Simulation Study in Probabilistic Boolean Network Models for Genetic Regulatory Networks[*]

Shu-Qin Zhang[†] Wai-Ki Ching [‡] Michael K. Ng[§] Tatsuya Akutsu [¶]

June 13, 2006

Abstract

Interactions between different genes become more and more important in understanding how they collectively make cells, tissues, organisms, even form a biological system. This gives rise to genetic regulatory networks. Mathematical models and computational methods are widely used to analyze them recently. Probabilistic Boolean network is one proposed to model such systems incorporating uncertainty. Evolution of the system is according to the transition probability matrix. Steady-state (long run behavior) analysis is a key aspect in studying the dynamics of genetic regulatory networks. In this paper, an efficient method to construct the sparse transition probability matrix is proposed, and the power method based on the sparse matrix-vector multiplication is applied to compute the steady-state probability distribution. Such methods provide a tool for us to study the sensitivity of the steady-state distribution to influence of input genes, gene connections and Boolean networks. Simulation results based on a real network are given to illustrate the method and to demonstrate the steady-state analysis.

Keywords: Genetic regulatory network, Probabilistic Boolean network, Steady-state probability distribution, Sensitivity, Robustness

[*]The corresponding author of this paper is Michael K. Ng.

[†]Advanced Modeling and Applied Computing Laboratory, Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: sqzhang@hkusua.hku.hk. This paper is done when the author is visiting Bioinformatics Center, Institute for Chemical Research, Kyoto University.

[‡]Advanced Modeling and Applied Computing Laboratory, Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: wkc@maths.hku.hk. Research supported in part by Hung Hing Ying Physical Research Fund, HKU GRCC Grants Nos. 10206647, 10206483 and 10206147.

[§]Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong. E-mail: mng@math.hkbu.edu.hk. Research supported in part by RGC 7046/03P, 7035/04P, 7035/05P and HKBU FRGs.

[¶]Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji-city, Kyoto 611-0011, Japan. E-mail: takutsu@kuicr.kyoto-u.ac.jp. Work partially supported by Grant-in-Aid Systems Genomics from MEXT, Japan.

1 Introduction

The interactions among different genes make the complexity of a living cell. The study of the genes collectively act to make cells, tissues and organisms is an importaint topics in bioinformatics. The advent of high-density cDNA microarrays and oligonucleotide chips opens a new era for this kind of research [6][14][22][23][27]. Instead of looking at one single gene, the global or holistic behavior perspectives become more and more important in understanding the manner in which genes and molecules collectively form a biological system. This gives rise to genetic regulatory networks structured by networks of regulatory interactions among DNA, RNA, proteins and small molecules. Based on microarray data generated, a number of genes and their regulatory sites have been found. In addition, the proteins involved in the control of the gene expressions have been identified. Since the amount of microarray data is huge, it is very expensive to analyze such data in order to understand and extract more useful biological information. Thus the development of mathematical models and computational methods for the construction of formalisms to model the gene interactions is an effective and efficient alternative.

There have been many formalisms proposed in the literature to study genetic regulatory networks, such as directed graphs, Bayesian networks, Boolean networks (BNs) and probabilistic Boolean networks (PBNs), ordinary and partial differential equations, qualitative differential equations and other mathematical models [16]. Differential equations can model and capture the biochemical reactions within genetic networks accurately, but it is required to employ a huge amount of data for model inference. In the biophysics community, BN models, later extended to PBN models have received much attention. BN model is originally introduced by Kauffman [9][17][18][19]. Reviews of BN models can be found in [13][20][35]. In a BN model, gene expression states are quantized to only two levels: on and off (represented as 1 and 0). Even though most biological phenomena manifest them in continuous domain, the binary expression shows promising and useful results [28]. For instance, in cDNA microarrays, the binary gene expression data can be generated by using the Hamming distance as a similarity metric to separate the type of gliomas and the type of sarcomas. Such binary expression data can retain meaningful biological information contained in the real continuous-domain gene expression profiles.

Using BN models, qualitative rather than quantitative relationships underlying genetic regulation and control can be discovered. Furthermore, with this model, many questions about the complex dynamic behavior of large genetic networks in the realistic biological system can be studied and answered [36][37]. In a BN, the target gene is predicted by several genes via a Boolean function. The genes that predict a certain gene are called its input genes. Once the input genes and the Boolean functions are determined, the BN model becomes deterministic. Since the biological system is stochastic in nature and microarray data sets used to infer the model may not be accurate due to experimental noise in the complex measurement processes, a deterministic model is not favorable to such real situations. To develop a model incorporating with uncertainty is necessary. Probabilistic Boolean networks are recently developed and studied in the literature. It is a generalization of BN from a deterministic format to a stochastic one. Detailed explanations of extending BN to PBN can be found in [30].

A PBN contains a family of BNs. In a PBN model, Boolean networks are allowed to switch from one to another with certain probabilities during state transitions. A state can transit into a number of states according to the realization of all Boolean networks at that moment. Thus the dynamics (transitions) of the system can be described by Markov chains. The theory of Markov chains can then be applied to the study of such systems. Once the PBN model is determined, the system will evolve toward its steady-state. In the theory of Markov chains, if a Markov chain is irreducible (reducible), the steady-state probability distribution is independent (dependent) on the initial state probability distribution of a PBN. A Boolean network will settle into a collection of state cycles called *attractors*. The states that can lead the system to a specific attractor is called the *basin of attractors*. Recently, a theoretical analysis of steady-state probabilities for attractors in both BN and PBN are studied and derived [5]. The steady-state probability distribution provides a first-order statistical information of a PBN. Using such information of a PBN, we can understand a genetic network, and identify the influence of different genes in a network. We can further figure out how to control some genes in a network such that the whole system can evolve into a target or desired steady-state probability distribution. Therapeutic gene intervention or gene control policy [7][8][24][31] can be developed and studied according to them. It is obvious that the steady-state probability distribution of a PBN is a very important information to be computed.

The main aim of this paper is to develop an efficient method to compute the steady-state probability distribution of a PBN and then use it to analyze the sensitivity of the steady-state probability distribution in a PBN to the change of input genes, connections between genes and Boolean functions. Markov chain Monte-Carlo (MCMC) method has been proposed in [34] to calculate the steady-state probability distribution of a PBN. This method considers PBN as a Markov chain. By simulating the underlying Markov chain for a sufficiently long time until it converges to the steady-state, one can get the approximation of the steady-state probability distribution. Although it has been shown that MCMC method can perform well in a small PBN, it can be successfully used only if we are sufficiently confident that the system has evolved to its steady-state. Theoretically, a priori bound on the number of iterations is too large to be useful even for a moderate size network [26]. Thus in practice, only empirical determination methods can be used to stop the chain and get the estimate of the steady-state probability distribution [34]. Matrix-based method (as a deterministic method) can obtain the steady-state probability more accurately than MCMC method (as a probabilistic method). Once the transition probability matrix is constructed, all the analysis based on the matrix will become very simple. Hence approaches based on transition probability matrix are more powerful and are still of great importance.

The rest of the paper is organized as follows. In Section 2, a brief review of both BN model and PBN model based on the mathematical expressions is given. In Section 3, the methodology to compute the steady-state probability is introduced. Then, we simulate a system based on a real regulatory network with the PBN model and examine the steady-state probabilities that are influenced by the input genes, the connectivity between genes and the Boolean function in section 4. Finally, in the last section, we make a brief conclusion and address some possible future research issues.

2 Probabilistic Boolean Networks

PBN is a generalization of the BN. Let us first give a brief review on BN. A BN $G(V, F)$ consists of a set of nodes V and Boolean functions F, where

$$V = \{x_1, x_2, \ldots, x_n\}$$

and

$$F = \{f_1, f_2, \ldots, f_n\}.$$

Let $x_i(t)$ represents the state of x_i at time t, where $x_i = 0$ represents that gene is unexpressed and $x_i = 1$ means it is expressed. In the following, we may use Gene i to refer the gene x_i, and x_i refer to the state of it. The overall expression levels of all the genes in the network at time step t is given by the following column vector

$$x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T.$$

This vector is referred to the *Gene Activity Profile* (GAP) of the network at time t. For $x(t)$ ranging from $[0, 0, \ldots, 0]^T$ (all entries are 0) to $[1, 1, \ldots, 1]^T$ (all entries are 1), it takes on all the 2^n possible states of the n genes. The list of Boolean functions represents rules of regulatory interactions among the nodes (genes):

$$x_i(t + 1) = f_i(x(t)), \quad i = 1, 2, \ldots, n.$$

The state of gene i can be predicted by k_i genes, where k_i is the connectivity of gene i. The maximum connectivity of a Boolean network is defined as

$$K = \max\{k_i\}.$$

In general, k_i may not be equal to n, but without loss of generality, we allow the unnecessary variables to be fictitious. The states of all genes can be updated synchronously according to the output of their corresponding Boolean functions. Table 1 is an artificial truth table of a BN. Here each gene will update its state according to the states of other genes in the previous step. The state transition of this BN can be described as in Figure 1. The system will eventually evolve into the state 000 or 111 depending on the initial state.

Table 1: The Truth Table of Boolean Functions in the BN

x_1	x_2	x_3	$f^{(1)}$	$f^{(2)}$	$f^{(3)}$
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	1	1

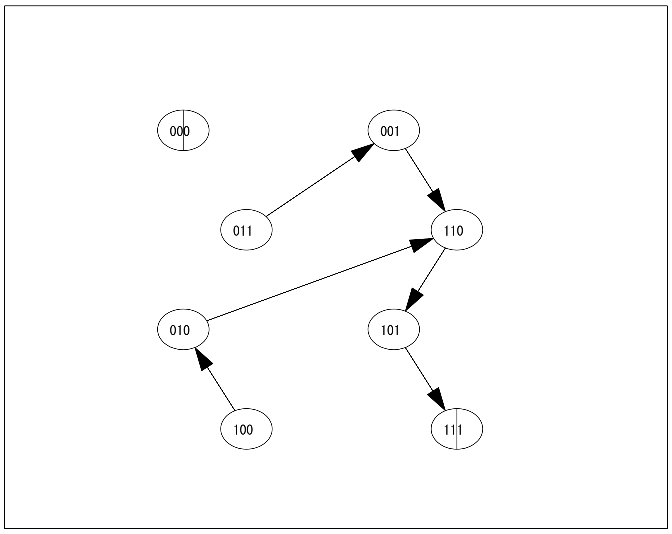

Figure 1: Evolution of the BN in Table 1

To build a BN model, the first step is to identify the network structure (i.e., $G(V,F)$) from real experimental data. A lot of recent works have been focused on this problem [1] [2] [3] [4] [12] [15] [25] [29]. Among all the methods, *coefficient of determination* (COD) is a general statistical approach to uncover the associations among the genes [21]. The coefficient measures the degree to which the input gene set can be used to improve the prediction accuracy of a target gene relative to the best possible prediction in absence of this input gene set. Let x_i be a target gene that we want to predict, and assume all genes can be used to predict it. The problem is how to estimate the state of x_i from the observed genes. Let f be an optimal prediction function of x_i. The COD is defined as follows:

$$\theta = \frac{\epsilon_i - \epsilon_{opt}}{\epsilon_i},$$

where ϵ_i is the error for the best prediction function in the absence of the observations and ϵ_{opt} is optimal error in the presence of the observations achieved by f. Since $\epsilon_{opt} < \epsilon_i$, the COD must be less than one. For example, if the error is the Mean-Square Error (MSE):

$$E[(x_{i_{pred}} - x_i)^2],$$

then the best prediction $x_{i_{pred}}$ of x_i in absence of other observed variables is its mean, and the error is the variance, i.e.

$$\epsilon_i = \sigma^2_{x_i}.$$

The optimal prediction of x_i based on the other observed data is the conditional expectation: $f = E(x_i|x)$. The optimal predicting function f can be determined by using neural networks [21]. With the COD method, the genes that can yield the highest COD will be chosen to be the input genes. But since in practice, some different input gene sets can give high COD, and we cannot constraint the number of sets, a natural way is to incorporate all the input gene sets with some probabilities, which also leads to the PBN.

We recall that PBN is an extension of BN. For each target gene, it allows many Boolean functions that have equivalent prediction abilities. All these Boolean functions can be selected randomly with some probabilities. We assume that for the ith gene, there corresponds $l(i)$ possible Boolean functions:

$$\left\{ f_j^{(i)} : \text{ for } j = 1, \ldots, l(i) \right\}$$

and the probability of selecting function $f_j^{(i)}$ is $c_j^{(i)}$, where $f_j^{(i)}$ is a function with respect to the activity levels of n genes. Here $c_j^{(i)}$ can be computed from COD, and it is proportional to its COD. Since $c_j^{(i)}$ are probabilities, they must satisfy the following

$$\sum_{j=1}^{l(i)} c_j^{(i)} = 1.$$

Thus one can compute it as follows:

$$c_j^{(i)} = \frac{\theta_j^i}{\sum_{j=1}^{l(i)} \theta_j^i},$$

where θ_j^i is the COD for Gene i with respect to the prediction function $f_j^{(i)}$. For a PBN with n genes, there are at most

$$N = \prod_{i=1}^{n} l(i)$$

different possible BNs among these n genes. This means that there are totally N possible realizations of the genetic networks. Let f_j be the jth possible realization,

$$f_j = [f_{j_1}^{(1)}, f_{j_2}^{(2)}, \ldots, f_{j_n}^{(n)}], \quad 1 \le j_i \le l(i), \quad i = 1, 2, \ldots, n.$$

Suppose that P_j is the probability of choosing the jth BN,

$$P_j = \prod_{i=1}^{n} c_{j_i}^{(i)}, \quad j = 1, 2, \ldots, N, \tag{1}$$

and let a and b be any two column vectors with n entries being either 0 or 1. Then

$$\begin{aligned} &\text{Prob } \{x(k+1) = a \mid x(k) = b\} \\ =&\sum_{j=1}^{N} \text{ Prob } \{x(k+1) = a \mid x(k) = b, \text{ where } j\text{th network is selected } \} \cdot P_j. \end{aligned} \tag{2}$$

By letting a and b ranging from $[0, 0, \ldots, 0]^T$ to $[1, 1, \ldots, 1]^T$ independently, we can get the transition probability matrix A. For the ease of presentation, we first transform the n-digit binary number vector, as discussed in [33], into a decimal number by the following formula:

$$y(k) = 1 + \sum_{i=1}^{n} 2^{n-i} x_i(k).$$

As $x(k)$ ranges from $[0, 0, \ldots, 0]^T$ to $[1, 1, \ldots, 1]^T$, $y(k)$ will cover all the values from 1 to 2^n. Since the mapping from $x(k)$ to $y(k)$ is one-to-one, we can just equivalently work with $y(k)$.

Let $w(k)$ be the probability distribution vector at time k, i.e.,

$$w_i(k) = \text{ Prob } \{y(k) = i\}, \quad i = 1, 2, 3, \ldots, 2^n.$$

It is straightforward to check that

$$w(k+1) = Aw(k) \tag{3}$$

where A satisfies

$$\sum_{i=1}^{2^n} [A]_{ij} = 1$$

Table 2: The Truth Table of Boolean Functions in the PBN

x_1 x_2 x_3	$f_1^{(1)}$	$f_2^{(1)}$	$f_1^{(2)}$	$f_1^{(3)}$	$f_2^{(3)}$
0 0 0	0	0	0	0	0
0 0 1	1	1	1	0	0
0 1 0	1	1	1	0	0
0 1 1	1	0	0	1	0
1 0 0	0	0	1	0	0
1 0 1	1	1	1	1	0
1 1 0	1	1	0	1	0
1 1 1	1	1	1	1	1
$c_j^{(i)}$	0.6	0.4	1	0.5	0.5

and it has at most $N \cdot 2^n$ non-zero entries of the 2^n-by-2^n transition probability matrix. Interested readers can consult [33] for more details about PBNs.

In the above BN example, if there is more than one Boolean function for at least one gene, it will construct a PBN model. Table 2 is an example of PBN discussed in [33]. It can be seen as an extension from the above BN model. There are two Boolean functions $f_1^{(1)}$, $f_2^{(1)}$ associated with x_1, one Boolean function $f_1^{(2)}$ associated with x_2, and two Boolean functions $f_1^{(3)}$, $f_2^{(3)}$ associated with x_3. There are totally $2 \times 1 \times 2 = 4$ BNs in this example. The transition probability matrix can be constructed according to the above description. For example, if we want to know where a given a state $[0, 1, 1]$ will go, from the table, one can see that there are four states that $[0, 1, 1]$ can go into: $[1, 0, 1], [1, 0, 0], [0, 0, 1], [0, 0, 0]$, and the probabilities are given respectively by

$$\begin{aligned}
[0,1,1] &\rightarrow [1,0,1] : 0.6 \times 1 \times 0.5 = 0.3, \\
[0,1,1] &\rightarrow [1,0,0] : 0.6 \times 1 \times 0.5 = 0.3, \\
[0,1,1] &\rightarrow [0,0,1] : 0.4 \times 1 \times 0.5 = 0.2, \\
[0,1,1] &\rightarrow [0,0,0] : 0.4 \times 1 \times 0.5 = 0.2.
\end{aligned}$$

The state transition of this PBN can be found in [33].

3 Computation of the Steady-state Probability Distributions

In this section, we first discuss the method of generating the transition probability matrix and then present the power method for computing the steady-state probability distribution.

3.1 Generation of Transition Probability Matrix

In the paper by Shmulevich et al. [33], the matrix A is constructed by computing all the entries one by one. It is based on the index of all the entries. For each entry, all the BNs should be considered to determine whether the network contributes to it or not. For example, if we want to compute $A(i,j)$, we need to consider if the first BN is applied, whether the state j will transition into state i. We then consider the second BN, and so on. After this, we need to consider $A(i,j+1)$ or $A(i+1,j)$ depending on the choice of row index or column index. Even the entry is zero, the process is still necessary to compute it. The value of $A(i,j)$ is the sum of probabilities of the BNs that can lead j to i. Since the transition probability matrix is sparse in practice, much time are spent to compute the zero entries. This involves many unnecessary computations and therefore it is inefficient.

One way to save time in computing the zero entries is that we only consider the nonzero entries. The method is based on the state space. Given a state i, if a specific Boolean function can lead it to state j, then $A(j,i)$ will have value corresponding to the probability of this BN. If another BN also can lead i to j, then the probability will be greater by the probability corresponding to the BN. Although this is only an improvement in computing the transition probability matrix, it can save much time and makes significant progress in computing the steady-state probability.

In Shmulevich's method, every entry and every BN should be considered, the complexity of the method is about $O(N2^{2n})$. For our proposed method, only the states and the BNs will be considered and it is of $O(N2^n)$. As the number the genes increases, the time saved will be significant.

3.2 Computation of the Steady-state Probability Distribution

To obtain the steady-state probability distribution of a PBN, power method is a good choice. It is an iterative method for solving the largest eigenvalue in modulus (the dominant eigenvalue) and its corresponding eigenvector [11]. If we assume the underlying Markov chain of the PBN is irreducible, then the maximum eigenvalue of the transition probability matrix is one and the modulus of the other eigenvalues are less than one. Moreover, the eigenvector corresponding to the maximum eigenvalue is the steady-state probability distribution. Given an initial vector $\mathbf{x}^{(0)}$, one can compute

$$\mathbf{x}^{(k)} = A\mathbf{x}^{(k-1)}$$

until

$$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_\infty < \epsilon$$

satisfies some tolerance ϵ. Here $\mathbf{x}^{(k)}$ is the eigenvector corresponding to eigenvalue one, i.e. the steady-state of the system with transition probability matrix A. The main computational cost of this method comes from the matrix-vector multiplications. Here, since the matrix A is sparse, the sparse matrix multiplication is applied. The convergence rate of the power method depends on the ratio of $|\lambda_2/\lambda_1|$ where λ_1 and λ_2 are respectively the largest and the second largest eigenvalue of the matrix A. Our computational results indicate that even when there are 16 genes, the steady-state probability distribution can be obtained within a minute within 100 iterations with the tolerance $\epsilon = 10^{-10}$. In the next section, we will present more results on the computational time required by our proposed method.

4 Simulation Study

All the simulation results are described based on the real network in [34]. Since Gene (TOP2A) is only the input gene of Gene ((SCYB10);(INP10);IP10) and the indegree of Gene (TOP2A) is zero, this gene is not considered in the simulation study. The total number of genes studied in the network is 14. The total number of possible states in the network is therefore equal to 2^{14}, i.e., 16384. For the ease of the presentation, we refer these fourteen genes to be the numbers 1 up to 14 in the following description, their corresponding names can be found in Table 3. Figure 2 describes the structure of the network. If two genes are the input gene of each other, the edge in the graph is undirected.

In the simulation study, the number of Boolean functions and input genes in a Boolean network are set to be no more than three. The input genes in a Boolean function are randomly selected from the input gene set of the target gene. The Boolean functions are generated randomly. In the following discussion we present the simulation results based on a particular setting. Detailed information of Boolean functions and input genes for such setting can be found in Tables 13, 14 and 15 of the Appendix. However, we have simulated a number of cases, and we would like to mention that similar phenomenon is observed in other simulation cases for the real genetic network in Figure 2.

In the following, we may use decimal number to describe the n-digit binary state. The method of transferring a binary state to a decimal number has been described in Section 2. For example, state 596 represents the state $[0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1]$. From the left to the right, it represents the state from Gene 1 to Gene 14.

4.1 Classification of States in a Network

The states can be divided into three different types: unreachable state, transient state and recurrent state. Unreachable state is the state that the system will never get into. In the transition probability matrix, there are many zero rows. Each zero row corresponds to a zero entry in the steady-state probability distribution. If the jth row in the transition matrix is full of zero entries, the jth entry in the steady-state probability distribution must be zero. These states are the unreachable states. A transient state is the state that the system can get into before a certain time, but after that time, the system will never re-visit it. When a row in

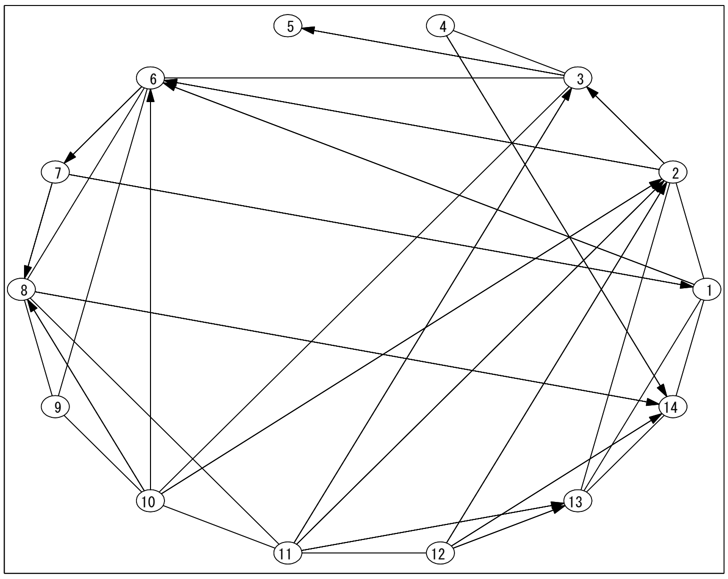

Figure 2: The Real Genetic Regulatory Network

Table 3: Names of 14 Genes in the Real Network

1	Tie-2
2	TGF-beta3; TGFB3
3	ERCC1
4	(HSP40); (HDJ1; DNAJ1)
5	(TDPX2); (PAG); (NKEFA)
6	(GSTP1); GST3; (FAEES3)
7	GNB1
8	(NDP kinase B; NDKB); (NME2);(PUF);NM23B
9	(SCYB10); (INP10); IP10
10	PDGFA; PDGFI
11	(NKEFB); (TSA); (TDPXI)
12	Beta Actin
13	NFKB1; KBF1
14	(BCL2A1); BFL1 protein; GRS protein

the transition probability matrix has nonzero entries, and the corresponding probability in the steady-state probability distribution is zero, then this row corresponds to a transient state in the system. The remaining states are the recurrent states.

Since the zero rows can be identified from the transition probability matrix, the unreachable states can be determined. Some of the unreachable states can be identified from the Boolean functions directly. Given two target genes which are predicted by the same gene, when this particular gene is in one certain state on or off, and the two target genes can be in only one state on or off, then the rows involving the state that the two target genes cannot get into at the same time will be zero. For example, in the real network structure, both Gene 4 and Gene 5 are predicted by Gene 3. Given the Boolean functions

$$x_4 = \begin{cases} f^4 = 0, & \text{if} \quad x_3 = 0; \\ f^4 = 1, & \text{if} \quad x_3 = 1, \end{cases}$$

and

$$x_5 = \begin{cases} f^5 = 1, & \text{if} \quad x_3 = 0; \\ f^5 = 0, & \text{if} \quad x_3 = 1, \end{cases}$$

the rows corresponding to the states $x_4 = x_5$ are all zeros. Then the system cannot evolve into the states corresponding to those rows. Figure 3 shows the steady-state probability distribution corresponding to the above Boolean functions for genes 4 and 5, all the other Boolean functions are set according to Tables 13, 14 and 15 of the Appendix. In the steady-state probability distribution, all the entries containing $(x_4 = 0, x_5 = 0)$ and $(x_4 = 1, x_5 = 1)$ are zero.

Some genes are not connected directly in the network, but they can influence each other through some other genes during the evolution process. So many unreachable states cannot be identified only from the Boolean functions themselves. Figure 4 presents a part of the steady-state probability distribution from State 513 to State 640 of Figure 3. Some other zero entries appear, which indicates that some particular states of the genes cannot coexist at the same time. In the figure, from State 577 to State 596 the probabilities are all zeros, these states correspond to the following situations

$$[\cdot,\cdot,\cdot,0,1,\cdot,\cdot,1,\cdot,0,\cdot,0,\cdot,\cdot]$$
$$[\cdot,\cdot,\cdot,0,1,\cdot,\cdot,1,\cdot,0,\cdot,1,\cdot,\cdot]$$
$$[\cdot,\cdot,\cdot,0,1,\cdot,\cdot,1,\cdot,1,\cdot,0,\cdot,\cdot],$$

i.e., Gene 4 is off, Gene 5 and 8 are on, and either Gene 10 and 12 are off, Gene 10 is off and Gene 12 is on, or Gene 10 is on and Gene 12 is off. From the numerical results, the probability of all the states including these three cases are zero.

Transient states and recurrent states must be determined from the steady-state probability distribution. Figure 5 is all the transient states in the system. The number of the transient states is 150. In the figure, the number corresponding to the nonzero values is the transient state.

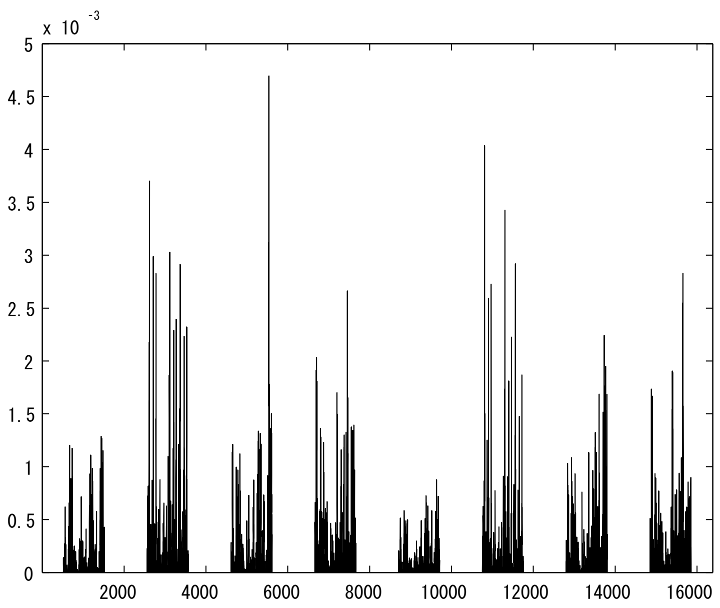

Figure 3: Steady-state Probability Distribution of the Genetic Network

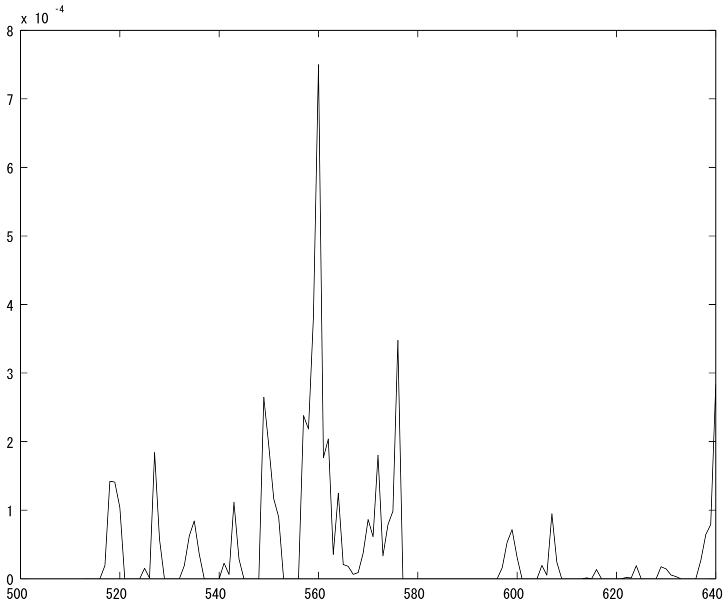

Figure 4: Steady-state Probability Distribution from State 513 to State 640

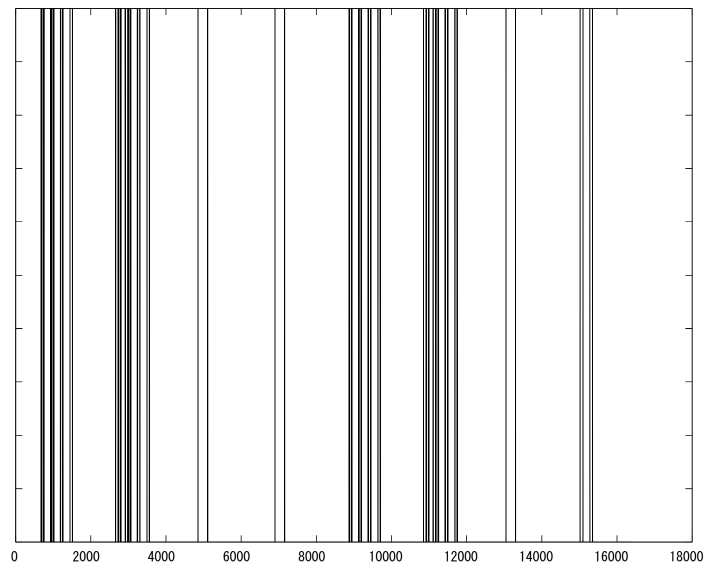

Figure 5: Transient States in the Genetic Network

We remark that the possible states that can transit to the transient states are all unreachable states. If the initial state is an unreachable state, transient states will be visited.

Recurrent states are the states that the system will revisit with certain probabilities. One interpretation of recurrent states is that the cellular types are characterized by stable recurrent patterns of gene expression. Another interpretation is that in a Boolean network, the steady states specify distinct cell states defined by patterns of gene activity. Each realization of a PBN can induce many attractors. During the evolution process, after certain time, the system will transition within the states of all the attractors in the whole PBN with some probabilities. We note that the total number of recurrent states is 4450, which is only about 25% of all the 2^{14} states.

4.2 Sensitivity and Robustness of Steady-state Probability Distribution in a PBN

In this section, the sensitivity of the steady-state probability distribution to the influence of gene connections, Boolean functions and input genes is studied.

4.2.1 Influence of Gene Connections

Connections between genes describe whether the genes are predicted by other genes or they predict others. Genetic regulatory network can be seen as a directed graph and the direction is towards the target gene from the input genes. If the connection between some genes is broken, the state of the target gene will not be related to the broken gene directly. In practice, this can

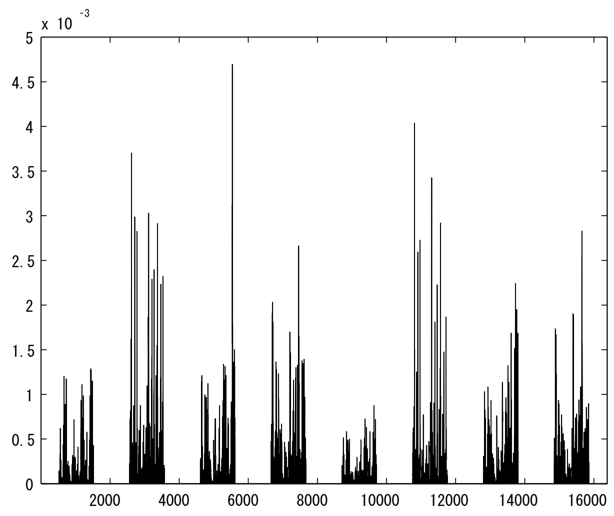

Figure 6: Steady-state Probability Distribution When Gene 2 Is Not Applied to Predict Gene 6

Table 4: Influence of Gene Connections: Number of Recurrent States, Where the Original Number of Recurrent States Is 4450.

Connections Broken	14→1	13→2	4→3	2→6	10→11	11→13	13 →14
Recurrent States	4450	4264	4375	4420	4450	4457	2269

be done by using some drugs or therapeutic methods. The target gene will evolve according to the states of the rest input genes. In our tests, when a connection is broken, the Boolean function is chosen such that the difference of output between the new Boolean function and the previous one is as small as possible.

Table 4 shows the number of recurrent states when some connections are broken. After one connection is broken, the number is changed. The number can be greater than or less than the original number of recurrent states. Connections among genes not only change the number of recurrent states, but also change their probabilities. Since a state in a system is a combination of the status of all genes, the connection can in fact make an influence on some genes in the whole genetic network.

Genes that are influenced greatly by breaking the connection can be identified from the setting of the Boolean functions. If we can break a connection for a certain gene and the break can make this gene evolve towards a state with higher probability from the setting of the Boolean functions, this gene can follow the rule in practice. Some other genes that are predicted by this gene can also be affected according to the change of its probability. For example, we may cut the connection from Gene 2 to Gene 6. Figure 6 shows the steady-state probability distribution.

Table 5: Influence of Gene Connection: $2 \rightarrow 6$ Is Broken

	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7
Original	0.5331	0.4397	0.3671	0.3671	0.6329	0.4638	0.5362
$2 \rightarrow 6$ is broken	0.5406	0.4347	0.4009	0.4009	0.5991	0.5626	0.4374

	Gene 8	Gene 9	Gene 10	Gene 11	Gene 12	Gene 13	Gene 14
Original	0.6888	0.4400	0.7561	0.4718	0.5282	0.4563	0.4581
$2 \rightarrow 6$ is broken	0.6804	0.4774	0.7317	0.4778	0.5222	0.4576	0.4476

Here, the Boolean function f_1^6 is set to be

$$x_6 = \begin{cases} f_1^6 = 1, & \text{if} \quad x_1 = 0; \\ f_1^6 = 0, & \text{if} \quad x_1 = 1. \end{cases}$$

Compared to previous f_1^6, Gene 6 will have a larger probability to be off. Since Gene 7 is only predicted by Gene 6, and when $x_6 = 0$, it is always on, thus the probability that Gene 7 is on becomes larger.

Table 5 shows the total probability that each gene is off for both the original system and the present system. It is consistent with the analysis above. The probabilities of Gene 6 and Gene 7 change most. Since other genes can be predicted by Gene 6 and Gene 7, their probabilities also change some. We choose the first 200 states with largest probability, compare them with those states when there is the connection, and we also find this phenomenon. But this does not mean that given any two states with all the genes having the same state except Gene 6 or Gene 7, the state with $x_6 = 0$ or $x_7 = 1$ will have larger probability. As mentioned before, one state of the system is a combination of all genes, it is possible that $x_6 = 0$ or $x_7 = 1$ can make the probability of all other genes less than that when $x_6 = 1$ or $x_7 = 0$. In our setting, most states with large probability in the original system remain large. Among the first 500 states with largest probability, more than 400 are among those in the original steady-state probability distribution.

4.2.2 Influence of Boolean Functions

Regarding the Boolean functions, if one can make a certain Boolean function inactive, and the selecting probabilities of other Boolean functions for the same target gene are re-scaled such that they are proportional to their previous probabilities, then the effect is similar to breaking some connections between genes. The number of recurrent states can be changed and the probabilities of the recurrent states can change too. To a certain extent, making a Boolean function inactive can be seen as breaking the connections between the target gene and all the input genes in a certain Boolean function.

Table 6: Influence of Boolean Functions: Number of Recurrent States

Inactive functions	f_2^1	f_1^2	f_1^3	f_3^6	f_1^{11}	f_1^{13}	f_2^{14}
Recurrent States	4424	3803	2229	4334	4272	4445	3815

Table 7: Influence of Boolean Function: f_1^2 Is Inactive

	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7
Original	0.5331	0.4397	0.3671	0.3671	0.6329	0.4638	0.5362
f_1^2 inactive	0.5268	0.3525	0.3515	0.3515	0.6485	0.4800	0.5200
	Gene 8	Gene 9	Gene 10	Gene 11	Gene 12	Gene 13	Gene 14
Original	0.6888	0.4400	0.7561	0.4718	0.5282	0.4563	0.4581
f_1^2 inactive	0.6876	0.4490	0.7534	0.4723	0.5277	0.4798	0.4628

The number of recurrent states is presented in Table 6. Inactive functions are the Boolean functions that we set to be inactive. Setting one Boolean function to be inactive can make the predicted gene have less connections. In this PBN setting, most of the number of recurrent states become smaller. If one Boolean function of a gene is inactive, how the probability of this gene (on or off) will be changed can be determined from the setting of the Boolean functions. From the numerical results, we find that the decision made directly from the Boolean function is consistent with the numerical results. Thus, we can determine the influence of a Boolean function from the setting of Boolean functions later. For example, if the Boolean function f_1^2 is set inactive, then Gene 2 will have higher probability to be on. The global state of the system with $x_2 = 1$ will have higher probability to be evolved into compared to that when there is the Boolean function. Table 7 shows the probability of all the genes in state 0. Gene 2 has higher probability to go into state 1, which is consistent with the analysis from the Boolean function. We find that in the first 100 states with largest probabilities, there are 77 states with $x_2 = 1$, while in the original steady-state probability distribution, there are only 58. As in the previous subsection, this does not mean that for any two states, if all the genes have same state except Gene 2, then the state with $x_2 = 1$ will have higher probability. Setting f_1^2 be inactive has little influence on other genes. The change of probability for other genes is very small. Figure 7 shows the difference of the steady-state probability distribution between the present one and the original one. The value of the vertical axis describes how much the present one changes from the original one. From the figure, we also can see that the states with $x_2 = 1$ have more probability.

From many tests, we find that the influence of setting one Boolean function inactive on the probability of all genes being in one state are not as large as the influence of gene connections.

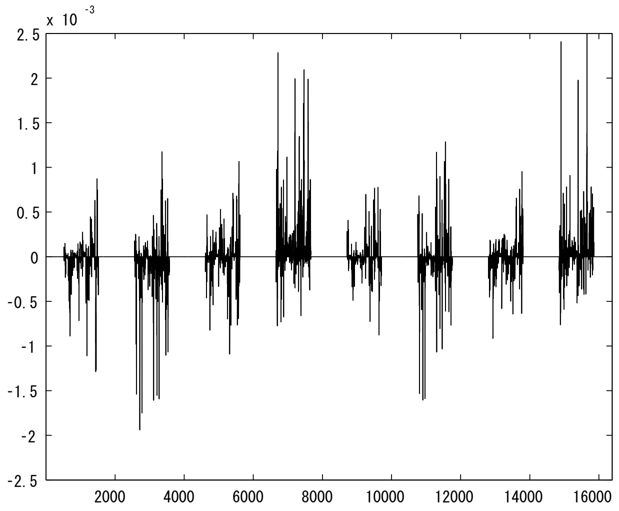

Figure 7: Steady-state Probability Distribution When f_1^2 Is Inactive

Table 8: Influence of Boolean Function: f_2^8 Is Inactive

	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7
Original	0.5331	0.4397	0.3671	0.3671	0.6329	0.4638	0.5362
f_2^8 is inactive	0.5536	0.4422	0.3737	0.3737	0.6263	0.4782	0.5218
	Gene 8	Gene 9	Gene 10	Gene 11	Gene 12	Gene 13	Gene 14
Original	0.6888	0.4400	0.7561	0.4718	0.5282	0.4563	0.4581
f_2^8 is inactive	0.7346	0.4453	0.7576	0.4794	0.5206	0.4516	0.4278

Another example is that when Boolean function f_2^8 is set inactive. From the setting of this network, Gene 8 will have larger probability to be off(Table 8). This is consistent with the numerical results. However, the probability of all other genes changes very little.

4.2.3 Influence of Genes

Resetting the state of a certain gene is an easily implementable method in gene control. If the state of a certain gene is set to be fixed in the whole process of evolution, then an interesting question will be: how will the steady-state probability distribution change? From the gene control point of view, this problem also can be addressed as if one control is applied, how will it affect the steady-state probability distribution of the system?

Table 9 shows the number of recurrent states when we set one of all the genes in a certain state. In most cases, the number is much less. But this also depends on the outdegree of each gene. For example, Gene 5 has no outdegree, when it is set in one state, it has no influence on other genes, thus the number of recurrent states does not change. If a gene has outdegree, the probability of its regulated gene being on or off will change. From the numerical results, we find that the tendency of this kind of changes can be determined from the setting of Boolean functions.

An example is that we set Gene 1 be off where Gene 1 is the input gene of Genes $2, 6, 13$ and 14. Figure 8 is the steady-state probability distribution when $x_1 = 0$. The system can only go into the first part of all the states. Table 10 shows the probability that each gene is off in the original steady-state distribution and in the present one. Among all the genes, Genes $6, 7$ and 14 are influenced greatly. From the Boolean functions, we can see that Gene 14 will have larger probability to get into 1 when $x_1 = 0$. Now it has a probability of 0.6886 to be on while in the original probability distribution, the probability is 0.5419. In the first 100 states with largest probabilities, there are 88 states with $x_{14} = 1$. From the Boolean function, we also can see when $x_1 = 0$, Gene 6 will go into 0 with greater probability. Now the total probability that Gene 6 is off is 0.5646, while it is only 0.4638 in the original steady-state probability distribution. Since Gene 6 is the predictor of Gene 7 and Gene 7 is off when Gene 6 is on, the probability of Gene 7 being off decreases from 0.5362 to 0.4354. In the first 100 states with largest probability, there are 15 states more with $x_6 = 0$, and there are 25 states more for $x_7 = 1$.

The probability of other genes also changes. Compared to the influence of BN, setting the state of one gene has larger influences on other genes.

4.3 Computational Time

The computational time goes up with the increase of number of genes and Boolean networks. The complexity of the algorithm is about $O(N2^n)$. All the experiments are done in a PC with CPU Pentium 4 and RAM 1G using MATLAB. Table 11 shows the time required for different number of Boolean networks when the number of genes is 14. The increase of computational

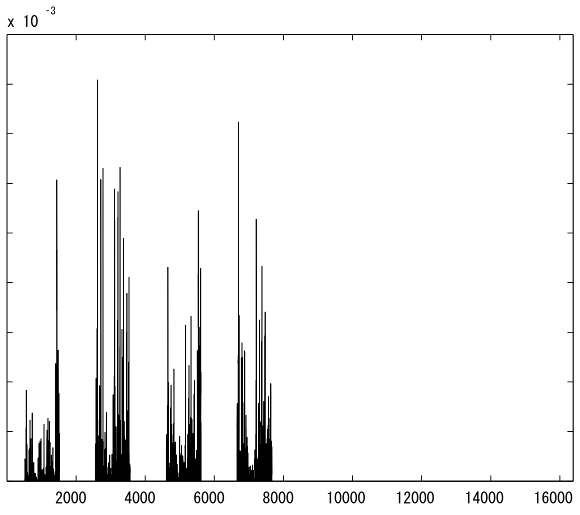

Figure 8: Steady-state Probability Distribution When $x_1 = 0$

Table 9: Influence of Genes: Recurrent States

State of Genes	$x_1 = 0$	$x_2 = 1$	$x_3 = 1$	$x_4 = 0$	$x_5 = 0$	$x_6 = 1$	$x_7 = 1$
Recurrent States	2036	2068	768	4450	4450	928	2288
State of Genes	$x_8 = 0$	$x_9 = 0$	$x_{10} = 1$	$x_{11} = 1$	$x_{12} = 1$	$x_{13} = 0$	$x_{14} = 0$
Recurrent States	1664	1920	1752	824	2302	2145	2269

Table 10: Influence of Genes: $x_1 = 0$

	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7	Gene 8
Original	0.4397	0.3671	0.3671	0.6329	0.4638	0.5362	0.6888
$x_1 = 0$	0.4542	0.4031	0.4031	0.5969	0.5646	0.4354	0.6731
	Gene 9	Gene 10	Gene 11	Gene 12	Gene 13	Gene 14	
Original	0.4400	0.7561	0.4718	0.5282	0.4563	0.4581	
$x_1 = 0$	0.4799	0.7355	0.4762	0.5238	0.4990	0.3114	

Table 11: Computational Time When $n = 14$

N	192	384	1024	1536
Computational Time in Minutes	10	20	54	82

Table 12: Computational Time When $N = 1024$

n	12	13	14	15
Computational Time in Minutes	5	10	21	43

time with respect to the total number of Boolean networks is linear. Table 12 shows the time required when the number of Boolean networks is fixed to 1024, and the number of genes varies. The increase of computational time is about twice when the number of genes increases by 1. Since the number of predictors for one gene cannot be very large (93% of genes are between 1 and 4 [10]), the expected number of Boolean functions for each gene also cannot be large. For a regulatory network of 20 genes, 10 of them have 2 Boolean functions, and the variables in each Boolean function are no more than 3, the computational time required to get the transition probability matrix is about 58 hours.

In all the tests, the power method can be used successfully. It can compute the steady-state probability distribution in a minute within 100 steps. Thus the computational cost for computing the steady-state probability distribution depends mainly on the generation of the transition probability matrix.

5 Concluding Remarks

In this paper, an efficient method to construct the transition matrix of probabilistic Boolean network was presented. The complexity of the algorithm to compute the transition probability matrix decreases from $O(N2^{2n})$ to $O(N2^n)$. And then the power method was used to get the steady-state probability distribution. With this method, we studied the influence of Boolean functions, gene connections and genes to the steady-state distribution. Since the goal to study gene regulatory network is to find some intervention or control strategies so that the system can evolve into the desirable states, this study is a necessary step before determining how to make controls.

Since a BN with small probability in the gene evolution process is rarely been chosen, if these networks can be removed during the analysis process, much time can be saved. Furthermore the problem whether these BNs can be removed directly or not and how to determine the proportion of BNs that can be removed will be studied in our future research.

Appendix: Settings of Numerical Experiments

Number of genes studied in the network: 14 genes
The names of all genes: Table 3
Number of Boolean functions for each gene: From Gene 1 to Gene 14: 2 2 2 1 1 3 1 2 1 1 2 1 2 2
Number of input genes in each Boolean function: Table 13
Predictors in each Boolean function: Table 14. Entries in each column are the variables in the Boolean functions.
Truth table of all Boolean functions: Table 15. Let f_j^i be the jth function of Gene i. Since the length of the truth table may vary between different functions, only the first $2^{nv(f_j^i)}$ bits are relevant to the f_j^i column of F, where $nv(f_j^i)$ is the number of variables in function f_j^i. Assume f_j^i is a function of three variables x_u, x_v, and x_w (variables are defined in Table 14). Then, $f_j^i(1)$(the first row of f_j^i) defines the output for the input vector (000). Correspondingly, $f_j^i(2)$ defines the output for the input vector (001), where $x_u = x_v = 0$ and $x_w = 1$, i.e., where the third input variable is equal to one. As another example, $f_j^i(5)$ defines the output for the input vector (100), where $x_u = 1$, $x_v = x_w = 0$.

Table 13: Number of Input Genes in Each Boolean Function

f_1^1	f_2^1	f_1^2	f_2^2	f_1^3	f_2^3	f^4	f^5	f_1^6	f_2^6	f_3^6	f^7	f_1^8	f_2^8	f^9
2	2	3	2	2	3	1	1	2	1	3	1	2	3	3

f^{10}	f_1^{11}	f_2^{11}	f^{12}	f_1^{13}	f_2^{13}	f_1^{14}	f_2^{14}
3	2	1	1	3	2	2	3

Table 14: Input Genes in Each Boolean Function

f_1^1	f_2^1	f_1^2	f_2^2	f_1^3	f_2^3	f^4	f^5	f_1^6	f_2^6	f_3^6	f^7	f_1^8	f_2^8	f^9
2	7	1	10	2	4	3	3	1	3	9	6	6	9	6
13	14	11	13	6	10			2		10		7	10	10
		12			11					8			11	8

f^{10}	f_1^{11}	f_2^{11}	f^{12}	f_1^{13}	f_2^{13}	f_1^{14}	f_2^{14}
9	8	12	11	1	12	1	12
3	10			2	14	4	8
11				11			13

References

[1] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano, Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions, in Proc. 9th Ann. ACM-SIAM Symp. Discrete Algorithms(SODA'98), pp. 695-702, 1998.

[2] T. Akutsu, S. Miyano, and S. Kuhara, Identification of genetic networks from a small number of gene expression patterns under the Boolean network model, in Proc. Pacific Symp. Biocomputing 4, pp. 17-28, 1999.

[3] T. Akutsu, S. Miyano, and S. Kuhara, Inferring qualitative relations in genetic networks and metabolic pathways, Bioinformatics, vol. 16, pp. 727-734, 2000.

[4] E. Boros, T. Ibaraki, and K. Makino, Error-free and best-fit extensions of partially defined Boolean functions, Information and Computation, vol. 140, pp. 254-283, 1998.

[5] M. Brun, E. R. Dougherty, and I. Shmulevich, Steady-state probabilities for attractors in probabilistic Boolean networks, Signal Processing, vol. 85, pp. 1993-2013, 2005.

[6] J. E. Celis, M. Kruhøfferm I. Gromova, C. Frederiksen, M.Østergaard, T. F. Ørntoft, Gene expression profiling: Monitoring transcription and translation products using DNA microarrays and proteomics, FEBS Lett. vol.480, no. 1, pp. 2-16, 2000.

[7] A. Datta, A. Choudhary, M. Bitter, and E. R. Dougherty, External control in Markovian genetic regulatory networks, Machine Learning, vol. 52, pp. 169-191, 2003.

[8] A. Datta, A. Choudhary, M. Bitter, and E. R. Dougherty, External control in Markovian genetic regulatory networks: the imperfect information case, Bioinformatics, vol. 20, pp. 924-930, 2004.

Table 15: Truth Table of All Boolean Functions

f_1^1	f_2^1	f_1^2	f_2^2	f_1^3	f_2^3	f^4	f^5	f_1^6	f_2^6	f_3^6	f^7	f_1^8	f_2^8	f^9
0	1	1	0	0	0	0	1	1	0	0	1	1	1	0
1	0	1	1	1	1	1	0	0	1	0	0	0	1	0
1	1	0	1	1	0			1		1		0	0	1
0	0	0	1	1	0			1		1		0	0	1
		1			0					0			0	1
		0			1					1			0	0
		1			1					0			1	1
		0			0					1			0	1

f^{10}	f_1^{11}	f_2^{11}	f^{12}	f_1^{13}	f_2^{13}	f_1^{14}	f_2^{14}
1	1	0	1	0	1	1	0
1	0	1	0	1	0	1	0
0	0			1	0	1	1
0	1			0	1	0	0
0				1			1
1				1			1
0				1			0
0				0			0

[9] K. Glass and S. A. Kauffman, The logical analysis of continuous, nonlinear biochemical control networks, J. Theoret. Biol., vol. 39, pp. 103-129, 1973.

[10] N. Guelzim, S. Bottani, P. Bourgine, and F. Képès, Topological and causal structure of the yeast transcriptional regulatory network, Nature Genetics, vol. 31, pp. 60-63, 2002.

[11] T. Haveliwala, and S. Kamvar, The second eigenvalue of the Google matrix, Stanford University, Technical Report (2003).

[12] O. Hirose, N. Nariai, Y. Tamada, H. Bannai, S. Imoto, and S. Miyano, Estimating gene networks from expression data and binding location data via Boolean networks, vol. 3480, pp. 349-356, Lecture Note in Computer Scineces.

[13] S. Huang, Gene expression profiling, genetic networks, and cellular states: An integrating concept for tumorigenesis and drug discovery, J. Mol. Med. , vol. 77, pp. 469-480, 1999.

[14] T. R. Hughes, M. Mao, A. R. Jones, J.Burchard, M. J. Marton, K. W. Shannon, S. M. Lefkowits, M. Ziman, J. M. Schelter, M. R. Meyer, S. Kobayashi, C. Davis, H. Dai, Y. D.

He, S. B. Stephaniants, G. Cavet, P.Blanchard, S. H. Friend, and P. S. Linsley, Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer, Nature Biotechnol., vol. 19, pp. 342-347, 2001.

[15] T. E. Ideker, V. Thorsson, and R. M. Karp, Discovery of regulatory interactions through perturbation: Inference and experimental design, in Proc. Pacific Symp. Biocomputing 5, pp. 302-313, 2000.

[16] H. D. Jong, Modeling and simulation of genetic regulatory systems: A literature review, J. Comp. Biol., vol. 9, pp. 67-103, 2002.

[17] S. A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, J. Theoret. Biol., vol. 22, pp. 437-467, 1969.

[18] S. A. Kauffman, Homeostasis and differentiation in random genetic control networks, Nature, vol. 224, pp. 177-178, 1969.

[19] S. A. Kauffman, The large scale structure and dynamics of genetic control circuits: an ensemble approach, J. Theoret. Biol., vol. 44, pp. 167-190, 1974.

[20] S. A. Kauffman, The origins of order: Self-organization and selection in evolution. New York: Oxford Univ. Press, 1993.

[21] S. Kim, E. R. Dougherty, M.L. Bitter, Y. Chen, K. Sivakumar, P. Meltzer, and J. M. Trent, General nonlinear framework for the analysis of gene interactions via multivariate expression arrays, J. Biomedical Optics, vol. 5, pp. 411-444, 2000.

[22] R. J. Lipshutz, S.P. A. Fodor, T. R. Gingeras, and D. J. Lockhart, High density synthetic oligonucleotide arrays, Nature Genetics, vol. 21, pp. 20-24, 1999.

[23] D. J. Lockhart, and E. A. Winzeler, Genomics, gene expression and DNA arrays, Nature, vol. 405, pp. 827-836, 2000.

[24] M. K. Ng, S. Zhang, W. Ching, and T. Akutsu, A control model for Markovian genetic regulatory networks, to appear in LNCS transactions on Computational Systems Biology.

[25] K. Noda, A. Shinohara, M. Takeda, S. Matsumoto, S. Miyano, and S. Kuhara, Finding genetic network from experiments by weighted network model, Genome Informatics, vol. 9, pp. 141-150, 1998.

[26] J. S. Rosenthal, Minorization conditions and convergence rates for Markov chain Monte Carlo. Journal of the American Statistical Association, vol. 90, pp. 558-566, 1995.

[27] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown, Quantitative monitoring of gene expression patterns with a complementary DNA microarray, Science, vol. 270, pp. 467-470, 1995.

[28] I. Shmulevich and W. Zhang, Binary analysis and optimization-based normalization of gene expression data, Bioinformatics, vol. 18, no. 4, pp. 555-565, 2002.

[29] I. Shmulevich, A. Saarinen, O. Yli-Harja, and J. Astola, Inference of genetic regulatory networks under the best-fit extension paradigm, in Computational and Statistical Approaches To Genomics, W. Zhang and I. Shmulevich, Eds. Boston, MA: Kluwer, 2002.

[30] I. Shmulevich, E. R. Dougherty, and W. Zhang, From Boolean to probabilistic Boolean networks as models of genetic regulatory networks. Proceedings of IEEE 90, pp. 1778-1792, 2002.

[31] I. Shmulevich, E. R. Dougherty, and W. Zhang, Gene perturbation and intervention in probabilistic Boolean networks, Bioinformatics, vol. 18, pp. 1319-1331, 2002.

[32] I. Shmulevich, E. R. Dougherty, and W. Zhang, Control of stationary behavior in probabilistic Boolean networks by means of structural intervention, J. Biol. Systems, vol. 10, pp. 431-445, 2002.

[33] I. Shmulevich, E. R. Dougherty, S. Kim and W. Zhang, Probabilistic Boolean Networks: A rule-based uncertainty model for gene regulatory networks, Bioinformatics, vol. 18, pp. 261-274, 2002.

[34] I. Shmulevich, I. Gluhovsky, R. F. Hashimoto, E. R. Dougherty, and W. Zhang, Steady-state analysis of genetic regulatory networks modeled by probabilistic Boolean networks, Comparative and Functional Genomics, vol. 4, pp. 601-608, 2003.

[35] R. Somogyi and C. Sniegoski, Modeling the complexity of gene networks: Understanding multigenic and pleiotropic regulation, Complexity, vol. 1, pp. 45-63, 1996.

[36] Z. Szallasi and S. Liang, Modeling the normal and neoplastic cell cycle with 'realistic Boolean genetic networks': Their application for understanding carcinogenesis and assessing therapeutic strategies, in Proc. Pacific Symp. Biocomputing 3, pp. 66-76, 1998.

[37] A. Wuensche, Genomic regulation modeled as a network with basins of attraction, in Proc. Pacific Symp. Biocomputing 3, pp. 89-102, 1998.