Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

Exploiting Binary Abstractions in Deciphering Gene Interactions

Sungroh Yoon, Abhishek Garg, Eui-Young Chung, Hyun Seok Park, Woong Yang Park, and Giovanni De Micheli, *Fellow, IEEE*

Abstract— **We consider computationally reconstructing gene regulatory networks on top of the binary abstraction of gene expression state information. Unlike previous Boolean network approaches, the proposed method does not handle noisy gene expression values directly. Instead, two-valued "hidden state" information is derived from gene expression profiles using a robust statistical technique, and a gene interaction network is inferred from this hidden state information. In particular, we exploit Espresso, a well-known 2-level Boolean logic optimizer in order to determine the core network structure. The resulting gene interaction networks can be viewed as dynamic Bayesian networks, which have key advantages over more conventional Bayesian networks in terms of biological phenomena that can be represented. The authors tested the proposed method with a time-course gene expression data set from microarray experiments on anti-cancer drugs doxorubicin and paclitaxel. A gene interaction network was produced by our method, and the identified genes were validated with a public annotation database. The experimental studies we conducted suggest that the proposed method inspired by engineering systems can be a very effective tool to decipher complex gene interactions in living systems.**

I. Introduction

Now equipped with well-established methods to sequence genes in many organisms including humans, we want to understand the interactions of individual genes as a next step. In particular, computational reconstruction of gene interaction networks has received much attention since the invention of the DNA microarray technology. Despite some controversial issues involved [1], it remains true that the DNA microarray technology delivers unprecedented throughput when we want to monitor the expression of a whole genome simultaneously. Large-scale gene expression data sets obtained from DNA microarray experiments provide invaluable information for gene network inference algorithms that typically require a large amount of empirical data.

The problem of reverse-engineering a complex system from its input and output behavior, as is the case in gene network reconstruction, has already been extensively studied in electrical engineering. In particular, approaches to model gene networks using Boolean networks [2], [3] bear great similarities to digital circuit synthesis. As will be seen shortly, the network connectivity information in Boolean networks can easily be obtained by Espresso, a well-known 2-level Boolean logic minimizer [4], even though this fact remained unnoticed in previous work on Boolean networks.

Although Boolean network approaches can be computationally more efficient than alternatives and are therefore scalable to a larger gene network, the impact of Boolean methods on the bioinformatics field has been somewhat limited. A possible reason is that representing gene expression levels by only two states can be oversimplification of continuous biological signals. Another reason may come from the deterministic nature of most Boolean approaches. Biological data can often be noisy, and statistical methods may provide a more robust solution. To alleviate the second problem, probabilistic Boolean networks have been proposed [5], [6], but they still suffer from the first issue – oversimplification of gene expression profiles.

Given the mature technologies and tools for processing binary information in engineering, we claim that a binary abstraction of biological information remains to be a very appealing technique for inferring gene interaction networks, assuming that we use binary abstractions in a right context. Most previous Boolean approaches start with a binary representation of gene expression profiles: if a gene expression level is lower than a threshold, the gene expression is considered 0 or OFF; otherwise the expression is regarded as 1 or ON. As previously stated, this bifurcation of gene expression values tends to be overly simplistic and prone to error. Just as Boolean logic minimizers are not designed for noisy electrical signals obtained directly from analog sensors, gene expression profiles (they are noisy biological signals from biosensors) are often not suitable for binary representations and processing.

In this paper, we introduce a new computational method to build gene interaction networks from time-course gene expression data. This method distinguishes two types of states associated with gene expression. The *observed* state of a gene is its empirically observed expression value. The *hidden* state of a gene represents its biological state that caused the observed state. The hidden state information is deduced from the observed state information by a statistical approach to handle noise in expression values. It is the hidden state information that is represented and processed in its binary form. The proposed method analyzes the hidden state information and finally produces gene interaction networks.

The introduction of hidden states resembles the state definitions in conventional hidden Markov models (HMMs) [7].

This work was supported by a grant of Jerry Yang and Akiko Yamazaki.

S. Yoon is with the Computer Systems Laboratory, Stanford University, Stanford, CA 94305, USA (E-mail: sryoon@stanford.edu)

A. Garg and G. De Micheli are with the Integrated Systems Center, Swiss Federal Institute of Technology (EPFL), Lausanne, CH-1015, Switzerland

E.-Y. Chung is with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Korea

H. S. Park is with the Department of Computer Science and Engineering, Ewha Womans University, Seoul 120-750, Korea and Macrogen Corporation, Seoul 153-023, Korea

W. Y. Park is with the Department of Biochemistry, Seoul National University Medical School, Seoul 110-744, Korea

There have been approaches that use HMMs to analyze time-course microarray data sets [8], [9], [10], but these methods report clusters of related genes instead of gene interaction networks, mostly due to the limited representational power of HMMs. Our method constructs a network that is more similar to dynamic Bayesian networks (DBNs) [11], [12], which includes HMMs and Kalman filters [13] as special cases. The core of building DBNs from time-course data is to learn the structure of a two-slice temporal network (2TN), which represents the causal relationships of variables between two adjacent time slices assuming the first-order Markov models. We use Espresso, the well-known 2-level Boolean logic minimizer, in order to learn the 2TN structure. If we assume higher-order Markov models, we can also employ multi-level logic minimization techniques.

The organization of this paper is as follows. Section II describes the details of extracting the hidden state information from the observed state information and learning the 2TN structure using a Boolean logic synthesis technique. In Section III, we test the proposed method with a biological data set and present the result. Section IV concludes this paper with future work.

II. Method

The input to the proposed method is a time-course gene expression data set obtained by microarray experiments, and the output is a directed (possibly cyclic) graph representing interactions of the genes involved in the microarray experiment. The method consists of the following major steps:

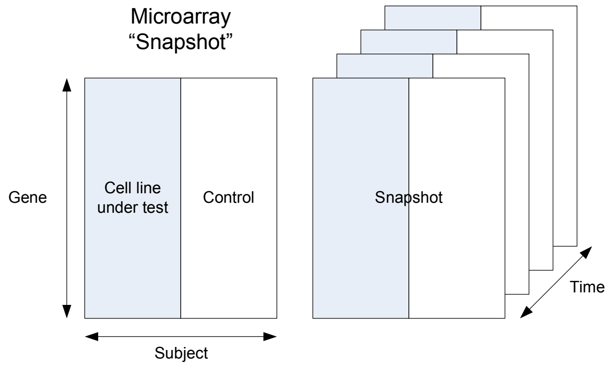
1) Extracting the hidden state information and construct a hidden state table (Section II-B);
2) Simplifying the hidden state table by removing non-informative genes (Section II-C);
3) Determining and optimizing gene dependency information (Section II-E); and
4) Representing the dependency information by a 2-slice temporal network and converting it into a gene interaction network (Section II-F).

The definitions and assumptions made are presented in Sections II-A and II-D, respectively.

A. Definitions

Suppose that we are monitoring the expression of N genes $G = \{g_1, g_2, \dots, g_N\}$ in M samples $S = \{s_1, s_2, \dots, s_M\}$ taken from two distinct cell lines. Assume that samples $S_1 = \{s_1, s_2, \dots, s_m\}$ are taken from one cell line and $S_2 = \{s_{m+1}, s_{m+2}, \dots, s_M\}$ from the other.

Assuming that we measure gene expression at K different time points $T = \{t_1, t_2, \dots, t_K\}$, a *snapshot* is a real-valued matrix denoted by $D_k \in \mathbb{R}^{N\times M}$, where element $(D_k)_{ij}$ represents the expression value of gene $g_i \in G$ in sample $s_j \in S$ measured at time $t_k \in T$. That is, a snapshot is a matrix of gene expression values measured at a certain fixed time for all genes and samples. A collection of snapshots, $\{D_1, D_2, \dots, D_P\}$, is called a *time-course expression data set*. Fig. 1 informally shows a snapshot and a time-course expression data set.



(a) Snapshot (b) Time-course expression data

Fig. 1. Definitions: snapshot and time-course expression data.

The *observed state* of gene g_i in sample s_j at time t_k is denoted by $o_{ijk} = (D_k)_{ij} \in \mathbb{R}$. The *hidden state* of gene g_i at time t_k is denoted by $h_{ik} \in \{DE, EE\}$, where DE and EE represent "Differentially Expressed" and "Equally Expressed" over different cell lines, respectively.

The *hidden state table* is a matrix of hidden states and is denoted by H, where the element $(H)_{ik}$ in row i and column k equals h_{ik}, the hidden state of gene $g_i \in G$ at time $t_k \in T$. As will be explained shortly, every hidden state table is associated with some statistic to represent statistical confidence of the table entries.

B. Extracting hidden state table

The first step of the proposed method is to extract hidden state information from a time-series microarray data set and build a hidden state table. We construct one column of the table per snapshot. For each $g_i \in G$ and $t_k \in T$, we do a hypothesis testing to determine if gene g_i is differentially expressed between cell line samples S_1 and S_2. More formally, we use the general statistical model of a gene expression value [14] to represent the observed state o_{ijk}, namely

$$o_{ijk} = a_{ik} + b_{ik} I\{s_j \in S_1\} + \epsilon_{ijk} \qquad (1)$$

where $I\{\cdot\}$ is the indicator random variable[1], ϵ_{ijk} is random errors with zero mean. That is, the mean expression values of gene g_i in two distinct cell lines at time t_k are $a_{ik} + b_{ik}$ and a_{ik}, respectively. Then, for each gene, we test the null hypothesis $\mathcal{H}_0 : b_{ik} = 0$ against the alternative hypothesis $\mathcal{H}_1 : b_{ik} \neq 0$ with significance level α. Finally, the hidden state h_{ik} is a binary variable defined as

$$h_{ik} = \begin{cases} DE \triangleq 1, & \text{if } \mathcal{H}_0 \text{ is rejected;} \\ EE \triangleq 0, & \text{otherwise.} \end{cases} \qquad (2)$$

Example 1: Assume that the elements of two sets $\{3.0, 3.1, 3.1, 3.2, 3.0\}$ and $\{0.0, 0.1, 0.2, 0.1, 0.1\}$ are the expression values of gene $g_i \in G$ measured at time $t_k \in T$ from 5 cancer patients and 5 normal ones, respectively. We show how to compute h_{ik}, the hidden state of gene g_i at time

[1]$I\{true\} = 1; I\{false\} = 0$.

t_k. Assume that the two populations from which the samples were drawn follow the normal distribution. We can use the t-test [15] to test if the two group means are statistically different. Let $\overline{Y_a}$, v_a and M_a denote the sample mean, the sample variance, and the sample size of the first group, and $\overline{Y_b}$, v_b and M_b for the second. Then, the t-statistic is

$$t = \frac{\overline{Y_a} - \overline{Y_b}}{\sqrt{\frac{v_a}{M_a} + \frac{v_b}{M_b}}} = \frac{3.08 - 0.1}{\sqrt{\frac{0.007}{5} + \frac{0.005}{5}}} = 60.829 \qquad (3)$$

and the degrees of freedom ν is

$$\nu = \left\lfloor \frac{(\frac{v_a}{M_a} + \frac{v_b}{M_b})^2}{(\frac{v_a}{M_a})^2/(M_a - 1) + (\frac{v_b}{M_b})^2/(M_b - 1)} \right\rfloor = 8. \qquad (4)$$

Assuming that we use significance level $\alpha = 0.05$,

$$t > t_{(\alpha/2,\nu)} = 2.3060 \qquad (5)$$

where $t_{(\alpha/2,\nu)}$ is the critical value of the t-distribution with significance level $\alpha = 0.05$ and $\nu = 8$ from the (two-tailed) t-distribution table [15]. Since $t > t_{(\alpha/2,\nu)}$, we reject the null hypothesis $\mathcal{H}_0$ that the two group means are the same. Therefore, $h_{ik} = DE$. □

When calculating h_{ik} for many genes, we need a correction of significance level α for multiple comparisons [16], thus ensuring that too many false positives are not declared. For instance, if the genome-wide significance level $\alpha = 0.05$ for $N = 1000$ genes, the (two-sided) gene-specific significance level $\alpha^* = \alpha/(2N) = 0.000025$ by the Bonferroni correction method [15].

The use of the t-statistic and the Bonferroni adjustment above was just to make explanation simple and clear. More sophisticated methods than have been proposed especially for the analysis of microarray data [14], [16]. For the experiments presented in Section III, we used the *significance analysis of microarray* (SAM) statistic [17] and the *false discovery rate* (FDR) [18] as the test statistic and the multiple comparison correction method, respectively, instead of the t-statistic and Bonferroni-corrected α.

C. Simplifying hidden state table

After the hidden state table is completed, we remove from it those rows that do not have any DE entry. We are not interested in the genes whose expression levels do not change between cell lines. In other words, we consider only those genes in set $G' = G - \{g_i | \forall t_k \in T, h_{ik} = EE\}$. However, we continue to use G in place of G' for notational convenience.

D. Assumption: stationary Markov process

Having determined the gene variables and their states, the next step is to specify the dependencies among the variables. To make computation feasible, two reasonable assumptions are typically made [11]. First, we assume that state changes result from a stationary process, where changes in the states are governed by laws that do not themselves over time.

Second, we make a Markov assumption that the current states depend only a finite history of previous states. In particular, we assume the first-order Makrov process, where the current states depend only on the previous states and not on any earlier states. Thus, h_{ik}, the hidden state of gene g_i at time t_k, depends only on the hidden states at time t_{k-1}.

	t_1	t_2	t_3	t_4
g_1	EE	EE	DE	EE
g_2	DE	DE	EE	EE
g_3	EE	DE	EE	DE

(a) Hidden state table

x_1	x_2	x_3	x_1^+	x_2^+	x_3^+
EE	DE	EE	EE	DE	DE
EE	DE	DE	DE	EE	EE
DE	EE	EE	EE	EE	DE

(b) State transitions

Fig. 2. An example of the hidden state table and state transitions.

E. Learning and optimizing dependency information

Based upon the Markov assumption made above, we can represent each hidden state h_{ik} as a function of the hidden states in the previous time slice, namely,

$$h_{ik} = f_{ik}\left(h_{1(k-1)}, h_{2(k-1)}, h_{3(k-1)}, \ldots, h_{i(k-1)}, \ldots\right). \qquad (6)$$

Since we also assume a stationary process, we can drop k from (6), which becomes

$$x_i^+ = f_i\left(x_1, x_2, x_3, \ldots, x_i, \ldots\right) \qquad (7)$$

where Boolean variables x_i^+ and x_i represent h_{ik} for arbitrary k and $k-1$, respectively. If the hidden state table has K columns, then we have $(K-1)$ input/output pairs for each f_i.

Example 2: Fig. 2 shows an example of the hidden state table and the state transition information implied. □

The task is then to determine each f_i from this input/output information appearing in the hidden state table. The difference between our method and others becomes salient in this process. Other methods use gene expression values (directly or after rounding them to "on" or "off") as the input/output variables of the function. Then, the function itself is estimated by techniques such as (non)parametric regression, covariance matrix estimation, maximum likelihood estimation, and information-theoretic approaches. In contrast, the proposed method uses the binary hidden state information as the input/output variables and then synthesize each f_i as a 2-level Boolean logic function.

To synthesize f_i, we first represent each column of the hidden state table as a minterm. For a Boolean function, a minterm is defined as a product term in which each input variable appears exactly once either in its complemented or uncomplemented form. Thus, if we denote by $\mathcal{M}_k$ the minterm for column k in the hidden state table, then

$$\mathcal{M}_k = \prod_{i=1}^{N} \left(x_i I\{h_{ik} = DE\} + \overline{x_i} I\{h_{ik} = EE\}\right). \qquad (8)$$

As a Boolean function, each f_i can be represented by a sum of minterms. Thus, we can represent x_i^+ as a sum of $(K'-1)$

```
.i 3
.o 3
010 011
011 100
100 001
.e
```

(a) Input

```
.i 3
.o 3
.p 3
--1 100
-10 010
--0 001
.e
```

(b) Output

$$x_1^+ = x_3$$
$$x_2^+ = x_2\overline{x_3}$$
$$x_3^+ = \overline{x_3}$$

(c) Minimized x^+ variables

Fig. 3. Minimizing dependency information by Espresso.

minterms:

$$x_i^+ = \sum_{i=1}^{K-1} \left(\mathcal{M}_k I\{h_{i(k+1)} = DE\} + \overline{\mathcal{M}_k} I\{h_{i(k+1)} = EE\}\right). \tag{9}$$

Example 3: In the hidden state table in Fig. 2(a),

$$\begin{aligned}\mathcal{M}_1 &= \overline{x_1}x_2\overline{x_3}\\ \mathcal{M}_2 &= \overline{x_1}x_2x_3\\ \mathcal{M}_3 &= x_1\overline{x_2}\overline{x_3}\end{aligned}$$

by (8). Then, by (9), we can obtain

$$\begin{aligned}x_1^+ &= \overline{\mathcal{M}_1} + \mathcal{M}_2 + \overline{\mathcal{M}_3}\\ &= \overline{\overline{x_1}x_2\overline{x_3}} + \overline{x_1}x_2x_3 + \overline{x_1\overline{x_2}\overline{x_3}},\end{aligned}$$

and variables x_2^+ and x_3^+ can be derived similarly. □

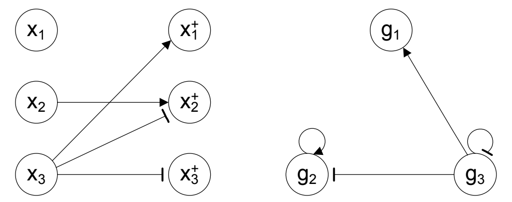
The function synthesized by (9) can further be simplified by 2-level Boolean logic minimization techniques. In this work, we use Espresso, a widely used 2-level logic optimizer. Espresso takes as input a two-level representation of a Boolean function and produces a minimal equivalent representation. The details on Espresso are beyond the scope of this paper but can be found in [4].

Example 4: The logic equation derived in Example 3 can be minimized by Espresso. Fig. 3(b) shows the Espresso output where `.i`, `.o`, `.p`, and `.e` specify the number of input variables, the number of output functions, the number of product terms, and the end of output, respectively. The lines before `.e` but after the `.p` line correspond to the encoded product terms, which can be decoded into the form shown in Fig. 3(c). □

The process of minimizing function f_i also serves the role of avoiding overly complex dependencies of a gene on others, because it is well known that typically a gene is regulated by or regulates only a small number of other genes. This is also common in some other methods. For instance, some regression approaches employed penalized regression that penalizes overly complicated dependence structures among variables.

F. Deriving gene interaction networks

After the logic minimization step, the dependency information among the genes are represented by a directed bipartite graph called a *2-slice temporal network* (2TN). Let $X = \{x_1, x_2, \ldots, x_N\}$ and $X^+ = \{x_1^+, x_2^+, \ldots, x_N^+\}$. The 2TN is a directed bipartite graph (V, E) with vertex set $V = X \cup X^+$ and edge set $E = \{(x, x^+) | x \in X, x^+ \in X^+, x^+$ is a function of x or $\overline{x}\}$. By using different edges, we can distinguish the dependency of x^+ on either x or $\overline{x}$. Finally, we can derive a gene interaction network from the 2TN by merging x_i and x_i^+ vertices and labeling the merged vertex by g_i.



(a) 2TN (b) Gene network

Fig. 4. The 2-slice temporal network and the gene interaction network derived from Fig. 3 (the vertical bars on edges represent negation).

This network can represent both positive and negative regulations as well as feedback loops, which frequently occur in biological processes but cannot be represented by conventional Bayesian networks.

Example 5: Fig. 4(a) shows the 2-slice temporal network derived from the dependency information listed in Fig. 3(c). Positive and negative regulations are denoted by edge $\rightarrow$ and edge $\dashv$, respectively. For instance, gene g_3 negatively regulates gene g_2 and positively regulates gene g_1. Also note that genes g_2 and g_3 have positive and negative self-regulation, respectively. □

III. Experimental Results

A. The time-course expression data used

We measured the expression of 10,305 human genes at 10 time points (0, 0.5, 1, 1.5, 2, 3, 4, 5, 6, and 8 hours) from 20 patients resistant to anti-cancer drugs doxorubicin and paclitaxel. For comparison, we also measured the expression of the same genes at the same set of time points from 20 patients sensitive to the same anti-cancer drugs. We used microarrays manufactured by MacroGen Incorporated (Seoul, Korea). The scanned images were processed by GenePix software from Molecular Devices Corporation (Sunnyvale, California) and were saved in the GPR format. More details on the experiments will be published elsewhere.

The Bioconductor packages [19] for the R language were utilized to read the GPR files and preprocess them. In particular, we used the Edwards method [20] for correcting background intensities and then normalized the data sets by functions **`normalizeWithinArrays()`** and **`normalizeBetweenArrays()`** in the limma package. We considered only those genes that showed at least a 2-fold change in their expression level.

B. Derivation of a gene interaction network

To extract the hidden state information and construct a hidden state table as described in Section II-B, the SAM package [17] was employed with the FDR value of 10^{-5}. SAM uses a modified t-test statistic with sample-label permutations to evaluate statistical significance. The SAM statistic was chosen because it does not make strong parametric assumptions and does not require any complex estimation

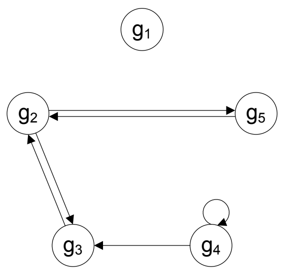
ID	Gene	Description
g_1	*TNFRSF10B*	tumor necrosis factor receptor superfamily
g_2	*GAD1*	glutamate decarboxylase 1
g_3	*PYGL*	phosphorylase, glycogen
g_4	*ASIP*	agouti signaling protein
g_5	*TXNRD1*	thioredoxin reductase 1

(a) Identified genes

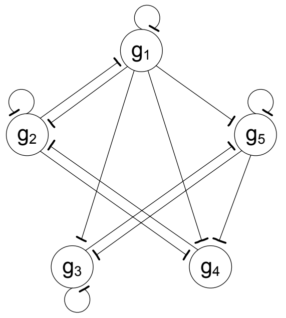
x_1	x_2	x_3	x_4	x_5	x_1^+	x_2^+	x_3^+	x_4^+	x_5^+
EE	EE	DE	DE	EE	EE	EE	EE	DE	EE
EE	EE	EE	DE	EE	EE	DE	EE	DE	EE
EE	DE	EE	DE	EE	EE	EE	EE	EE	EE
EE	EE	EE	EE	EE	DE	EE	EE	EE	EE
DE	EE	EE	EE	EE	EE	DE	EE	DE	EE
EE	DE	EE	DE	EE	EE	EE	EE	DE	EE
EE	EE	EE	DE	EE	EE	DE	EE	DE	EE
EE	DE	EE	DE	EE	EE	EE	DE	EE	DE
EE	EE	DE	EE	DE	EE	DE	EE	EE	EE

(b) State transitions

Fig. 5. The genes that were found differentially expressed and their transition information.



(a) Positive regulation



(b) Negative regulation

Fig. 6. Gene interaction networks derived from the transition information in Fig. 5(b).

procedures [14], [17]. The FDR is used by SAM as an alternative to controlling the false positive rate in handling multiple comparisons.

Fig. 5(a) lists the genes and their descriptions identified by this procedure, and Fig. 5(b) shows the transition information of these genes. This transition information was simplified by Espresso and was converted into a 2-slice temporal network, which was finally transformed to the gene networks shown in Fig. 6.

Excluding preprocessing, the entire procedure to infer the gene network took less than an hour on a reasonably-equipped workstation with the majority of the running time spent on the SAM procedure.

C. Expression profiles of identified genes

Fig. 7 shows the time-course expression profiles of the individual genes listed in Fig. 5(a). In each plot, a red circle (a blue square) represents the average expression value of the corresponding gene at a time point across all patients who are resistant (sensitive) to the drug. Centered on each circle or square is a vertical error bar showing the range of a standard deviation. The expression of genes *GAD1* and *TXNRD1* was higher in the samples resistant to the drug whereas the expression of genes *PYGL* and *ASIP* was lower in those samples. The difference was more subtle for

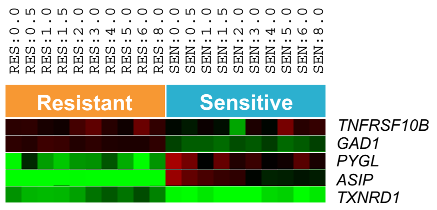


Fig. 8. The heat map representing the expression of the identified genes (red = high level of gene expression, green = low level of gene expression).

gene *TNFRSF10B*, which might not be detected by a simple discretization method such as thresholding.

The heat map shown in Fig. 8 also contrasts the expression of the five genes between resistant and sensitive samples. The rows and columns of the map correspond to genes and time points, respectively. Just as in a typical heat map for microarray data visualization, each red or green box represents gene expression levels that are high or low, respectively. The values shown here are the median values over the 20 samples in each cell line.

D. Validation with Gene Ontology

In order to further validate the findings, we utilized database Gene Ontology (GO) [21]. GO consists of three collections of controlled vocabularies that describe molecular functions, biological processes, or cellular components related to gene activities. By identifying common GO terms that annotate a certain set of genes, researchers can informatically validate the coherency of the genes in the set, assuming that the sharing is statistically significant. Generally, the hypergeometric distribution is used to calculate the p-value of a GO term shared by certain genes, and this p-value is usually corrected to reflect the multiple comparison issue involved in computation [22]. We found that term GO:0006091 (generation of precursor metabolites and energy) annotates the genes listed in Fig. 5(a) with the corrected p-value of 0.00136. These genes are related to an energy pathway, namely the formation from simpler components of precursor metabolites, substances from which energy is derived, and the processes involved in the liberation of energy from these substances.

IV. Conclusions

We described a method to reverse-engineer gene interaction networks from empirical time-course gene expression data. The proposed method utilizes a robust statistical technique that can capture the transition information of gene expression levels in a binary form. We then explained how to use Espresso, a Boolean logic optimizer to simplify this binary transition information and derive gene networks from it. According to the preliminary experimental study we performed, we expect that the proposed method can be a promising tool to computationally infer the interactions of genes in complex biological systems.

We can extend the technique proposed in this paper by relaxing the two assumptions on the underlying biological model – a stationary first-order Markov process. Instead of 2-slice temporal networks, we can use n-slice temporal

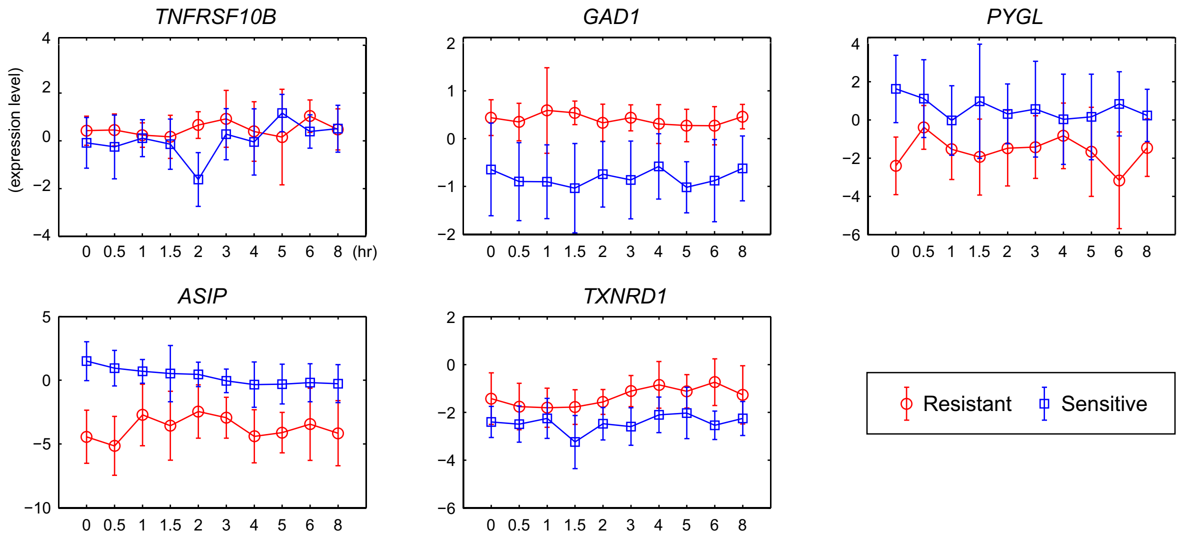


Fig. 7. The expression profiles for the genes listed in Fig. 5(a).

networks by assuming $(n-1)^{\text{th}}$ order Markov process and determine the network structure by multi-level logic synthesis techniques. Also, we can assume a non-stationary process and aim at finding gene dependencies that change over time. The price we should pay for these two extensions is increased computational complexity.

Another extension that is possible without a big computational overhead is to use multi-valued logic instead of binary logic. For instance, we can divide state DE into $DE+$ and $DE-$, which represent the expression level of a gene under one condition is statistically higher and lower than the other condition, respectively. In Espresso, multi-valued logic can easily be handled by the 1-hot encoding.

Finally, the criteria used by Boolean logic optimizers may be enhanced by incorporating additional constraints derived from biological perspectives.

Acknowledgments

The authors would like to thank Professor Russ B. Altman at Stanford University for helpful discussions.

References

[1] G. L. G. Miklos and R. Maleszka, "Microarray reality checks in the context of a complex disease," *Nature Biotechnology*, vol. 22, no. 5, pp. 615–621, May 2004.

[2] S. Liang, S. Fuhrman, and R. Somogyi, "REVEAL, a general reverse engineering algorithm for inference of genetic network architectures," in *Proceedings of the 3rd Pacific Symposium on Biocomputing*, 1998, pp. 3:18–29.

[3] 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 *Proceedings of the 4th Pacific Symposium on Biocomputing*, 1999, pp. 4:17–28.

[4] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Boston, MA: Kluwer, 1984.

[5] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, no. 2, 2002.

[6] I. Shmulevich, E. R. Dougherty, and W. Zhang, "From boolean to probabilistic boolean networks as models of genetic regulatory networks," *Proceedings of the IEEE*, no. 11, 2002.

[7] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, no. 2, 1989.

[8] A. Schliep, I. G. Costa, C. Steinhoff, and A. Schönhuth, "Analyzing gene expression time-courses," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 02, no. 3, Jul-Sept 2005.

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

[10] A. Schliep, A. Schönhuth, and C. Steinhoff, "Using hidden markov models to analyze gene expression time course data," *Bioinformatics*, vol. 19, no. suppl. 1, pp. i255–i263, 2003.

[11] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. New Jersey: Prentice Hall, December 2002.

[12] K. Murphy, "Dynamic Bayesian networks: Representation, inference and learning," Ph.D. dissertation, University of California, Berkeley, 2002.

[13] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[14] W. Pan, "A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments," *Bioinformatics*, vol. 18, no. 4, pp. 546–554, April 2002.

[15] B. Rosner, *Fundamentals of Biostatistics*, 5th ed. Pacific Grove, California: Duxbury, 2000.

[16] S. Dudoit, J. P. Shaffer, and J. C. Boldrick, "Multiple hypothesis testing in microarray experiments," *Statistical Science*, vol. 18, no. 1, pp. 71–103, 2003.

[17] V. G. Tusher, R. Tibshirani, and G. Chu, "Significance analysis of microarrays applied to the ionizing radiation response," *Proc. Natl. Acad. Sci USA*, vol. 98, no. 9, pp. 5116–5121, April 2001.

[18] B. Efron, J. D. Storey, R. Tibshirani, and V. Tusher, "Empirical Bayes analysis of a microarray experiment," *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1151–1160, December 2001.

[19] R. Gentleman, V. Carey, W. Huber, R. Irizarry, and S. Dudoit, Eds., *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. New York: Springer, 2005.

[20] D. Edwards, "Non-linear normalization and background correction in one-channel cDNA microarray studies," *Bioinformatics*, vol. 19, no. 7, pp. 825–833, May 2003.

[21] The Gene Ontology Consortium, "Gene ontology: tool for the unification of biology." *Nature Genetics*, vol. 25, no. 1, pp. 25–29, 2000.

[22] E. I. Boyle, S. Weng, J. Gollub, H. Jin, D. Botstein, J. M. Cherry, and G. Sherlock, "GO::TermFinder," *Bioinformatics*, vol. 20, no. 18, pp. 3710–3715, December 2004.