A System for Identifying Genetic Networks from Gene Expression Patterns Produced by Gene Disruptions and Overexpressions

Tatsuya Akutsu [1]
takutsu@ims.u-tokyo.ac.jp
Satoru Kuhara [2]
kuhara@grt.kyushu-u.ac.jp
Osamu Maruyama [1]
maruyama@ims.u-tokyo.ac.jp
Satoru Miyano [1]
miyano@ims.u-tokyo.ac.jp

[1] Human Genome Center, Institute of Medical Science, University of Tokyo
4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan
[2] Graduate School of Genetic Resources Technology, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan

Abstract

A hot research topic in genomics is to analyze the interactions between genes by systematic gene disruptions and gene overexpressions. Based on a boolean network model without time delay, we have been investigating efficient strategies for identifying a genetic network by multiple gene disruptions and overexpressions. This paper first shows the relationship between our boolean network model without time delay and the standard synchronous boolean network model. Then we present a simulator of boolean networks without time delay for multiple gene disruptions and gene overexpressions, which includes a genetic network identifier with a graphic interface that generates instructions for experiments of gene disruptions and overexpressions.

1 Introduction

S. cerevisiae is the first eukaryotic organism whose complete genome was determined [14]. Its whole DNA sequences have been open to public since 1996. Then the complete genomes of *E. coli* [5] and *B. subtilis* [11] were determined in 1997. The sequencing of a multicellular organism *C. elegans* will finish in 1998 with DNA sequences of total size 100M bp. Along with these advances of sequencing projects, ambitious researches are heating up for systematic analysis of gene functions and genetic networks by using large-scale gene expression patterns produced with DNA microarrays and DNA chips [6].

Aiming at systematic analysis of gene interactions in *S. cerevisiae*, our research group has installed a large-scale experimental method that can generate gene expression patterns of *S. cerevisiae* by multiple gene disruptions and overexpressions. With this method, we are now on the schedule to determine a rough figure of the genetic network of *S. cerevisiae*. A task crucial to this aim is to develop strategies of experiments with which we can obtain enough gene expression pattern data to identify the underlying network. For this purpose, we modeled a genetic network as a boolean network without time delay and defined a strategy as an algorithmic process each of whose step consists of an experiment of multiple gene disruptions and overexpressions.

Fig. 1 shows an artificial example of a genetic network in the boolean network model without time delay, where each node corresponds to a gene and each gene takes one of two states: 0 (not-express) or 1 (express). The network in Fig. 1 is interpreted in the following manner in this model: We assume that `A` and `C` express under no condition. Arrows with $\oplus$ and $\ominus$ mean *activation* and *inactivation*, respectively. Gene `B` expresses if gene `A` expresses. Gene `F` (resp. `G`) expresses if gene `E` (resp. `F`) does not express. Gene `D` expresses if neither `B` nor `C` expresses. Gene `E` expresses if both `B` and `G` express. Table 1 shows four cases of gene expression patterns for normal, disruption of `A`, overexpression of `F`,

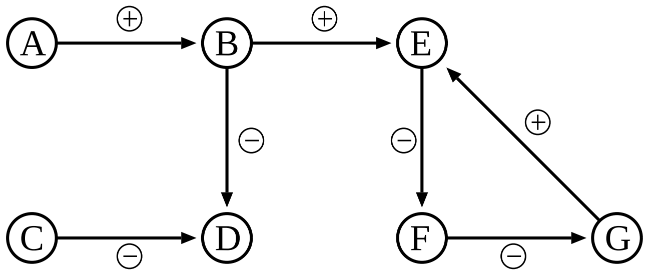

Figure 1: Boolean model of a genetic network.

and disruption of `A` and `C`. Note that, in order to activate `D`, multiple gene disruptions are required. Thus, gene disruption (overexpression) of single gene is not enough for identifying the network.

Our aim is to identify genetic networks as in Fig. 1 from experimental results of gene disruptions and overexpressions as in Table 1. Moreover, we are interested in efficient strategies of experiments. That is, we are interested in minimizing the number of experiments required to identify networks.

Computational learning of boolean functions has been extensively studied, e.g., [3, 4]. However, they are not not directly applicable, and therefore, more appropriate approaches are required for our aim. In [1], we investigated strategies of experiments for this boolean network model and proved some upper and lower bounds on the number of experiments required to determine networks. This paper continues the work of [1] from a viewpoint of practice. The model we use in this paper is a boolean network model without time delay where no time clock is assumed for the values of nodes while the synchronous boolean network model does. We employed this model in our research since the synchronous model is rather idealistic and does not fit our experimental data very well.

Table 1: Gene expressions by disruption and overexpression.

	Gene Expression						
Gene Name	A	B	C	D	E	F	G
Normal Condition	1	1	1	0	1	0	1
Disruption of A	0	0	1	0	0	1	0
Overexpression of F	1	1	1	0	0	1	0
Disruption of A and C	0	0	0	1	0	1	0

The contributions of this paper are as follows: First, we consider the relationship between our boolean network model without time delay and the synchronous boolean network model discussed in the literature [9, 13, 16, 17, 20]. In [13], some arguments are provided for the complexity of algorithmic procedures to determine the synchronous boolean networks. By establishing a relationship to the synchronous model, we derive by using the result in [1] lower and upper bounds on the number of experiments to determine synchronous boolean networks from gene expression patterns. For our system, an efficient method for simulating boolean networks without time delay plays an important role. Our second contribution is a simulator of boolean networks without time delay for multiple gene disruptions and gene overexpressions. Since the global state of a network is not necessarily stable, we need to enumerate global states of a network to analyze its behavior. An efficient enumeration technique is implemented in this simulator by using feedback vertex sets. This simulator is also useful for developing practical strategies which employ some already known genetic network models as background knowledge. The main contribution is a genetic network identifier with a graphic interface. This identifier generates instructions of experiments of gene disruptions and overexpressions. It can also be used to analyze a network by working interactively with the simulator which returns global states corresponding to the experiments. The strategy developed in [1] is a key to the development of this identifier, and the graphic interface helps viewing the network structure and its state. Some computational experiments by this system are also shown for real genetic networks.

2 Boolean Network Model without Time Delay

2.1 Model and gene disruptions and overexpressions

We first briefly review the *boolean network model without time delay* by following [1]. Readers not familiar with mathematical notation may skip mathematical descriptions and may refer only figures. A *genetic network* $G = (V, F)$ is a boolean network with the set V of nodes and the set $F = \{f_v \mid v \in V\}$ of boolean functions assigned to the nodes, where this network may have cycles but does not have self-loops. We confuse a genetic network $G = (V, F)$ with its underlying directed graph $G(V, E)$. We call a node of G a *gene*. The boolean function assigned to the node represents the condition for the gene to express and is a *gene regulation rule*. We denote a boolean function assigned to a node v by $f_v(u_1, \ldots, u_k)$ where a state of v is affected by states of $u_1, \ldots, u_k$ (i.e., $u_1, \ldots, u_k$ are input for v) (see Fig. 2). A node v with no input has a constant value (1 or 0). Unless otherwise stated, we assume without loss of generality that the value of a node with no input is 1. We say that the *state* of a gene v is *active* (*inactive*) if the value of v is 1 (0). If the value $f_v(u_1, \ldots, u_k)$ of node v is determined by the formula $l(u_1) \wedge l(u_2) \wedge \cdots \wedge l(u_k)$ ($l(u_1) \vee l(u_2) \vee \cdots \vee l(u_k)$), we call v an *AND node* (*OR node*), where $l(u_i)$ is either u_i or $\neg u_i$ (NOT u_i). We call an edge (u_i, v) an *activation edge* (*inactivation edge*) if $l(u_i)$ is a positive literal (negative literal).

For a gene v, *gene disruption* of v enforces v inactive and *gene overexpression* of v enforces v active. Let $x_1, \ldots, x_p$, $y_1, \ldots, y_q$ be mutually distinct genes of G. An *experiment* with gene overexpressions $x_1, \ldots, x_p$ and gene disruptions $y_1, \ldots, y_q$ is denoted by $e = \langle x_1, \ldots, x_p, \neg y_1, \ldots, \neg y_q \rangle$. The *cost* of e is defined by the number $p + q$.

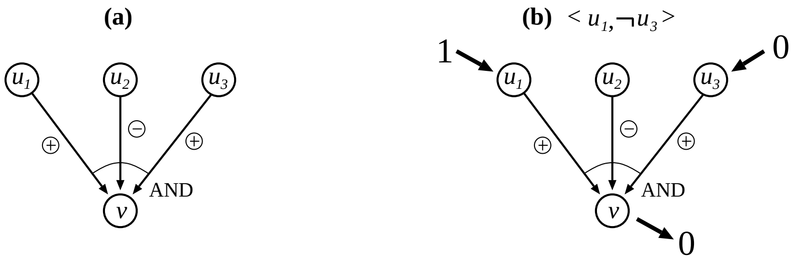

Figure 2: Example of a boolean function and an experiment. **(a)** Boolean function $u_1 \wedge \neg u_2 \wedge u_3$ (u_1 and (not u_2) and u_3) is assigned to v, where each node corresponds to a gene and this boolean function corresponds to a gene regulation rule for v. In this case, v is called an AND node and the value (i.e., state) of v is 1 if and only if the values of u_1, u_2 and u_3 are 1,0,1 respectively. **(b)** Experiment $\langle u_1, \neg u_3 \rangle$. In this case, the values of u_1 and u_3 are enforced to be 1 and 0 respectively. And then, the value of v becomes 0 because the value of u_3 is 0. Note that under an experiment $\langle u_1, u_3, \neg u_2 \rangle$, the value of v becomes 1.

A *global state* of G is a mapping $\psi : V \to \{0, 1\}$. Each global state must satisfy $\psi(x_i) = 1$ and $\psi(y_i) = 0$ under an experiment $\langle x_1, \ldots, x_p, \neg y_1, \ldots, \neg y_q \rangle$. The global state of the genes need not be consistent with the gene regulation rules. We say that a global state ψ of G is *stable* under an experiment $\langle x_1, \ldots, x_p, \neg y_1, \ldots, \neg y_q \rangle$ if it is consistent with all gene regulation rules except those assigned to nodes $x_1, \ldots, x_p, y_1, \ldots, y_q$. Otherwise, it is called *unstable*. We say that a genetic network G is *stable* under an experiment e if there is a global state of G which is stable under e. When no experiment is made on G, we simply remove "under e" from the terminology.

Note that a stable global state is not necessarily unique (Fig. 3). Moreover there may not exist any stable global state (Fig. 3). As a result of an experiment, however, we assume that all the states of the genes can be observed. Hence we should define an observed global state.

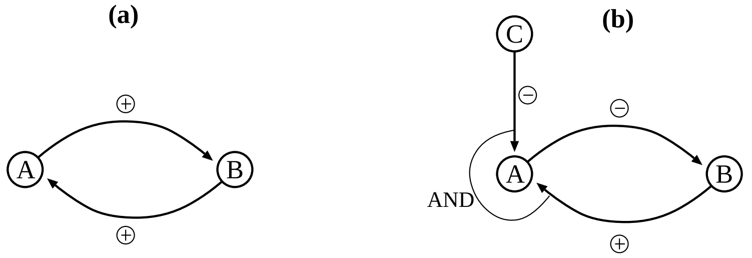

Figure 3: Global state and stable state. There are two stable global states ($[A = 0,\ B = 0]$ and $[A = 1,\ B = 1]$) for Network (a) because $A = 1$ if and only if $B = 1$. There is one stable global state $[A = 0,\ B = 1,\ C = 1]$ for Network (b) because $C = 1$ from the assumption and $A = 0$ and $B = 1$ are consistent with regulation rules. However, under experiment $e = \langle \neg C \rangle$ (i.e., C is disrupted), there is no stable global state for Network (b) because B must be 1 if $A = 0$ and A must be 1 if $B = 1$.

Before defining an observed global state, we define the set I of *invariant nodes* under an experiment $e = \langle x_1, \ldots, x_p, \neg y_1, \ldots, \neg y_q \rangle$, along with a mapping ϕ from I to $\{0, 1\}$ (see Fig. 4). Invariant nodes and ϕ are defined inductively in the following way:

1. If v appears in e or has no incoming edge (i.e., indegree = 0), v is an invariant node. Moreover, $\phi(x_i) = 1$ for $x_1, \ldots, x_p$, $\phi(y_j) = 0$ for $y_1, \ldots, y_q$, and $\phi(v) = f_v$ for the other nodes v.
2. Let U be the set of incoming nodes to v and let $U' = \{u_1, \ldots, u_h\} \subseteq U$ be the current set of invariant nodes in U, where ϕ is defined for any node in U'. If f_v with inputs $\phi(u_1), \ldots, \phi(u_h)$ is invariant for any states of nodes in $U - U'$, then v is invariant and we define $\phi(v)$ by the value of f_v.

An *observed global state* ψ under an experiment e is an arbitrary global state such that $\psi(v) = \phi(v)$ for all $v \in I$. A *native global state* of G is an observed global state when no experiment is made on G (i.e., $p = q = 0$). Note that if v is not an invariant node under the experiment then the state of v may not be determined uniquely.

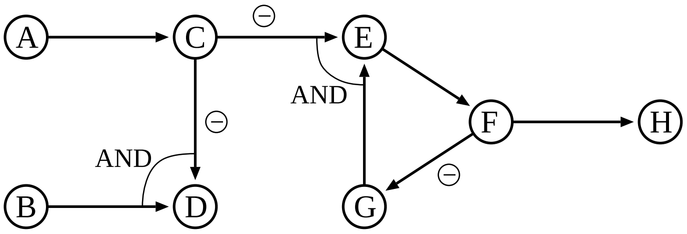

Figure 4: A native global state for this network is $[A = 1, B = 1, C = 1, D = 0, E = 0, F = 0, G = 1, H = 0]$, where all nodes are invariant nodes because the values of A and B are 1 from the assumption and then the values of the other nodes are determined uniquely. Under an experiment $e = \langle \neg C \rangle$ (i.e., C is disrupted), E, F, G, H become non-invariant nodes because their values are not determined uniquely from the same reason as in Fig. 3(b), and thus any states for E, F, G, H can be observed in this case. Note that we omit $\oplus$ symbols for activation edges.

2.2 Upper and lower bounds of experiments for network identification

We have studied in [1] the problems and algorithms for identifying genetic networks from gene expression data in various situations. The *indegree* of a gene is the number of genes directly affecting it in a genetic network. The indegree of most genes may be only one or two except some special genes. Therefore, it has an important sense in practice to cope with genetic networks with a small indegree constraint when the genes are restricted to a specific region. We have proved upper and lower bounds of experiments required for identifying a genetic network with n genes in regard to the indegree constraint and acyclicity. The results are summarized in Table 2.

Table 2: Summary of our previous results on identification, where n denotes the number of genes and D denotes the maximum indegree.

	Number of Experiments	
Constraints	Lower Bounds	Upper Bounds
No constraint	$\Omega(2^{(n-1)/2})$	$O(n\,2^{n-1})$
Indegree $\leq D$	$\Omega(n^D)$	$O(n^{2D})$
Indegree $\leq D$ & AND nodes (OR nodes) only	$\Omega(n^D)$	$O(n^{D+1})$
Indegree $\leq D$ & Acyclic	$\Omega(n^D)$	$O(n^D)$
Indegree ≤ 2 & no inactivation edges	$\Omega(n^2)$	$O(n^2)$

Note that we assume that the cost of an experiment is bounded by a constant except the first case (i.e., the case with no constraint). This assumption is natural because many changes of genes (i.e., high cost experiments) may cause the death of organisms. These results in Table 2 are rather discouraging since the lower bounds are too high for experiments even for $n \approx 6000$ and $D = 2$. However, these lower bounds hold for the worst case. Therefore, more practical (heuristic) strategies should be developed that require much fewer number of experiments for most networks. Moreover, we should make use of known genetic networks as knowledge and reduce the number of experiments significantly.

3 Relationship with Synchronous Boolean Network Model

In the synchronous boolean network model discussed in the literature [9, 13, 16, 17, 20], the state of a node changes synchronously from time to time. Let $\psi_t(v)$ denotes a state of node v at time t. Then $\psi_{t+1}(v)$ is determined by $\psi_{t+1}(v) = f_v(\psi_t(u_1), \ldots, \psi_t(u_k))$, where $u_1, \ldots, u_k$ are input nodes to v. This section relates the boolean network model without time delay and the *synchronous boolean network model* and show that our model is useful to analyze the synchronous boolean network model.

3.1 Stability and attractor

In the synchronous boolean network model, a consecutive sequence of global states $\psi_t, \psi_{t+1}, \cdots, \psi_{t+p}$ is called an *attractor* (or, an attractor cycle) with *period* p if $\psi_t(v) = \psi_{t+p}(v)$ for all genes $v \in V$. An attractor with period 1 is called a *point attractor*.

In Section 2 we defined a stable global state for the boolean network model without time delay. The following relationship holds for a stable global state and an attractor.

Proposition 1. ψ is a stable global state in a boolean network without time delay if and only if ψ_t is a point attractor in the synchronous boolean model of the same network.

Therefore, finding a point attractor is equivalent to finding a stable global state. Since finding a stable global state is known to be NP-hard [1], finding a point attractor is also NP-hard, from which the following corollary follows.

Corollary 1. Finding an attractor with the shortest period is NP-hard.

This corollary suggests that finding short attractors may take very long CPU time.

3.2 Synchronous boolean network identification

The upper and lower bound results reviewed in Section 2 are proved for the boolean network model without time delay [1]. A simple transformation from the synchronous boolean network model to the boolean network model without time delay allows us to obtain upper and lower bound results for the synchronous boolean network model.

Let $u_1, \ldots, u_k$ be input nodes to v in a synchronous network $G(V, F)$, where self-loops are allowed. For each node v, we construct two nodes v^t and v^{t+1} and we construct an edge from u_i^t to v^{t+1}. Let the constructed network (without time delay) be $G'(V', F')$ (see Fig. 5). It is easy to see that the identification strategy for $G'(V', F')$ can be directly applied to the identification of $G(V, F)$. Therefore, since $G'(V', F')$ is always acyclic, we can obtain an identification strategy for the synchronous boolean network model, which requires $O(n^D)$ experiments. Moreover, $\Omega(n^D)$ lower bound also holds for this model.

Proposition 2. Synchronous boolean networks of maximum indegree D can be identified by $\Theta(n^D)$ experiments each of whose cost is bounded by some constant.

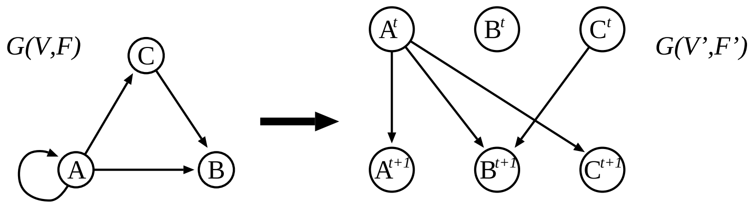

Figure 5: Transformation from a synchronous boolean network $G(V, F)$ to a boolean network without time delay $G'(V', F')$. Due to this transformation, the results on the number of experiments for identifying boolean networks without time delay can be applied to synchronous boolean networks.

4 Simulator for Gene Disruptions and Overexpressions

A simulator is developed for analyzing the behavior of a genetic network under multiple gene disruptions and overexpressions. This system is not only necessary for the genetic network identifier system, which shall be described in Section 5, but also useful for developing practical strategies for experiments.

4.1 Search for an observed global state

As a basic function, the simulator is required to compute the following problem:

Input: a boolean network $G(V, F)$ and an experiment e

Output: an observed global state ψ under e.

It is easy to compute an observed global state under e because the definition of an observed global state in Section 2 gives a straightforward algorithm running in $O(n^2)$ time. That is, we compute the states of all invariant nodes and we assign arbitrary states to the nodes other than the invariant nodes. This is realized by a rather naive implementation technique.

For example, consider the network in Fig. 4. Under a native state, the states of nodes are determined in the following order: $A = 1 \rightarrow\ B = 1 \rightarrow\ C = 1 \rightarrow\ D = 0 \rightarrow\ E = 0 \rightarrow\ F = 0 \rightarrow\ G = 1 \rightarrow\ H = 0$. Under an experiment $e = \langle \neg C \rangle$, the states of nodes are determined in the following order: $A = 1 \rightarrow\ B = 1 \rightarrow\ C = 0 \rightarrow\ D = 1$ and arbitrary states are assigned for E, F, G, H.

4.2 Enumerating all stable global states

It is an important task for the analysis of the network to test whether or not there exists a stable global state [16, 17]. Unfortunately, the problem of finding a stable global state is NP-hard [1]. Moreover, as to the number of stable global states, we have the following result (the proof is omitted; see [19] for the definition of #P-completeness):

Proposition 3. The problem of counting the number of stable global states of a genetic network is #P-complete.

Here we developed a practical method for finding a stable global state. This method is modified for enumerating all stable global states and is implemented in the simulator. Note that similar techniques have been already applied to the analysis of electronic circuits and VLSIs [12].

First note that if we examine all possible global states (i.e., 2^n global states), we can easily find a stable global state, if any. However, examining 2^n global states is time consuming even for $n \approx 20$.

Therefore, we need to reduce the number of global states to be examined. Next note that if the network is acyclic then all nodes are invariant nodes and thus a stable global state can be found by using the method described in Section 4.1. This fact leads to the following simple algorithm (Algorithm **A**, see Fig. 6):

STEP 1: Find a minimum set of nodes $U \subseteq V$ such that removal of incoming edges into U makes the network acyclic,

STEP 2: For each assignment of states to U, determine a global state using the method in Section 4.1 and check whether or not this global state is stable (i.e, consistent with all regulation rules).

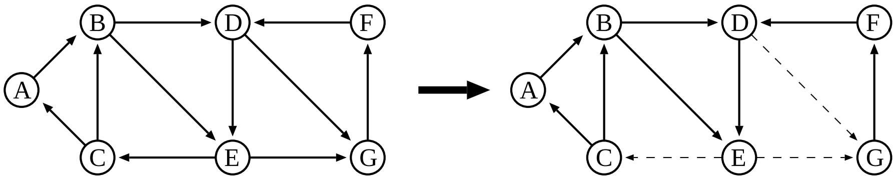

Figure 6: Finding a stable global state. In step 1, all incoming edges into $U = \{C, G\}$ are removed and thus the network becomes acyclic. In step 2, all assignments to U ($[C = 0, G = 0]$, $[C = 0, G = 1]$, $[C = 1, G = 0]$, $[C = 1, G = 1]$) are examined.

Since only $2^{|U|}$ global states are examined and the size of such U is expected to be small (compared with the size of V), this algorithm is expected to be practically efficient. However, the problem of finding such a set U is known as the FEEDBACK VERTEX SET problem and is known to be NP-hard [7]. Thus, we use the following simple greedy algorithm (Algorithm **B**) for finding a (not necessarily minimum) set U.

(i) Let $U := \{\}$,

(ii) Compute the set I of invariant nodes of the network obtained by removing all incoming edges into U,

(iii) If $V - U - I$ is empty, output U and stop,

(iv) Select an arbitrary node $v \in V - U - I$ and let $U := U \cup \{v\}$, and goto (ii).

Although nothing is theoretically proved, this greedy algorithm is expected to work well for most practical networks.

Note that the above algorithm (**A**+**B**) can be used both for finding a stable global state and for enumerating all stable global states. Moreover, the above algorithm can also be modified for enumerating attractors of a short period (i.e., attractors with period such that $|U| \times$ period $\ll n$) in the synchronous boolean network model, where we omit details here.

5 System Overview

We have developed a genetic network analyzer on SUN ULTRA workstation using C language. This analyzer consists of three parts: `Simulator`, `Identifier`, and `Graphic Interface` (Fig. 7).

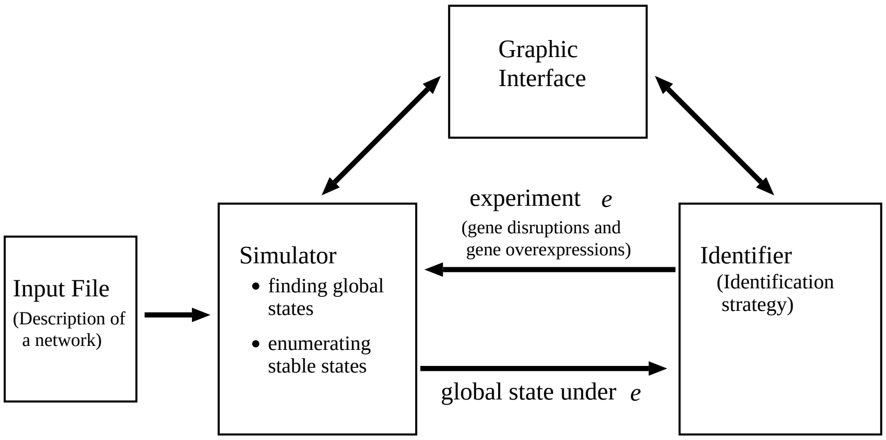

Figure 7: Overview of genetic network analyzer.

`Simulator` computes a global state and/or enumerates (or finds) stable global states of a given boolean network under an experiment specified by user or `Identifier`. Details of a boolean network is described in a flat text file (network description file), which consists of network topology, boolean functions assigned to nodes, and xy-coordinates of nodes. No graph drawing algorithm is not yet implemented and thus users have to specify xy-coordinates of nodes for drawing networks on windows. Currently, a form of a boolean function is limited to AND of literals or OR of literals.

Although `Simulator` knows the network topology and associated boolean functions, `Identifier` knows neither the network topology nor boolean functions, and only knows the number of nodes (n) in the network. When `Identifier` is used for identifying an unknown network by gene disruptions and overexpression, it generates a series of experiments in cooperation with the gene expression pattern data produced by the experiments. `Identifier` can be also used for analyzing a network by generating experiments e iteratively and by asking `Simulator` to return a global state under e. Our result in [1] is implemented for realizing `Identifier`. Currently, a strategy for the case of networks consisting of AND and/or OR nodes of maximum indegree 2, which examines $O(n^3)$ experiments (see Table 2), is implemented. Although `Identifier` can handle a case that the network contains a node of indegree > 2, `Identifier` may miss a correct boolean function for such a node. However, even for such a case, `Identifier` can always find a correct boolean function for each node of indegree ≤ 2.

`Graphic Interface` was built on X11 Window System. Using this interface, users can view the network topology, associated boolean functions and states of nodes. Moreover, using the `mouse`, users can input commands (simulation, enumeration, identification, $\cdots$) and can specify an experiment.

6 Simulation Results of Identification

In this section, we show some results of preliminary computational experiments on our strategy for identifying genetic networks. Prior to biological experiments of gene disruptions and overexpressions, we used networks described in the literature, and we examined how many experiments were required in order to identify a network, using the genetic network analyzer.

First, we used a network shown in Fig. 8 (a), which was proposed in [16] as a model of `cis regulation site`. In this case, the network was identified correctly by 12 experiments.

Next, we used a model of `Head GAP Genes Network` (Fig. 8 (b)) of Drosophila Segmentation described in [15], and a subnetwork relating with formation of `dauer larva` of `C. elegans` (Fig. 8 (c)) [10]. In the former case, the network was identified correctly by 2288 experiments except boolean functions assigned to two nodes ('btd' and 'otd') with indegree > 2. Since we only implemented an identification strategy for networks with maximum indegree 2, boolean functions assigned to nodes with indegree > 2 could not be identified correctly. In the latter case, the network was identified correctly by 1760 experiments except a boolean function assigned to one node ('daf-12') with indegree > 2.

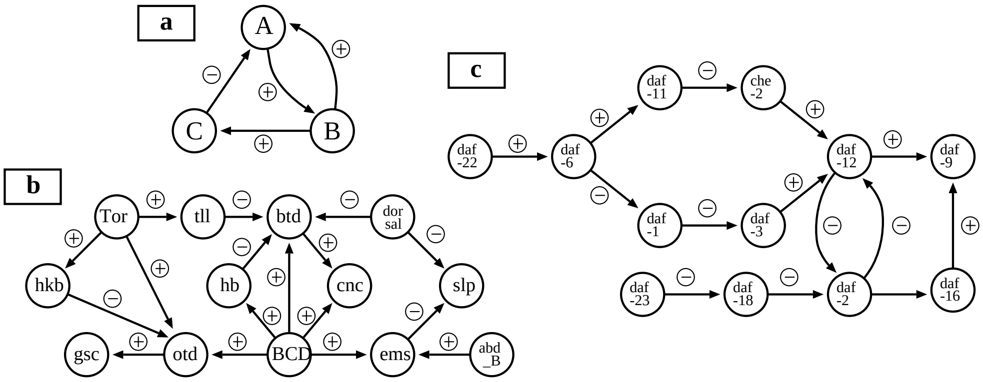

Figure 8: Genetic network models used in computational experiments: (a) simplified model of cis regulation site, (b) Head Gap genes network of Drosophila segmentation, (c) subnetwork relating with formation of dauer larva of *C. elegans*.

From the above results, it seems that our current strategy is not yet sufficient from a practical viewpoint. Although we proved rather high lower bounds, these lower bounds hold only for the worst cases. Therefore, the above results suggest that we should develop practical heuristic strategies. We believe that our analyzer will help such a development.

7 Conclusions

By employing our results on the strategies for genetic network identification [1], we have developed a prototype system that identifies genetic networks by interactively generating instructions for experiments of gene disruptions and overexpressions. We have also developed a simulator for networks that is combined with the identifier for analyzing the behavior of networks. Computational experiments and theoretical analysis show that the number of experiments required for identifying the networks is still too large.

On the other hand, we recently showed that, if we can observe time series of global states (under the synchronous boolean network model) in which states of many genes change simultaneously, we can identify the network from much fewer number of expression patterns ($\Theta(\log n)$ patterns for bounded indegree case) [2]. From this and the results of this paper, we can see that the number of experiments and/or expression patterns depends on which model we consider. Thus, it is important future work to develop a genetic network model suited for real genetic networks and technologies for biological experiments. The developed simulator will be useful for such a development.

Acknowledgments

This work was supported in part by a Grant-in-Aid "Genome Science" for Scientific Research on Priority Areas from The Ministry of Education, Science, Sports and Culture of Japan.

References

[1] Akutsu, T., Kuhara, S., Maruyama, O., and Miyano, S., Identification of Gene Regulatory Networks by Strategic Gene Disruptions and Gene Overexpressions, *Proc. Ninth ACM-SIAM Symp. Discrete Algorithms* (SODA'98), 695–702, 1998.

[2] Akutsu, T., Miyano, S., and Kuhara, S., Identification of genetic networks from a small number of gene expression patterns under the boolean network model, to appear in *Pacific Symposium on Biocomputing'99* (PSB'99).

[3] Angluin, D., Queries and concept learning, *Machine Learning*, 2:319–342, 1992.

[4] Bioch, J.C. and Ibaraki, T., Complexity of identification and dualization of positive boolean functions, *Information and Computation*, 123:50–63, 1995.

[5] Blattner, F.R. *et al.*, The complete genome sequence of Escherichia coli K-12, *Science*, 277:1453–1474, 1997.

[6] DeRisi, J.L., Iyer, V.R., and Brown, P.O., Exploring the metabolic and genetic control of gene expression on a genomic scale, *Science*, 278:680–686, 1997.

[7] Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, NY, 1979.

[8] http://genome-www.stanford.edu/Saccharomyces or http://speedy.mips.biochem.mpg.de/mips/yeast

[9] Kauffman, S.A., *The Origins of Order, Self-Organization and Selection in Evolution*, Oxford University Press, 1993.

[10] Kohara, Y. (*ed.*), *C. elegans. Symphony of 1000 cells* (in Japanese), Kyoritsu Shuppan, Tokyo, 1997.

[11] Kunst, F. *et al.*, The complete genome sequence of the gram-positive bacterium Bacillus subtilis, *Nature*, 390:249–256, 1997.

[12] Lathrop, R.H., Hall, R.J., and Kirk, R.S., Functional abstraction from structure in VLSI simulation models, *Proc. 24th ACM/IEEE Design Automation Conference*, 822–828, 1987.

[13] Liang, S., Fuhrman, S., and Somogyi, R., REVEAL, a general reverse engineering algorithm for inference of genetic network architectures, *Proc. Pacific Symposium on Biocomputing'98*, 18–29, 1998.

[14] Mewes, H.W., Albermann, K., Bahr, M., Frishman, D., Gleissner, A., Hani, J., Heumann, K., Kleine, K., Maierl, A., Oliver, S.G., Pfeiffer, F., and Zollner, A., Overview of the yeast genome, *Nature*, 387(6632 Suppl.):7–65, 1995.

[15] Samsonova, M.G., Spirov, A.V., and Serov, V.N., The GeNet database, *Poster Abstracts for Pacific Symposium on Biocomputing'98*, 110, 1998.

[16] Somogyi, R. and Sniegoski, C.A., Modeling the complexity of genetic networks: Understanding multigene and pleiotropic regulation, *Complexity*, 1:45–63, 1996.

[17] Thomas, R., Thieffry, D., and Kauffman, M., Dynamical behaviour of biological regulatory networks -I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state, *Bulletin of Mathematical Biology*, 57:247–276, 1995.

[18] Thomas, R., Thieffry, D., and Kauffman, M., Dynamical behaviour of biological regulatory networks -II. Immunity control in bacteriophage lambda, *Bulletin of Mathematical Biology*, 57:277–297, 1995.

[19] Valiant, L.G., The complexity of computing the permanent, *Theoretical Computer Science*, 8:189–201, 1979.

[20] Wuensche, A., Genomic regulation modeled as a network with basins of attraction, *Proc. Pacific Symposium on Biocomputing'98*, 89–102, 1998.