_2$ ), which was shown to be strongly predicted by various predictor sets in [36]

and for which the highest unconstrained (full-logic) predictor had CoD 0.655 with genes  $RCH1$  ( $x_1$ ),  $SSAT$  ( $x_9$ ), and  $p21$  ( $x_{11}$ ). As seen in Table 11,  $\{RCH1, SSAT, p21\}$  is one of the parent sets found by our approach, and it has MCMC CoD 0.6256. Note also in the table the parent set  $\{RCH1, SSAT\}$ , which is a subset of  $\{RCH1, SSAT, p21\}$ . Also note the last set in the table. It contains  $RCH1$  and  $SSAT$ , but it achieves a high CoD using 8 genes, something that would not have been accomplished were we restricted to 3 or 4 predictor genes.

## 7. Conclusions

In this study, we have developed a procedure for constructing gene regulatory networks, e.g.,

Table 11  
Regulatory rules for gene  $x_2$  (BCL3)

	Parent sets	Reversible jump MCMC		Perceptron	
		CoD	$c_{l(2)}^{(2)}$	CoD	$c_{l(2)}^{(2)}$
$X_1^{(2)}$	$\{x_1, x_9\}$	0.4096	0.1735	0.4104	0.1872
$X_2^{(2)}$	$\{x_1, x_9, x_{11}\}$	0.6256	0.2650	0.4556	0.2078
$X_3^{(2)}$	$\{x_4, x_9\}$	0.3628	0.1537	0.3611	0.1647
$X_4^{(2)}$	$\{x_4, x_6, x_9, x_{10}\}$	0.3578	0.1516	0.3578	0.1632
$X_5^{(2)}$	$\{x_1, x_3, x_4, x_6, x_7, x_8, x_9, x_{14}\}$	0.6047	0.2562	0.6072	0.2770

probabilistic gene regulatory networks, using the following approach. First, the number of possible parent gene sets and the input set of variables corresponding to each gene are determined via a novel clustering technique based on mutual information minimization, with simulated annealing being employed to solve the optimization problem. Second, each function is then modelled by a perceptron consisting of a linear term and a nonlinear term. A reversible jump Markov chain Monte Carlo (MCMC) method is used to calculate the model order and the parameters. Finally, the coefficient of determination is employed to compute the probability of selecting different predictors for each gene. To test this approach for constructing probabilistic gene regulatory networks, we have applied our approach to microarray data from known gene response pathways including ionizing radiation and downstream targets of inactivating gene mutations.

### Appendix A. Operations in the reversible jump MCMC annealing algorithm

Suppose that the current state of the Markov chain is in  $(J, \Theta_J)$ .

#### Birth jump

- Propose a new RBF center randomly from the interval in (20).

- Compute

$$r_{\text{birth}} = \left[ \left( \frac{\mathbf{y}^T \mathbf{P}_J^* \mathbf{y}}{\mathbf{y}^T \mathbf{P}_{J+1}^* \mathbf{y}} \right)^{N/2} \right] \frac{1.5d}{N(J+1)},$$

where  $\mathbf{P}_J^*$  is given in (23).

- If  $u \sim \mathcal{U}_{[0,1]} \leq \min(1, r_{\text{birth}})$  then the state becomes  $(J+1, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{J+1})$ , otherwise it remains unchanged.

#### Death jump

- Randomly remove one of the  $J$  RBF terms in (17).
- Compute

$$r_{\text{death}} = \left[ \left( \frac{\mathbf{y}^T \mathbf{P}_J^* \mathbf{y}}{\mathbf{y}^T \mathbf{P}_{J-1}^* \mathbf{y}} \right)^{N/2} \right] \frac{J}{1.5dN}.$$

- If  $u \sim \mathcal{U}_{[0,1]} \leq \min\{1, r_{\text{death}}\}$  then the state becomes  $(J-1, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{J-1})$ , otherwise it remains unchanged.

#### Split jump

- Randomly choose one of the  $J$  RBF terms.
- Replace it by two RBF terms as  $\boldsymbol{\mu}_1 = \boldsymbol{\mu} - u\boldsymbol{\zeta}$  and  $\boldsymbol{\mu}_2 = \boldsymbol{\mu} + u\boldsymbol{\zeta}$ , where  $\boldsymbol{\zeta}$  is a constant ( $\boldsymbol{\zeta} = 0.01$  in our experiments) and  $u \sim \mathcal{U}_{[0,1]}$ . The new centers must lie in the space  $\Omega_k$  in (20).
- Compute

$$r_{\text{split}} = \left[ \left( \frac{\mathbf{y}^T \mathbf{P}_J^* \mathbf{y}}{\mathbf{y}^T \mathbf{P}_{J+1}^* \mathbf{y}} \right)^{N/2} \right] \frac{J\boldsymbol{\zeta}}{N(J+1)}.$$

- If  $u \sim \mathcal{U}_{[0,1]} \leq \min\{1, r_{\text{split}}\}$  then the state becomes  $(J + 1, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{J+1})$ , otherwise it remains unchanged.

*Merge jump*

- Randomly choose one of the  $J$  RBF term, say  $\boldsymbol{\mu}_1$ . Then find its closet neighbor, say  $\boldsymbol{\mu}_2$ .
- If  $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\| < 2\zeta$ , replace the two RBF terms by a single RBF term  $\boldsymbol{\mu} = (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)/2$ .
- Compute

$$r_{\text{merge}} = \left[ \frac{\left( \frac{\mathbf{y}^T \mathbf{P}_j^* \mathbf{y}}{\mathbf{y}^T \mathbf{P}_{j-1}^* \mathbf{y}} \right)^{N/2}}{\zeta N (J - 1)} \right]$$

- If  $u \sim \mathcal{U}_{[0,1]} \leq \min\{1, r_{\text{merge}}\}$  then the state becomes  $(J - 1, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{J-1})$ , otherwise it remains unchanged.

*No jump*

For  $j = 1, 2, \dots, J$

- Draw a sample  $u \sim \mathcal{U}_{[0,1]}$ ;
- If  $u < 0.5$ 
  - Randomly sample a RBF center  $\boldsymbol{\mu}_j^*$  from a fixed initial interval in (20).
  - Compute  $r_{\text{update}} = \left( \frac{\mathbf{y}^T \mathbf{P}_j^* \mathbf{y}}{\mathbf{y}^T \mathbf{P}_j^* \mathbf{y}} \right)^{N/2}$ , where  $\mathbf{P}_j^*$  is similar to  $\mathbf{P}_j^*$  with  $\{\boldsymbol{\mu}_i\}_{i=1}^J$  replaced by  $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_{j-1}, \boldsymbol{\mu}_j^*, \boldsymbol{\mu}_{j+1}, \dots, \boldsymbol{\mu}_J\}$ .
  - If  $v \sim \mathcal{U}_{[0,1]} \leq \min\{1, r_{\text{update}}\}$  then the state becomes  $(J, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{j-1}, \boldsymbol{\mu}_j^*, \boldsymbol{\mu}_{j+1}, \dots, \boldsymbol{\mu}_J)$ , otherwise it remains unchanged.
- If  $u \geq 0.5$ 
  - Randomly sample a RBF center  $\boldsymbol{\mu}_j^*$  from the distribution  $\boldsymbol{\mu}_j^* | \boldsymbol{\mu}_j \sim \mathcal{N}(\boldsymbol{\mu}_j, 0.001 \mathbf{I}_d)$ .
  - Compute

$$r_{\text{update}} = \left( \frac{\mathbf{y}^T \mathbf{P}_j^* \mathbf{y}}{\mathbf{y}^T \mathbf{P}_j^* \mathbf{y}} \right)^{N/2}$$

- If  $v \sim \mathcal{U}_{[0,1]} \leq \min\{1, r_{\text{update}}\}$  then the state becomes  $(J, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{j-1}, \boldsymbol{\mu}_j^*, \boldsymbol{\mu}_{j+1}, \dots, \boldsymbol{\mu}_J)$ , otherwise it remains unchanged.

**Appendix B. CoD estimation**

The CoD must be redefined when the error of the best constant predictor is 0 for a random data split

during cross-validation estimation of the CoD. Let  $e_0$  denote this error and  $e_{\text{opt}}$  denote the error of the optimal predictor, so that the CoD is given by

$$\theta = \frac{e_0 - e_{\text{opt}}}{e_0} \tag{29}$$

If  $e_0 = 0$ , this definition is invalid. To redefine the CoD in this case, we employ two pieces of prior knowledge concerning the theoretical values that should be applied:

$$e_{\text{opt}} \leq e_0, \tag{30}$$

and

$$e_0 > 0. \tag{31}$$

The second condition in (31) holds because we only consider genes that change across the samples. Let us now reconsider the four possibilities for the empirical values:

- (1)  $0 < e_{\text{opt}} \leq e_0$ : We do not use the prior knowledge, and the CoD definition applies directly.
- (2)  $0 < e_0 < e_{\text{opt}}$ : We employ the condition in (30) and set  $e_{\text{opt}} = e_0$ . Then  $\theta = 0$  according to the original definition.
- (3)  $e_0 = e_{\text{opt}} = 0$ : We employ the condition in (31) and set  $e_0 = z$ , where  $z$  is an unknown small positive number. Then  $\theta = 1$  according to the original definition.
- (4)  $0 = e_0 < e_{\text{opt}}$ : By the condition in (31), we set  $e_0 = z$ , as in the previous case, so that
 
$$\theta = \frac{z - e_{\text{opt}}}{z} \tag{32}$$

We now let  $z \rightarrow 0$  (which could have been done in the previous case but was unnecessary because the quotient reduced to case 1). As  $z \rightarrow 0$ , it will become less than  $e_{\text{opt}}$ , in which case, by the condition in (30), we set  $e_{\text{opt}} = z$ , so that  $\theta = 0$  for very small  $z$ . Hence the limit yields  $\theta = 0$ .

The difference between this redefinition and that taken in [36] is that there, the CoD was set to 0 in the third case,  $e_0 = e_{\text{opt}} = 0$ .

**References**

[1] E.H.L. Aarts, H.L. Emile, Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial

- Optimization and Neural Computing, Wiley, Chichester, England; New York, 1989.
- [2] T. Akutsu, S. Miyano, S. Kuhara, Identification of genetic networks from a small number of gene expression patterns under Boolean network model, Pacific Symposium on Biocomputing, Vol. 4, Hawaii, 1999, pp. 17–28.
- [3] T. Akutsu, S. Miyano, S. Kuhara, Inferring qualitative relations in genetic networks and metabolic pathways, *Bioinformatics* 16 (2000) 727–743.
- [4] S.A. Amundson, M.L. Bittner, Y. Chen, J.M. Trent, P. Meltzer, A.J. Fornace, Fluorescent cDNA microarray hybridization reveals complexity and heterogeneity of cellular genotoxic stress response, *Oncogene* 18 (1999) 3666–3672.
- [5] C. Andrieu, J. Freitas, A. Doucet, Robust full Bayesian learning for neural networks, <http://www.cs.berkeley.edu/~jfgf/software.html>, 1999.
- [6] A. Ben-Dor, R. Shamir, Z. Yakhini, Clustering gene expression patterns, *Comput. Biol.* 6 (3/4) (1999) 281–297.
- [7] M. Bittner, P. Meltzer, P. Khan, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dor, E. Dougherty, E. Wang, F. Marincola, C. Gooden, J. Lueders, A. Glatfelter, P. Pollock, E. Gillanders, A. Leja, K. Dietrich, C. Beaudry, M. Berrens, D. Alberts, V. Sondak, N. Hayward, J.M. Trent, Molecular classification of cutaneous malignant melanoma by gene expression profiling, *Nature* 406 (2000) 536–540.
- [8] A.J. Butte, I.S. Hohane, Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements, Pacific Symposium on Biocomputing, Vol. 4, Hawaii, 2000.
- [9] Y. Chen, E.R. Dougherty, M. Bittner, Ratio-based decisions and the quantitative analysis of cDNA microarray images, *J. Biomed. Opt.* 2 (4) (1997) 364–374.
- [10] T. Chen, V. Filkov, S. Skiena, Identifying gene regulatory networks from experimental data, in: Proceedings of the RECOMB, 1999, pp. 94–103.
- [11] E. Cinlar, Introduction to Stochastic Process, Prentice-Hall, Englewood Cliffs, NJ, 1997.
- [12] J.M. Claverie, Computational methods for the identification of differential and coordinated gene expression, *Hum. Mol. Genetics* 8 (10) (1998) 1821–1832.
- [13] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley, New York, 1991.
- [14] C. Debouck, P.N. Goodfellow, DNA microarrays in drug discovery and development, *Nat. Genet.* 21 (1) (1999) 48–50.
- [15] E.R. Dougherty, Small sample issues for microarray-based classification, *Comp. Functional Genomics* 2 (2001) 28–34.
- [16] E.R. Dougherty, J. Barrera, M. Brun, S. Kim, R.M. Cesar, Y. Chen, M. Bittner, J.M. Trent, Inference from clustering: application to gene-expression time series, *Comput. Biol.* 9 (1) (2002).
- [17] E.R. Dougherty, M.L. Bittner, Y. Chen, S. Kim, K. Sivakumar, J. Barrera, P. Meltzer, J.M. Trent, Nonlinear filters in genomic control, Proceedings of the IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing, Antalya, Turkey, 1999.
- [18] E.R. Dougherty, S. Kim, Y. Chen, Coefficient of determination in nonlinear signal processing, *Signal Processing* 80 (10) (2000) 2219–2235.
- [19] M.B. Eisen, P.T. Spellman, P.O. Brown, D. Botstein, Cluster analysis and display of genome-wide expression patterns, Proceedings of the National Academy of Science USA, 1998, pp. 14863–14868.
- [20] N. Friedman, M. Linial, I. Nachman, D. Pe'er, Using Bayesian network to analyze expression data, *J. Comput. Biol.* 7 (2000) 601–620.
- [21] N. Friedman, I. Nachman, D. Pe'er, Learning Bayesian network structure from massive data sets: the “Sparse Candidate” algorithm, Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI), Stockholm, 1999.
- [22] A.J. Glass, S.A. Kauffman, The logical analysis of continuous, non-linear biochemical networks, *J. Theor. Biol.* 39 (1973) 103–129.
- [23] M. Gorospe, S. Shack, K.Z. Guyton, D. Samid, N.J. Holbrook, Up-regulation and functional role of *p21* Wal/Cip 1 during growth arrest of human breast carcinoma MCF-7 cells by phenylacetate, *Cell Growth Differ* 7 (1996) 1609–1615.
- [24] P.J. Green, Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, *Biometrika* 82 (1995) 711–732.
- [25] P. D’Haeseleer, P. Liang, S. Fuhrman, R. Somogyi, Genetic network inference: from co-expression clustering to reverse engineering, *Bioinformatics* 16 (8) (2000) 707–726.
- [26] P. D’Haeseleer, X. Wen, S. Fuhrman, R. Somogyi, Linear modeling of mRNA expression levels during CNS development and injury, Pacific Symposium on Biocomputing, Vol. 4, 1999, pp. 41–52.
- [27] A.J. Hartemink, D.K. Gifford, T.S. Jaakkola, R.A. Young, Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks, Pacific Symposium on Biocomputing, Vol. 6, 2001, pp. 23–32.
- [28] J. Hasty, D. McMillen, F. Isaacs, J.J. Collins, Computational studies of gene regulatory networks: in numero molecular biology, *Nat. Reviewers Genet.* 2 (2001) 268–279.
- [29] L.J. Heyer, S. Kruglyak, S. Yooseph, Exploiting expression data: identification and analysis of coexpressed genes, *Genome Res.* 9 (1999) 1106–1115.
- [30] C.C. Holmes, B.K. Mallick, Bayesian wavelet network for nonparametric regression, *IEEE Trans. Neural Networks* 11 (1) (2000) 27–35.
- [31] S. Huang, Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis drug biology, *J. Mol. Med.* 77 (1999) 469–480.
- [32] L. Ingber, B. Rosen, Genetic algorithms and very fast simulated reannealing: a comparison, *Math. Comput. Modelling* 16 (16) (1992) 87–100.
- [33] V.R. Iyer, M.B. Eisen, D.T. Ross, G. Schuler, T. Moore, J.C.F. Lee, J.M. Trent, L.M. Staudt, J. Hudson, M.S. Boguski, D. Lakhari, D. Shalon, D. Botstein, P.O. Brown, The transcriptional program in the response of human fibroblasts to serum, *Science* 283 (5398) (1999) 83–87.

- [34] S.A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, *J. Theor. Biol.* 22 (1969) 437–467.
- [35] S.A. Kauffman, *The Origins of Order: Self-organization and Selection in Evolution*, Oxford University Press, New York, 1993.
- [36] S. Kim, E.R. Dougherty, M.L. Bittner, Y. Chen, K. Sivakumar, P. Meltzer, J.M. Trent, General nonlinear framework for the analysis of gene interaction via multivariate expression arrays, *J. Biomed. Opt.* 5 (4) (2000) 411–424.
- [37] S. Kim, E.R. Dougherty, Y. Chen, K. Sivakumar, P. Meltzer, J.M. Trent, M. Bittner, Multivariate measurement of gene expression relations, *Genomics* 67 (2000) 201–209.
- [38] T. Mestl, E. Plahte, S.W. Omholt, A mathematical framework for describing and analysing gene regulatory networks, *J. Theor. Biol.* 176 (2) (1995) 291–300.
- [39] E.J. Moler, D.C. Radisky, I.S. Mian, Integrating naive Bayes models and external knowledge to examine copper and iron homeostasis in *S. cerevisiae*, *Physiol. Genomics* 4 (2000) 127–135.
- [40] K. Murphy, S. Mian, Modelling gene expression data using dynamic Bayesian networks, Technical Report, University of California, Berkeley, <http://www.berkeley.edu/~murphyk/publ>, 1999.
- [41] I. Shmulevich, E.R. Dougherty, S. Kim, W. Zhang, Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks, *Bioinformatics* 18 (1) (2002).
- [42] I. Shmulevich, O. Yli-Harja, J. Astola, Inference of genetic regulatory networks under the best-fit extension paradigm, *Proceedings of the IEEE-EURASP Workshop on Nonlinear Signal and Image Processing*, Baltimore, MD, 2001.
- [43] P. Smolen, D. Baxter, J. Byrne, Mathematical modeling of gene networks, *Neuron* 26 (2000) 567–580.
- [44] E.B. Suh, E.R. Dougherty, S. Kim, D.E. Russ, R. Martina, Parallel computing methods for analyzing gene expression relationships, *Proceedings of the SPIE Microarrays: Optical Technologies and Informatics*, San Jose, 2001.
- [45] E.P. van Someren, L.F.A. Wessels, M.J.T. Reinders, Linear modeling of genetic networks from experimental data, 8th International Conference on Intelligent Systems for Molecular Biology, San Diego, 2000.
- [46] D.C. Weaver, C.T. Workman, G.D. Stormo, Modeling regulatory networks with weight matrices, *Pacific Symposium on Biocomputing* 3 (1999) 112–123.