Hindawi Publishing Corporation
EURASIP Journal on Bioinformatics and Systems Biology
Volume 2008, Article ID 521407, 16 pages
doi:10.1155/2008/521407

Research Article

Algorithms and Complexity Analyses for Control of Singleton Attractors in Boolean Networks

Morihiro Hayashida,[1] Takeyuki Tamura,[1] Tatsuya Akutsu,[1] Shu-Qin Zhang,[2] and Wai-Ki Ching[3]

[1] *Bioinformatics Center, Laboratory of Biological Information Networks, Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Kyoto 611-0011, Japan*
[2] *School of Mathematical Sciences, Fudan University, Shanghai 200433, China*
[3] *Advanced Modeling and Applied Computing Laboratory, Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong*

Correspondence should be addressed to Takeyuki Tamura, tamura@kuicr.kyoto-u.ac.jp

Received 12 January 2008; Revised 4 April 2008; Accepted 2 June 2008

Recommended by Ilya Shmulevich

A Boolean network (BN) is a mathematical model of genetic networks. We propose several algorithms for control of singleton attractors in BN. We theoretically estimate the average-case time complexities of the proposed algorithms, and confirm them by computer experiments. The results suggest the importance of gene ordering. Especially, setting internal nodes ahead yields shorter computational time than setting external nodes ahead in various types of algorithms. We also present a heuristic algorithm which does not look for the optimal solution but for the solution whose computational time is shorter than that of the exact algorithms.

1. Introduction

One of the important challenges of computational systems biology and bioinformatics is to develop a control theory for biological systems [1, 2]. Development of such a control theory is interesting from both a theoretical viewpoint and a practical viewpoint. From a theoretical viewpoint, biological systems are highly nonlinear. For control of linear systems, extensive studies have been done, and rigorous theories and useful methods have been developed. Furthermore, many of these methods have been applied to control various kinds of real systems. However, it is recognized that control of nonlinear systems is far more difficult than control of linear systems. Though there are some established methods for control of nonlinear systems [3, 4], these can only be applied to certain classes/special cases. In particular, it is very difficult to control large-scale nonlinear systems. From a practical viewpoint, as Kitano wrote [1, 2], identification of a set of perturbations that induces desired changes in cellular behaviors may be useful for systems-based drug discovery and cancer treatment. For example, Takahashi (this author along with Morihiro Hayashida contributed equally to this work) and Yamanaka developed induced pluripotent stem cells (iPS cells) by introducing 4 kinds of transcription factors (Oct3/4, Sox2, c-Myc, Klf4) into fibroblast cells of mouse [5]. Furthermore, Takahashi et al. [6] and Yu et al. [7] independently succeeded to develop iPS cells by introducing 4 kinds of factors into human cells. It is to be noted that Yamanaka et al. introduced 4 transcription factors of Oct3/4, Sox2, c-Myc, and Klf4 into fibroblast cells, whereas Thomson et al. introduced 4 factors of OCT4, SOX2, NANOG, and LIN28 into somatic cells. Though these seminal discoveries were achieved based on their knowledge, experience, and many experiments, systematic methods might help such kind of works. Therefore, we study systematic methods for control of biological systems. In this paper, we focus on control of gene regulatory networks because these networks play a fundamental role in cells and may be efficiently controlled by overexpression and suppression of genes.

Various kinds of mathematical models have been proposed for modeling gene regulatory networks. These models include neural networks, differential equations, Petri nets,

Boolean networks, probabilistic Boolean networks (PBNs), and multivariate Markov chain model [8–11]. Among these models, *Boolean network* (BN) [12–14] has been well studied. BN is a very simple model; each node (e.g., gene) takes either 0 (inactive) or 1 (active), and the states of nodes change synchronously. Although BN is very simple, its dynamic process is complex and can give insight into the global behavior of large genetic regulatory networks [15].

The total number of possible global states for a Boolean network with n genes is 2^n. However, for any initial condition, the system will eventually evolve into a limited set of stable states called *attractors*. The set of states that can lead the system to a specific attractor is called the *basin of attraction*. Each attractor can contain one or many states. An attractor having only one state is called a *singleton attractor*. Otherwise, it is called a *cyclic attractor*. Attractors are biologically interpreted so that different attractors correspond to different cell types [14] or different cell states [16].

Motivated by this biological interpretation, extensive studies have been done on the average-case analysis of the number and length of attractors in randomly generated BNs [14, 17–19], although there is no conclusive result. Recently, several methods have been developed for efficiently finding or enumerating attractors in BNs [20–23], whereas it is known that finding a singleton attractor (i.e., a fixed point) is NP-hard [24, 25]. Devloo et al. developed a method using transformation to a constraint satisfaction problem [20]. Garg et al. developed a method based on binary decision diagrams (BDDs) [21]. Irons developed a method that makes use of small subnetworks [22]. However, theoretical analysis of the average-case complexity was not addressed in these works. We recently developed algorithms for identifying singleton attractors and small attractors, and analyzed the average-case time complexities of these algorithms [23].

Finding a sequence of control actions for BNs is another important topic on BNs. Datta et al. proposed methods for finding control actions for probabilistic Boolean networks (PBNs) [26–28], where a PBN is a probabilistic extension of a BN [29]. In their approach, the control problem is defined as minimization of the total of control cost and the cost of terminal state. The control cost is defined as the cost of applying control inputs in some particular states, and higher terminal costs are usually assigned to those undesirable states. Their approach is based on the theory of controlled Markov chains, and makes use of the theory of probabilistic dynamic programming. They extended their approach for handling context-sensitive PBNs [30] and/or infinite-horizon optimal control [31]. Since BNs are special cases of PBNs, their methods can also be applied to finding control actions for BNs. However, all of these approaches need to handle $2^n \times 2^n$ matrices, which limits application of these approaches only to small size (e.g., less than 20 nodes) networks. Therefore, we studied computational complexity of the control problem on BN and PBN, and proved that finding an optimal control strategy is NP-hard for both BN and PBN [32]. In order to break the barrier of computational complexity, an approximate finite-horizon optimal control has been introduced [33] and a heuristic method based on Q-learning algorithm for approximating the optimal infinite-horizon control policy has been proposed [34]. However, application of these approaches is still limited to small networks.

In this paper, we propose a new model for control of BN, that is, *control of attractors* of BN. Though our model can be extended to cyclic attractors to some extent (as shown in Section 3.9), here we focus on singleton attractors. Since cyclic attractors correspond to cell cycles appearing in such cases as cell division and cell growth whereas singleton attractors correspond to steady states of cells or cell types, it is reasonable to begin with singleton attractors. We assume that a BN and a score function are given as an input, where the score function indicates the closeness of the attractor state to the desired state. We also assume that nodes in a BN are divided into *internal nodes* and *external nodes*, where states of external nodes can only be controlled. Then, our objective is to determine 0/1 states of external nodes so that the score of the resulting singleton attractor is maximized. However, if there exist multiple attractors, the attractor into which a BN is evolved depends on an initial state of a BN. Since it is very difficult to know the initial state exactly, we modify the objective so that the minimum score of the singleton attractors is maximized or exceeds a given threshold. In this model, external nodes correspond to candidate genes and/or transcription factors to be added or to be deleted (suppressed), and the objective is to make a cell to go to a preferable state regardless of the current state of the cell.

In order to solve the proposed problem, we develop several algorithms based on our previous work [23]. In [23], we developed a series of algorithms for finding singleton and small attractors in a BN. The most important feature of the algorithms is that the average-case time complexity was theoretically analyzed and was experimentally corroborated. It was shown that most of these are much faster than $O(2^n)$ if the maximum indegree is bounded by some constant K. For example, one of the algorithms works in $O(1.19^n)$ time and $O(1.27^n)$ time (in the average case) for $K = 2$ and $K = 3$, respectively, which are much faster than $O(2^n)$. Many of the algorithms proposed in this paper have similar properties. For example, it is shown that one of the algorithms works in $O(1.266^n)$ and $O(1.393^n)$ times for $K = 2$ and $K = 3$, respectively, under some reasonable conditions. Though these time complexities are worse than those in [23], the problem considered in this paper is much more difficult than the one in [23]. Therefore, these results are reasonable and are still much faster than $O(2^n)$. It is to be noted that some of the proposed algorithms are far from straightforward extensions of [23], and novel ideas are introduced in some of the theoretical analyses. Most of the theoretical results are corroborated through computational experiments.

It is to be noted that the state-space-based methods [26–28, 31, 33] need at least $O(2^n)$ time. Though a Q-learning-based method [34] needs polynomial update time, it seems that an exponential number of repetitions are required to obtain preferable control actions. Our proposed model may be interpreted as a variant of the infinite-horizon control model [31]. However, our developed algorithms are quite different from those in [31]. Though our proposed algorithms are based on [23], the problems to be solved

are different from those in [23] and several new ideas are introduced in development of the algorithms. As a related work, Pal et al. studied the problem of generating BNs with a prescribed attractor structure [28]. Though their model has some similarity with our model, applicability of their methods is limited to small size networks.

The organization of the paper is as follows. First, we briefly review BN and then give a formal definition of the problem. Next, we present our proposed algorithms, their theoretical analyses, and the results on computational experiments. Then, we present an approximate but faster heuristic algorithm. Finally, we conclude with future work.

2. Problem of Controlling Singleton Attractors

In this section, we briefly review the Boolean network model, and then formulate the problem explained above. After that we present enumeration-based algorithms and perform theoretical and empirical analyses.

2.1. Boolean Network and Attractor

Let $G(V, F)$ represent a Boolean network which consists of a set of n nodes, $V = \{v_1, v_2, \ldots, v_n\}$, and n Boolean functions, $F = \{f_1, f_2, \ldots, f_n\}$. Generally, V and F are regarded as genes and a set of regulatory rules of genes, respectively. Let $v_i(t)$ denote the state of v_i at the time step t, where $v_i(t) = 0$ means that the ith gene is not expressed, and $v_i(t) = 1$ means that it is expressed. The overall expression level of all genes in the BN is represented by $\text{gap}(t) = [v_1(t), v_2(t), \ldots, v_n(t)]$, which is called the *gene activity profile* (GAP) of the network at time t. Since $\text{gap}(t)$ ranges from $[0, 0, \ldots, 0]$ to $[1, 1, \ldots, 1]$, there are 2^n possible global states. Regulatory rules of gene states are given as follows:

$$v_i(t+1) = f_i(v_{i_1}(t), v_{i_2}(t), \ldots, v_{i_{k_i}}(t)), \quad i = 1, \ldots, n. \quad (1)$$

This rule means that the state of gene v_i at time $t+1$ depends on the states of k_i genes at time t, where k_i is called the *indegree* of v_i. Furthermore, the maximum indegree of a BN is defined as $K = \max_i\{k_i\}$. The number of genes which are directly influenced by gene v_i is called the *outdegree* of gene v_i. The states of all genes are changed synchronously according to the corresponding Boolean functions. A consecutive sequence of GAPs $(\text{gap}(t), \text{gap}(t+1), \ldots, \text{gap}(t+p))$ is called an attractor with period p if $\text{gap}(t) = \text{gap}(t+p)$. When $p = 1$, an attractor is called a *singleton attractor*. When $p > 1$, it is called a *cyclic attractor*.

An example of a truth table of a BN is shown in Table 1. Every gene v_i updates its state according to a regulatory rule f_i. Since the state transitions of this BN are as shown in Figure 1, the system will eventually evolve into one of three attractors. Two of them are singleton attractors, $[0, 1, 1]$ and $[1, 1, 0]$. The other is a cyclic attractor with period 3, $[0, 0, 0] \rightarrow [1, 0, 0] \rightarrow [0, 1, 0]$.

In this paper, we assume that there are two types of nodes in a BN: *external nodes* and *internal nodes*. Let $v_{e,1}, v_{e,2}, \ldots, v_{e,m}$ and $v_{i,1}, v_{i,2}, \ldots, v_{i,n}$ be external and internal nodes of a BN, respectively. Note that the total number of nodes in a BN is $m + n$ hereafter. When it is not necessary to distinguish internal and external nodes, $v_1, v_2, \ldots, v_{m+n}$ are used to specify nodes. Furthermore, let $\text{gap}(t, \text{ex})$ and $\text{gap}(t, \text{in})$ denote $[v_{e,1}(t), v_{e,2}(t), \ldots, v_{e,m}(t)]$ and $[v_{i,1}(t), v_{i,2}(t), \ldots, v_{i,n}(t)]$, respectively.

Now, we formulate the main problem of this paper.

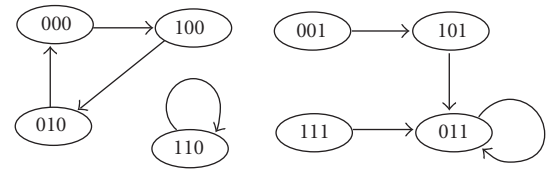

Figure 1: State transitions of the Boolean network shown in Table 1.

Table 1: Example of a truth table of a Boolean network.

v_1	v_2	v_3	f_1	f_2	f_3
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	1	1

2.2. Singleton Attractor Controlling Problem (SACP)

(i) *Input*: a Boolean network which consists of m external nodes and n internal nodes, and a score function $S(v_i, a)$, that is, a function from $V \times \{0, 1\}$ to real. We assume that Boolean functions are randomly assigned to nodes and that the parent nodes of each node are also randomly determined with $k_i \leq K$.

(ii) *Output*: a 0-1 assignment to external nodes, which maximizes the minimum score of singleton attractors, where the score of an attractor is given as $\sum_{v_i \in V} S(v_i, a)$.

For example, in a BN of Table 1, let v_1 be an external node and let v_2 and v_3 be internal nodes. Furthermore, assume that score functions of nodes of this BN are given as in Table 2. If v_1 is fixed as 0, the BN of Table 1 is converted to that shown in Table 3, and its state transition is shown in Figure 2. In this BN, there are three singleton attractors, $[0, 0, 0]$, $[0, 0, 1]$, and $[0, 1, 1]$, and their scores are $3 + 1 + 2 = 6$, $3 + 1 + 4 = 8$, and $3 + 5 + 4 = 12$, respectively. Therefore, when v_1 is fixed as 0 in the BN of Table 1, the minimum score of singleton attractors is 6. On the other hand, if v_1 is fixed as 1, the BN of Table 1 is converted to that shown in Table 4, and its state transition is shown in Figure 3. In this BN, there are two singleton attractors, $[1, 1, 0]$ and $[1, 1, 1]$, and their scores are $0 + 5 + 2 = 7$ and $0 + 5 + 4 = 9$, respectively. Therefore,

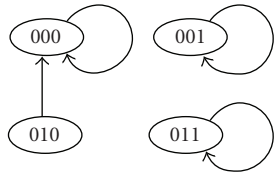

Figure 2: State transitions of the Boolean network shown in Table 3.

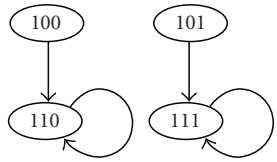

Figure 3: State transitions of the Boolean network shown in Table 4.

Table 2: Example of a score function of a Boolean network.

	v_1	v_2	v_3
0	3	1	2
1	0	5	4

Table 3: If v_1 is fixed as 0 in the truth table of Table 1, the following one is obtained.

v_1	v_2	v_3	f_1	f_2	f_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	0
0	1	1	0	1	1

Table 4: If v_1 is fixed as 1 in the truth table of Table 1, the following one is obtained.

v_1	v_2	v_3	f_1	f_2	f_3
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	1	1

when v_1 is fixed as 1 in the BN of Table 1, the minimum score of singleton attractors is 7. Thus, in order to maximize the minimum score of singleton attractors, we should fix the external node v_1 as 1 since $6 < 7$.

For this problem, one of the robust algorithms is to enumerate all singleton attractors and check the score of every singleton attractor. For this strategy, it is reasonable to utilize the basic recursive algorithm [23] as a subroutine. Although algorithms proposed in this paper are to some extent similar to those in [23], further observations and different approaches are necessary to estimate their computational time since [23] does not include the notion of external and internal nodes.

3. Enumeration-Based Algorithms

Before presenting enumeration-based algorithms for SACP, we briefly review the basic recursive algorithm in [23]. In this algorithm, partial GAPs are extended one by one towards a complete GAP according to a given gene ordering. If it is found that a partial GAP cannot be extended to a singleton attractor, the next partial GAP is examined. Although all proposed algorithms in this section are based on the same framework which includes the basic recursive algorithm as a subroutine, gene orderings are different from each other. Therefore, we explain only methods of gene ordering for most algorithms although we present the whole pseudocode of the first algorithm.

In what follows, we present algorithms for SACP and estimate their average computational time. Since some approximations are used for these theoretical analyses, each estimated computational time is not exactly the same as the result of the computer experiments shown in Section 3.8.

3.1. Algorithm 1: ExternalAhead

Theoretical Analysis

Assume that b of n internal nodes have already been examined. The overall computational time can be represented by

$$\text{time}(b = 0) + \text{time}(b = 1) + \cdots + \text{time}(b = n - 1). \quad (2)$$

The number of terms is n, and each term will be exponential function of n as shown below. The overall average time complexity will only be affected by the largest term in (2) since $n \cdot a^n \ll (a + \epsilon)^n$ holds for arbitrary $\epsilon > 0$ when $a > 1$ and n is large enough. Similar discussions will also be applied to the other algorithms.

For internal nodes, we have

$$\begin{aligned} P(v_j(t) \neq v_j(t+1)) &= 0.5 \cdot \frac{\binom{m+b}{k_i}}{\binom{m+n}{k_i}} \\ &\approx 0.5 \cdot \left(\frac{m+b}{m+n}\right)^{k_i} \\ &\geq 0.5 \cdot \left(\frac{m+b}{m+n}\right)^{K}. \end{aligned} \quad (3)$$

The probability that the algorithm examines the $(m+b+1)$th gene is not more than

$$\left\{1 - 0.5 \cdot \left(\frac{m+b}{m+n}\right)^{K}\right\}^{b}. \quad (4)$$

The number of recursive calls executed for the first $(m + b)$ genes is at most

$$2^{m+b} \cdot \left\{1 - 0.5 \cdot \left(\frac{m+b}{m+n}\right)^{K}\right\}^{b} = 2^{m} \cdot \left\{2 - \left(\frac{m+b}{m+n}\right)^{K}\right\}^{b}. \quad (5)$$

Pseudocode
Input: Boolean network (V, F) and score function $S(v_i, a)$.
Output: 0-1 assignment to external nodes, which maximizes the minimum score of singleton attractors.
Begin
Initialize $maxmin = -\infty$.
For $z = 0$ **to** $2^m - 1$ **do**
For $d = 1$ **to** m **do**
$v_{e,m-d+1}(t) =$ the dth digit of the binary number representation of z.
Initialize $min = \infty$; $g = 1$.
Procedure $MinScoreAttractor(v, g)$
If $g = n + 1$ and $score(v_{e,1}(t), \ldots, v_{e,m}(t), v_{i,1}(t), \ldots, v_{i,n}(t)) < min$, **then**
$min = score(v_{e,1}(t), \ldots, v_{e,m}(t), v_{i,1}(t), \ldots, v_{i,n}(t))$;
for $h = 0$ **to** 1 **do** $v_{i,g}(t) = h$;
if it is found that $v_{i,j}(t + 1) \neq v_{i,j}(t)$ for some $j \leq g$, **then continue**;
else $MinScoreAttractor(v, g + 1)$;
if $min \neq \infty$ and $maxmin < min$,
then $maxmin = min$;
for $d = 1$ **to** m **do**
$v'_{e,d}(t) = v_{e,d}(t)$;
if $maxmin \neq -\infty$, **then return** $\{v'_{e,1}(t), \ldots, v'_{e,m}(t)\}$;
else return null.
End

Algorithm 1: *Algorithm for gene ordering*. First, all external nodes are examined. After that all internal nodes are examined.

By setting $s = (m + b)/(m + n)$, we can obtain $b = sm + sn - m$. Furthermore, we assume that $m = \alpha n$. Therefore, (5) is rewritten as

$$2^m \cdot \{2 - s^K\}^{sm+sn-m} = 2^{\alpha n} \cdot (2 - s^K)^{\{(1+\alpha)s-\alpha\}n} = \{f(s)\}^n. \tag{6}$$

Thus, the average computational time can be estimated as

$$n \cdot \max_{\alpha/(1+\alpha) \leq s \leq 1} \{f(s)\}^n \sim O\left(\max_{\alpha/(1+\alpha) \leq s \leq 1} \{f(s)\}^n\right). \tag{7}$$

With simple numerical calculations, we can confirm that the maximum values of (6) for fixed K and α are as shown in Tables 5 and 7.

3.2. Algorithm 2: Basic

Algorithm for gene ordering. Nodes are chosen at random.

Theoretical Analysis

Assume that a of m external nodes and b of n internal nodes have already been examined. We can assume that $m/n = a/b$ holds approximately. When $a + b$ is large (compared with k),

$$\begin{aligned} P(v_j(t) \neq v_j(t+1)) &= 0.5 \cdot \frac{\binom{a+b}{k_i}}{\binom{m+n}{k_i}} \\ &\approx 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{k_i} \\ &\geq 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{K}. \end{aligned} \tag{8}$$

The probability that the algorithm examines the $(a + b + 1)$th gene is not more than

$$\left\{1 - 0.5 \cdot \left(\frac{a+b}{m+n}\right)^K\right\}^b. \tag{9}$$

The number of recursive calls executed for the first $(a + b)$ genes is at most

$$2^{a+b} \cdot \left\{1 - 0.5 \cdot \left(\frac{a+b}{m+n}\right)^K\right\}^b. \tag{10}$$

Note that the above term can be ignored when $a + b$ is small. By setting $a = mb/n$ and $m = \alpha n$, the above term can be rewritten as

$$2^{(1+\alpha)b} \cdot \left[1 - 0.5 \cdot \left\{\frac{(1+\alpha)b}{(1+\alpha)n}\right\}^K\right]^b = 2^{\alpha b} \cdot \left\{2 - \left(\frac{b}{n}\right)^k\right\}^b. \tag{11}$$

By setting $s = b/n$,

$$2^{\alpha sn}(2 - s^K)^{sn} = \left[\{2^\alpha (2 - s^K)\}^s\right]^n = g(s). \tag{12}$$

Similar to the analysis of the previous algorithm, the average computational time can be estimated as $\max_{0 \leq s \leq 1}\{g(s)\}$ and its maximum values for fixed K and α are shown in Tables 5 and 7. Note that the range of s is different from that of the previous algorithm.

Intuitively, this algorithm is the same as the basic recursive algorithm in [23]. However, the computational time depends on α since $v_i(t) = v_i(t + 1)$ always holds

Table 5: Theoretical time complexities for $K = 2$.

α	ExAhead	Basic	ExBehind	ExLastOne	LastOneAny	LastOne
0.01	1.354^n	1.353^n	1.351^n	1.350^n	1.232^n	1.231^n
0.02	1.361^n	1.359^n	1.355^n	1.353^n	1.237^n	1.234^n
0.03	1.368^n	1.365^n	1.359^n	1.357^n	1.241^n	1.238^n
0.04	1.375^n	1.370^n	1.363^n	1.360^n	1.246^n	1.242^n
0.05	1.382^n	1.376^n	1.367^n	1.363^n	1.251^n	1.245^n
0.06	1.389^n	1.383^n	1.371^n	1.367^n	1.256^n	1.249^n
0.07	1.397^n	1.389^n	1.375^n	1.370^n	1.261^n	1.253^n
0.08	1.404^n	1.395^n	1.379^n	1.374^n	1.266^n	1.256^n
0.09	1.412^n	1.401^n	1.383^n	1.377^n	1.271^n	1.260^n
0.10	1.419^n	1.407^n	1.388^n	1.381^n	1.276^n	1.264^n
0.167	1.471^n	1.451^n	1.415^n	1.406^n	1.312^n	1.290^n
0.20	1.498^n	1.473^n	1.429^n	1.419^n	1.331^n	1.303^n
0.30	1.584^n	1.544^n	1.473^n	1.464^n	1.392^n	1.343^n
0.333	1.614^n	1.569^n	1.486^n	1.480^n	1.414^n	1.357^n

Table 6: Empirical time complexities for $K = 2$.

α	ExAhead	Basic	ExBehind	ExLastOne	LastOneAny	LastOne	OutdLastOne
0.10	1.521^n	1.512^n	1.480^n	1.468^n	1.282^n	1.258^n	1.182^n
0.167	1.583^n	1.559^n	1.513^n	1.477^n	1.299^n	1.272^n	1.213^n
0.20	1.684^n	1.658^n	1.581^n	1.560^n	1.380^n	1.338^n	1.284^n

for an external node. Therefore, assigning an external node always leads to the next recursive loop, and thus the computational time becomes higher than that of the basic recursive algorithm in [23].

3.3. Algorithm 3: ExternalBehind

Algorithm for gene ordering. First all internal nodes are examined (Step 1). After that all external nodes are examined (Step 2).

Theoretical Analysis

At Step 1, the number of recursive calls executed for the first b genes is at most

$$2^b \cdot \left\{1 - 0.5 \cdot \left(\frac{b}{m+n}\right)^K\right\}^b = \left\{2 - \left(\frac{b}{m+n}\right)^K\right\}^b = f(b). \tag{13}$$

By setting $s_1 = b/(m+n)$, we can obtain $b = s_1 m + s_1 n$. Note that the definition of s is different from those of the previous algorithms. Therefore,

$$f(b) = \left(2 - {s_1}^K\right)^{s_1 m + s_1 n}. \tag{14}$$

Furthermore, by setting $m = \alpha n$,

$$f(b) = \left(2 - {s_1}^K\right)^{s_1(1+\alpha)n}. \tag{15}$$

At Step 2, the number of recursive calls executed for the first $n + a$ genes is at most

$$2^n \cdot \left\{1 - 0.5 \cdot \left(\frac{n+a}{m+n}\right)^K\right\}^n = \left\{2 - \left(\frac{n+a}{m+n}\right)^K\right\}^n = f(a). \tag{16}$$

By setting $s_2 = (n+a)/(m+n)$,

$$f(a) = \left(2 - {s_2}^K\right)^n. \tag{17}$$

The whole computational time of ExternalBehind can be bounded by

$$n \cdot \max\left\{\max_{0 \le b \le n} f(b), \max_{0 \le a \le m} f(a)\right\} \sim O\left(\max\left\{\max_{0 \le s_1 \le 1/(1+\alpha)} \left\{2 - {s_1}^K\right\}^{s_1(1+\alpha)n}, \max_{1/(1+\alpha) \le s_2 \le 1} \left(2 - {s_2}^K\right)^n\right\}\right). \tag{18}$$

It can be confirmed that the maximum values for fixed K and α are as shown in Tables 5 and 7.

3.4. Algorithm 4: ExternalLastOne

To achieve smaller time complexity, it is necessary to detect a contradiction for the condition of a singleton attractor at early stage. To detect a contradiction from a node, the node and all its parent nodes must be assigned. Therefore, one of the reasonable methods is to find an assigned node v_i for

Table 7: Theoretical time complexities for $K = 3$.

α	ExternalAhead	Basic	ExternalBehind	ExternalLastOne	LastOneAny	LastOne
0.01	1.434^n	1.434^n	1.432^n	1.431^n	1.353^n	1.351^n
0.02	1.442^n	1.440^n	1.437^n	1.434^n	1.359^n	1.356^n
0.03	1.449^n	1.447^n	1.442^n	1.438^n	1.365^n	1.361^n
0.04	1.457^n	1.454^n	1.447^n	1.442^n	1.370^n	1.365^n
0.05	1.465^n	1.461^n	1.453^n	1.446^n	1.376^n	1.370^n
0.06	1.472^n	1.468^n	1.458^n	1.450^n	1.383^n	1.375^n
0.07	1.480^n	1.475^n	1.463^n	1.454^n	1.389^n	1.379^n
0.08	1.488^n	1.482^n	1.468^n	1.457^n	1.395^n	1.384^n
0.09	1.496^n	1.489^n	1.473^n	1.461^n	1.401^n	1.389^n
0.10	1.504^n	1.496^n	1.479^n	1.465^n	1.407^n	1.393^n
0.167	1.559^n	1.545^n	1.514^n	1.492^n	1.451^n	1.426^n
0.20	1.588^n	1.570^n	1.532^n	1.506^n	1.473^n	1.442^n
0.30	1.678^n	1.649^n	1.587^n	1.550^n	1.544^n	1.493^n
0.333	1.707^n	1.677^n	1.606^n	1.566^n	1.569^n	1.510^n

Table 8: Empirical time complexities for $K = 3$.

α	ExAhead	Basic	ExBehind	ExLastOne	LastOneAny	LastOne	OutdLastOne
0.10	1.618^n	1.613^n	1.597^n	1.571^n	1.399^n	1.378^n	1.266^n
0.167	1.685^n	1.665^n	1.634^n	1.584^n	1.441^n	1.421^n	1.309^n
0.20	1.753^n	1.725^n	1.680^n	1.637^n	1.486^n	1.483^n	1.358^n

Table 9: ExternalLastOne, LastOneAny, and LastOne.

	(ii) is applied to only external nodes	(ii) is applied to both external and internal nodes
(i) is applied to only internal nodes	ExternalLastOne	LastOne
(i) is applied to both external and internal nodes		LastOneAny

which $k_i - 1$ of k_i parent nodes have already been assigned, and then assign the nonassigned node so that all parent nodes of v_i are assigned. We call such a nonassigned node *LastOne* node. In the following three algorithms, we utilize the notion of "LastOne." The frameworks of these three algorithms are the same. (i) First, a nonassigned node is randomly chosen. (ii) Second, if there is a "LastOne" node, assign it either 0 or 1. By further restricting (i) and (ii), we developed the following three algorithms as shown in Table 9.

Algorithm for gene ordering. If there is an external node v_c which satisfies the following condition, v_c is chosen to be assigned either 0 or 1. Otherwise, a nonassigned internal node is randomly chosen. *v_i and all parent nodes of v_i have already been assigned except v_c.*

If there are multiple external nodes and both of them satisfy the condition, one of them is randomly selected to be assigned. Moreover, if some external nodes are still nonassigned when all internal nodes have been assigned, remaining nodes will be randomly chosen one by one.

Example 3.1. Assume that v_2, v_3, v_4, and v_5 have already been assigned either 0 or 1 as shown in Figure 4(a). Furthermore, assume that v_1 is an external node and has not been assigned yet. In such a case, we select v_1 instead of randomly selecting a nonassigned internal node.

For another example, assume that v_3, v_4, and v_5 have been assigned as shown in Figure 4(b). Moreover, assume that both v_1 and v_2 are nonassigned external nodes. If all internal nodes have already been assigned at this point, one of v_1 and v_2 will randomly be chosen to be assigned and then the other will be assigned. However, such a case rarely happens since K is small.

Theoretical Analysis

Assume that a of m external nodes and b of n internal nodes have already been assigned. The average number of edges which are from internal nodes to v_i is $Kn/(m+n)$. The average number of internal nodes of which all parent internal nodes have already been assigned is

$$n \cdot \left(\frac{a+b}{m+n}\right)^{Kn/(m+n)}. \tag{19}$$

Since the average outdegree of an external node is also $Kn/(m+n)$,

$$a \cdot \frac{Kn}{m+n} = n \cdot \left(\frac{a+b}{m+n}\right)^{Kn/(m+n)} \cdot \frac{Km}{m+n} \tag{20}$$

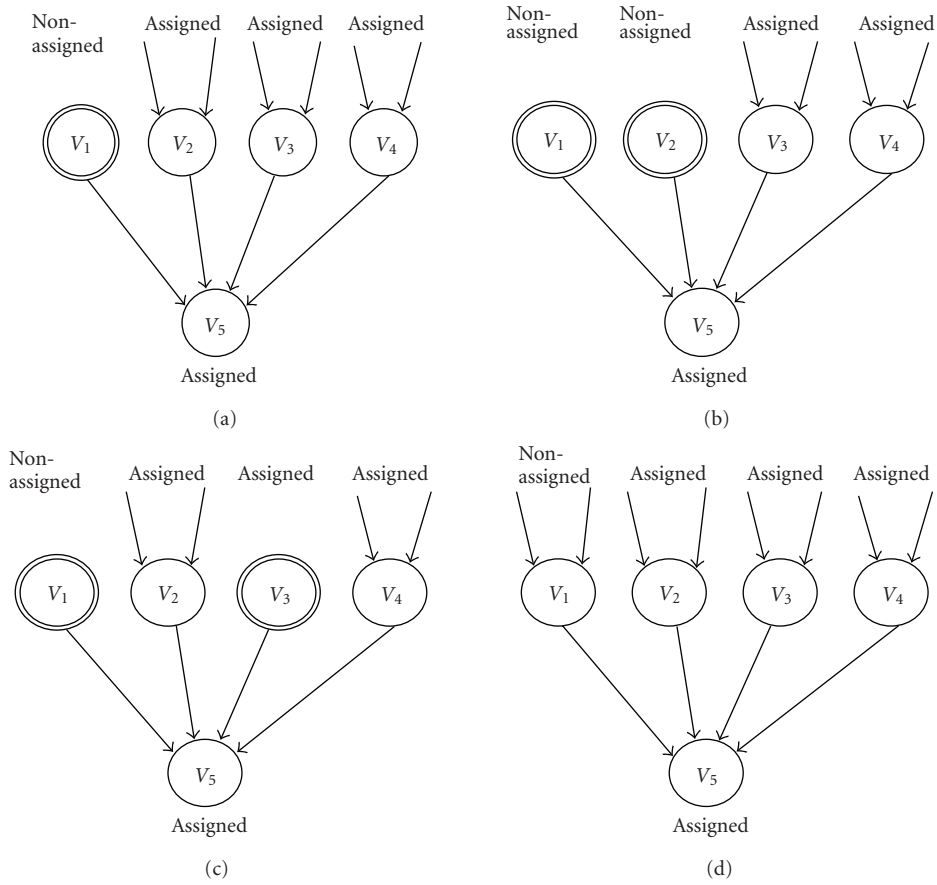

Figure 4: Example for gene ordering.

holds approximately. Therefore, we have

$$a = m \cdot \left(\frac{a+b}{m+n}\right)^{Kn/(m+n)}. \tag{21}$$

By setting $s = (a+b)/(m+n)$ and $m = \alpha n$,

$$a = \alpha n \cdot s^{K/(1+\alpha)} \tag{22}$$

holds.

On the other hand,

$$\begin{aligned} P(v_j(t) \neq v_j(t+1)) &\approx 0.5 \cdot \frac{\binom{a+b}{Kn/(m+n)}}{\binom{m+n}{Kn/(m+n)}} \\ &\approx 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{Kn/(m+n)} \end{aligned} \tag{23}$$

holds when K is small. Therefore, the probability that ExternalLastOne examines the next internal node of v_i is not more than

$$\left\{1 - 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{Kn/(m+n)}\right\}^b. \tag{24}$$

The number of recursive calls executed for the first $a+b$ nodes is at most

$$\begin{aligned} &2^{a+b} \cdot \left\{1 - 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{Kn/(m+n)}\right\}^b \\ &\quad = 2^a \cdot \left\{2 - \left(\frac{a+b}{m+n}\right)^{Kn/(m+n)}\right\}^b \\ &\quad = 2^a \cdot \left(2 - s^{K/(1+\alpha)}\right)^{s(1+\alpha)n-a} \end{aligned} \tag{25}$$

by setting $s = (a+b)/(m+n)$ and $m = \alpha n$. From (22) and (25), the computational time of ExternalLastOne can be bounded by

$$\begin{aligned} &n \cdot \max_{0 \le s \le 1} \left\{2^{\alpha n \cdot s^{K/(1+\alpha)}} \cdot \left(2 - s^{K/(1+\alpha)}\right)^{s(1+\alpha)n - \alpha n \cdot s^{K/(1+\alpha)}}\right\} \\ &\quad \sim O\left(\max_{0 \le s \le 1} \left\{2^{\alpha \cdot s^{K/(1+\alpha)}} \cdot \left(2 - s^{K/(1+\alpha)}\right)^{s(1+\alpha) - \alpha \cdot s^{K/(1+\alpha)}}\right\}^n\right). \end{aligned} \tag{26}$$

It can be confirmed that the maximum values for fixed K and α are as shown in Tables 5 and 7.

3.5. Algorithm 5: LastOneAny

Algorithm for gene ordering. If there is a node v_i of which all parent nodes have already been assigned except v_j, v_j

will be selected to be assigned either 0 or 1. Otherwise, a nonassigned node is randomly chosen to be assigned. If there are multiple nodes and both of which satisfy the above condition, one of them is randomly selected to be assigned.

Example 3.2. Assume that v_2, v_3, v_4, and v_5 have already been assigned either 0 or 1 as shown in Figure 4(c). Furthermore, assume that v_1 has not been assigned yet. In such a case, we select v_1 instead of randomly selecting a nonassigned node. Note that v_1 is not limited to an external node. Moreover, external nodes and internal nodes are not distinguished in this algorithm at all.

Theoretical Analysis

We have that

$$\begin{aligned} P(v_j(t) \neq v_j(t+1)) &\approx 0.5 \cdot \frac{\binom{a+b}{K-1}}{\binom{m+n}{K-1}} \\ &\approx 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{K-1} \end{aligned} \tag{27}$$

holds when K is small. The probability that LastOneAny examines the $(a+b+1)$th gene is not more than

$$\left\{1 - 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{K-1}\right\}^{b}. \tag{28}$$

The number of recursive calls executed at this step is at most

$$\begin{aligned} &2^{a+b+1} \cdot \left\{1 - 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{K-1}\right\}^{b} \\ &= 2^{sm+sn+1} \cdot \left(1 - 0.5 \cdot s^{K-1}\right)^{sn} \\ &= 2^{s(1+\alpha)n+1} \cdot \left(1 - 0.5 \cdot s^{K-1}\right)^{sn} \\ &= \left\{\left\{2^{\alpha} \cdot \left(2 - s^{K-1}\right)\right\}^{s}\right\}^{n} = f(s) \end{aligned} \tag{29}$$

by setting $s = (a+b)/(m+n)$, $m = \alpha n$, and $a/m = b/n$. Thus, the average computational time can be estimated as

$$n \cdot \max_{0 \le s \le 1} \{f(s)\} \sim O\left(\max_{0 \le s \le 1} \{f(s)\}\right). \tag{30}$$

With simple numerical calculations, we can confirm that the maximum values of (30) for fixed K and α are as shown in Tables 5 and 7.

3.6. LastOne

Algorithm for gene ordering. If there is a node v_j which satisfies the following condition, v_j is chosen to be assigned either 0 or 1. Otherwise, a nonassigned internal node is randomly chosen. *v_i and all its parent nodes have been assigned except v_j.*

If there are multiple nodes and both of which satisfy the above condition, one of them is randomly selected to be assigned.

Example 3.3. Assume that v_2, v_3, v_4, and v_5 have already been assigned either 0 or 1 as shown in Figure 4(d). Furthermore, assume that v_1 has not been assigned yet. In such a case, we select v_1 instead of randomly selecting a nonassigned internal node. Note that v_1 is not limited to an external node, but external nodes and internal nodes are distinguished when nonassigned nodes are randomly selected.

Theoretical Analysis

Since the average outdegree of an external node is also $Kn/(m+n)$,

$$a \cdot \frac{Kn}{m+n} = n \cdot \left(\frac{a+b}{m+n}\right)^{K-1} \cdot \frac{m}{m+n} \tag{31}$$

holds approximately. Therefore, we have

$$a = \frac{m}{K} \cdot \left(\frac{a+b}{m+n}\right)^{K-1}. \tag{32}$$

By setting $s = (a+b)/(m+n)$ and $m = \alpha n$,

$$a = \frac{\alpha n \cdot s^{K-1}}{K} \tag{33}$$

holds.

On the other hand, the probability that LastOne examines the $(a+b+1)$th gene is not more than

$$\left\{1 - 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{K-1}\right\}^{b}. \tag{34}$$

The number of recursive calls executed for the first $a+b$ genes is at most

$$\begin{aligned} 2^{a+b} \cdot \left\{1 - 0.5 \cdot \left(\frac{a+b}{m+n}\right)^{K-1}\right\}^{b} &= 2^{a} \cdot \left\{2 - \left(\frac{a+b}{m+n}\right)^{K-1}\right\}^{b} \\ &= 2^{a} \cdot \left(2 - s^{K-1}\right)^{s(1+\alpha)n-a} \end{aligned} \tag{35}$$

by using $s = (a+b)/(m+n)$. From (33) and (35), the average computational time can be estimated as

$$\begin{aligned} &n \cdot \max_{0 \le s \le 1} \left\{2^{(\alpha n \cdot s^{K-1})/K} \cdot \left(2 - s^{K-1}\right)^{\{s(1+\alpha)n - (\alpha n \cdot s^{K-1})/K\}}\right\} \\ &\sim O\left(\max_{0 \le s \le 1} \left\{\left(\frac{2}{2 - s^{k-1}}\right)^{\alpha s^{K-1}/K} \cdot \left(2 - s^{K-1}\right)^{s(1+\alpha)}\right\}^{n}\right). \end{aligned} \tag{36}$$

With simple numerical calculations, we can confirm that the maximum values of (36) for fixed K and α are as shown in Tables 5 and 7.

3.7. OutdLastOne

In addition to the above algorithms, we tried to find faster algorithms for SACP in terms of empirical time complexity.

As a result, the following algorithm yielded the best as shown in Tables 6 and 8, although theoretical analysis has not been performed. This algorithm is the extension of "outdegree-based algorithm" of [23].

Algorithm for gene ordering. If there is a node v_j which satisfies the following condition, v_j is chosen to be assigned either 0 or 1. Otherwise, a nonassigned internal node with the highest outdegree is randomly chosen. *v_i and all its parent nodes have been assigned except v_j.*

If there are multiple nodes and both of which satisfy the above condition, the one with the highest outdegree is randomly selected to be assigned.

Example 3.4. Assume that v_2, v_3, v_4, and v_5 have already been assigned either 0 or 1 as shown in Figure 4(d). Furthermore, assume that v_1 has not been assigned yet. In such a case, we select v_1 instead of randomly selecting an internal node with the highest outdegree.

3.8. Computer Experiments for Enumeration-Based Algorithms

In this section, we evaluate the proposed algorithms by performing computer experiments on random networks, and compare empirical time complexities with theoretical ones. We randomly generated 100 Boolean networks with indegree K, and took the average values. These computational experiments were done on a PC with Xeon 3.6 GHz CPUs and 3 GB RAM under the Linux (version 2.6.16) operating system, where the icc compiler (version 10.1) was used with optimization option-O3-ipo. For each K ($K = 2, 3$) and each α ($\alpha = 1/10, 1/6, 1/5$), we plotted 4 or 5 points for each method. For example, Figure 5 shows the experimental result for $K = 2$, $\alpha = 0.2$. In the experiment, we randomly generated 100 Boolean networks for $(m, n) = (1, 5), (2, 10), (3, 15), (4, 20), (5, 25)$. We used a tool for GNUPLOT to fit the function $n \log a + \log b$ to the logarithms of the experimental results. The tool uses the nonlinear least-squares (NLLSs) Marquardt-Levenberg algorithm.

As a result, empirical time complexities for each algorithm with $\alpha = 0.1, 0.167, 0.2$ and $K = 2, 3$ are shown in Tables 6 and 8. Since some approximations are used in the theoretical analyses, the theoretical time complexities shown in Tables 5 and 7 are not exactly the same as those of empirical time complexities shown in Tables 6 and 8. However, magnitude correlations of these algorithms are the same for each K and α. Furthermore, differences between theoretical time complexities and empirical time complexities are not very large for each K and α. Thus, we can say that our estimation of the theoretical time complexity of each algorithm is relatively appropriate although we used several theoretical approximations to estimate them.

3.9. Comparison Among Proposed Algorithms

As a result of theoretical and empirical analyses for the proposed algorithms for SACP, if α is not large, it is seen that "LastOne < LastOneAny < ExternalLastOne < ExternalBehind < Basic < ExternalAhead" holds in terms of necessary computational time, where A < B means that A is faster than B. One of the reasonable methods for analyzing the above result is to distinguish these algorithms by depending on whether external nodes or internal nodes are assigned first.

Let us classify these algorithms into the following three types. (i) First, assign internal nodes. After that assign external nodes. (ii) First, assign external nodes. After that assign internal nodes. (iii) Do not distinguish internal and external

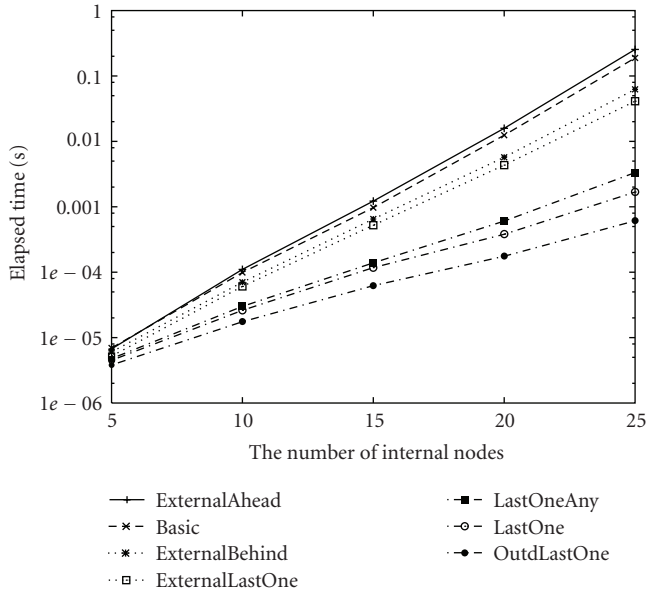

Figure 5: Elapsed time of enumeration-based algorithms for SACP with $K = 2$ and $\alpha = 0.2$.

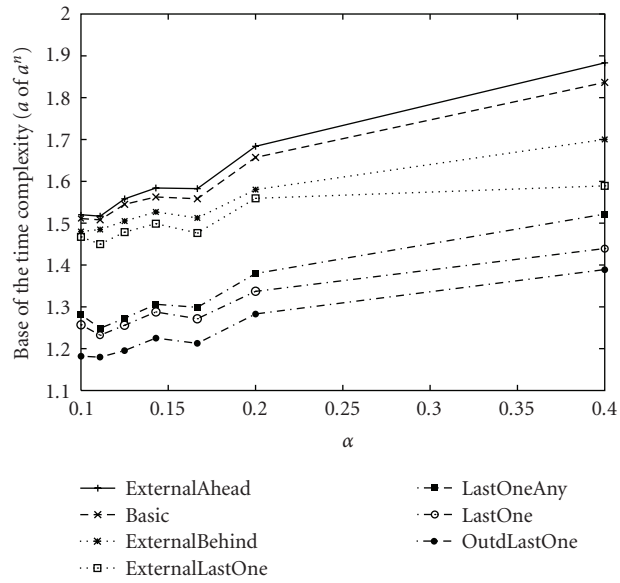

Figure 6: Base of the empirical time complexities of the enumeration-based algorithms for SACP with $K = 2$.

Table 10: Empirical time complexities for ACP(2) with $K = 2$.

α	ExAhead	Basic	ExBehind	ExLastOne	LastOneAny	LastOne	OutdLastOne
0.10	1.722^n	1.723^n	1.760^n	1.668^n	1.472^n	1.458^n	1.359^n
0.167	1.793^n	1.808^n	1.869^n	1.727^n	1.515^n	1.510^n	1.365^n
0.20	1.833^n	1.855^n	1.918^n	1.779^n	1.564^n	1.563^n	1.395^n

Table 11: Empirical time complexities for ACP(3) with $K = 2$.

α	ExAhead	Basic	ExBehind	ExLastOne	LastOneAny	LastOne	OutdLastOne
0.10	1.849^n	1.882^n	2.025^n	1.836^n	1.679^n	1.651^n	1.486^n
0.167	1.928^n	1.942^n	2.090^n	1.894^n	1.696^n	1.701^n	1.571^n
0.20	2.006^n	2.081^n	2.183^n	1.962^n	1.745^n	1.769^n	1.568^n

Table 12: Empirical time complexities of OutdLastOne for $K = 2$ with $\theta = \phi, 0.1, 0, -0.1$ and selectivity.

α	Without θ	$\theta = 0.1$	$\theta = 0$	$\theta = -0.1$
0.1	1.180^n	1.178^n	1.180^n	1.166^n
0.111	1.183^n	1.182^n	1.180^n	1.170^n
0.125	1.204^n	1.198^n	1.192^n	1.183^n
0.143	1.217^n	1.208^n	1.197^n	1.181^n
0.167	1.241^n	1.226^n	1.214^n	1.197^n
0.2	1.275^n	1.269^n	1.234^n	1.219^n

Table 13: Empirical time complexities of OutdLastOne for SACP in scale-free network.

α	OutdLastOne
0.1	1.203^n
0.167	1.262^n
0.2	1.292^n

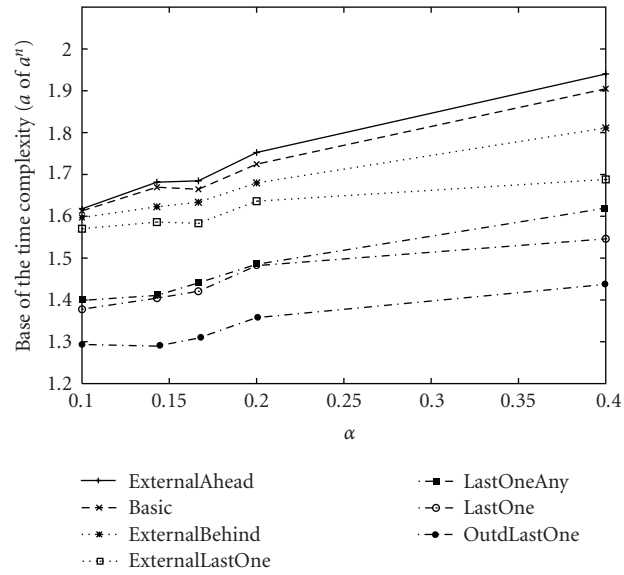

Figure 7: Base of the empirical time complexities of the enumeration-based algorithms for SACP with $K = 3$.

nodes. From " ExternalBehind < Basic < ExternalAhead", it is seen that (i) < (iii) < (ii) holds for the most basic type of algorithms. Although the other algorithms utilize the notion of "last one," they can also roughly be classified into the above three types. For example, the only difference between "LastOne" and "LastOneAny" is that "LastOne" randomly selects only internal nodes when there are no special nodes, whereas "LastOneAny" randomly selects nodes from both internal and external nodes in the same condition. Therefore, it is reasonable to regard "LastOne" and "LastOneAny" as (i) and (iii), respectively, when comparing these two and we can confirm that (i) < (iii) holds again. On the other hand, the only difference between "ExternalLastOne" and "LastOne" is that the notion of "last one node" is only applied to external nodes in "ExternalLast," whereas the notion is applied to both internal and external nodes in "LastOne". Therefore, it is also reasonable to regard "ExternalLastOne" and "LastOne" as (ii) and (iii), respectively, in this comparison, and we can confirm that (iii) < (ii) holds. Note that "LastOne" is classified into (i) in the previous comparison but is classified into (iii) this time. It depends on which two are compared. Thus, we can confirm that (i) < (iii) < (ii) holds for various types of comparisons. Intuitively, to reduce the computational time, it is necessary to detect a contradiction for the condition of a singleton attractor at early stage. To detect a contradiction from a node, the node and all its parent nodes must be assigned. However, since $v_i(t) = v_i(t + 1)$ always holds for an external node, algorithms cannot detect the contradiction from external nodes. That is why assigning internal nodes first reduces the computational time.

However, if cyclic attractors are taken into consideration, the above property does not hold. Now, we formulate the extended version of SACP as follows.

ACP(x): Attractor Controlling Problem

(i) *Input*: a Boolean network which consists of m external nodes and n internal nodes, and a score function $S(v_i, a)$, that is, a function from $V \times \{0, 1\}$ to real. We assume that Boolean functions are randomly

Table 14: Average and standard deviations of c by OutdLastOne for SACP with $K = 2$ and selectivity.

α	θ	Average of c	Standard deviation of c	The number of all singleton attractors in 100 BNs	The number of singleton attractors whose scores are more than θ in 100 BNs
0.1	−0.1	0.0228	0.0245	64	64
($m = 10$	0.0	0.0176	0.0242	64	54
$n = 100$)	0.1	0.0046	0.0124	64	14
0.111	−0.1	0.0271	0.0216	66	66
($m = 10$	0.0	0.0235	0.0217	66	58
$n = 90$)	0.1	0.0060	0.0133	66	18
0.125	−0.1	0.0329	0.0323	66	66
($m = 10$	0.0	0.0218	0.0281	66	51
$n = 80$)	0.1	0.0060	0.0195	66	12
0.143	−0.1	0.0260	0.0252	70	70
($m = 10$	0.0	0.0213	0.0235	70	63
$n = 70$)	0.1	0.0064	0.0164	70	23
0.167	−0.1	0.0294	0.0278	73	73
($m = 10$	0.0	0.0252	0.0272	73	66
$n = 60$)	0.1	0.0050	0.0118	73	27
0.2	−0.1	0.0340	0.0347	66	66
($m = 10$	0.0	0.0305	0.0325	66	61
$n = 50$)	0.1	0.0146	0.0236	66	37

Table 15: Average and standard deviations of c by OutdLastOne for SACP with $K = 3$ and selectivity.

α	θ	Average of c	Standard deviation of c	The number of all singleton attractors in 100 BNs	The number of singleton attractors whose scores are more than θ in 100 BNs
0.1	−0.1	0.0552	0.0407	94	94
($m = 10$	0.0	0.0443	0.0388	94	85
$n = 100$)	0.1	0.0108	0.0300	94	21
0.111	−0.1	0.0582	0.0403	95	95
($m = 10$	0.0	0.0476	0.0395	95	91
$n = 90$)	0.1	0.0147	0.0329	95	31
0.125	−0.1	0.0619	0.0466	93	93
($m = 10$	0.0	0.0462	0.0422	93	86
$n = 80$)	0.1	0.0199	0.0354	93	41
0.143	−0.1	0.0680	0.0405	94	94
($m = 10$	0.0	0.0526	0.0450	94	91
$n = 70$)	0.1	0.0212	0.0376	94	32
0.167	−0.1	0.0619	0.0448	94	94
($m = 10$	0.0	0.0528	0.0441	94	89
$n = 60$)	0.1	0.0206	0.0344	94	44
0.2	−0.1	0.0782	0.0502	96	96
($m = 10$	0.0	0.0661	0.0474	96	92
$n = 50$)	0.1	0.0300	0.0450	96	58

Table 16: Empirical time complexities of ExternalAhead for $K = 2$ with $\theta = 0.1, 0, -0.1$ and selectivity.

α	$\theta = 0.1$	$\theta = 0$	$\theta = -0.1$
0.1	1.514^n	1.477^n	1.492^n
0.111	1.526^n	1.499^n	1.479^n
0.125	1.560^n	1.522^n	1.536^n
0.143	1.570^n	1.574^n	1.561^n
0.167	1.616^n	1.580^n	1.577^n
0.2	1.676^n	1.624^n	1.590^n

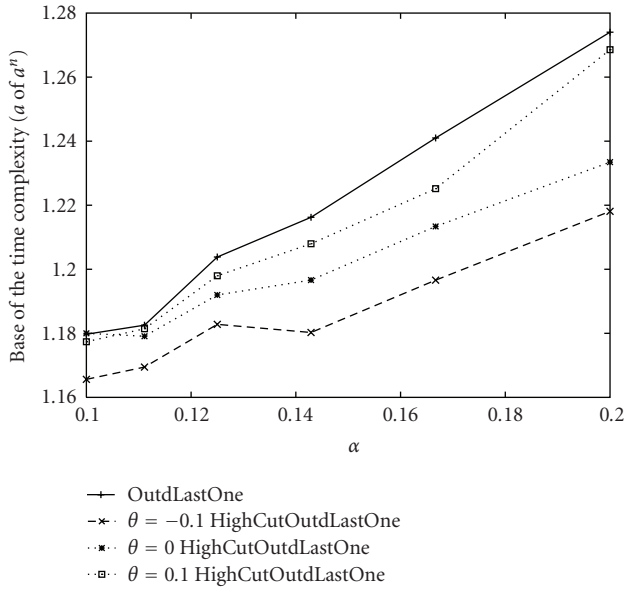

Figure 8: Base of the empirical time complexities of OutdLastOne for SACP with $K = 2$ and $\theta = \phi, -0.1, 0, 0.1$.

assigned to nodes, and parent nodes of each node are also randomly determined with $k_i \leq K$.

(ii) *Output*: a 0-1 assignment to external nodes, which maximizes the minimum score of attractors whose periods are p, where $p \leq x$. The score of an attractor is given as $\sum_{t=1}^{p} \sum_{v_i \in V} S(v_i(t), a)$.

Note that the score of a cyclic attractor is defined as the sum of the score of GAP for each t, but it can be extended to other definitions such as the sum of the minimum score of each node.

Although our proposed algorithms were introduced for SACP, we extended and implemented them for ACP(2) and ACP(3). A pseudocode of ExternalAhead for ACP(x) is shown in Algorithm 2. Although the main part of each algorithm is the same as that for SACP, the process for checking whether the partial assignments contradict the condition of attractors is different. Let *x-ancestor* of v_i be nodes which have a directed path to v_i with length less than or equal to x. For SACP, algorithms only check the relationship between the assignment of each node and its parent nodes. However, for ACP(x), algorithms check the relationship between the assignment of each node and its x-ancestors.

Empirical time complexities for ACP(2) and ACP(3) are shown in Tables 10 and 11, respectively. Since the number of *x-ancestors* is relatively large when compared with $m + n$ (around 30) for ACP(3), some elements in Table 11 are larger than $O(2^n)$. Note that these values would be less than $O(2^n)$ if $m + n$ were much larger. It seems that (ii) < (iii) < (i) holds for $x \geq 2$ since "ExternalAhead < Basic < ExternalBehind" holds in Tables 10 and 11 although the complexities of "LastOne" and "LastOneAny" are almost the same. It seems that the number of x-ancestors affects the empirical time complexities largely. For example, "ExternalAhead" is the slowest for SACP but faster than "Basic" and "ExternalBehind" for ACP(2) and ACP(3). We believe that the reason is that the number of x-ancestors of assigned nodes for "ExternalAhead" is smaller than that for "Basic" and "ExternalBehind" in the cases of ACP(2) and ACP(3), but it is larger in the case of SACP.

3.10. SACP in Scale-Free BN

It is known that gene regulatory networks have the scale-free property; that is, the degree distribution approximately follows the power law [35]. Moreover, it is observed that the outdegree distribution follows the power law and the indegree distribution follows the Poisson distribution [36]. We implemented OutdLastOne for SACP with scale-free networks, where indegrees are 2 and outdegrees are proportional to k^{-2}. (Note that this k does not mean indegrees.) The average empirical time complexities of randomly generated 100 BNs are shown in Table 13, and we can confirm that OutdLastOne in scale-free networks is almost as fast as OutdLastOne in random networks examined in Section 3.8. $(m, n) = (6, 30), (8, 40), (10, 50), (12, 60), (14, 70)$ were used for $\alpha = 0.2$, and similar numbers of nodes were also used for $\alpha = 0.1, 0.167$.

4. Heuristic Algorithms for SACP

In the previous section, we analyzed enumeration-based algorithms for SACP. Although these algorithms are guaranteed to output optimal solutions, it may not be necessary to find the rigorous optimal solutions in some practical cases. One of the possible approaches for this purpose is to use a threshold. Based on it, we develop heuristic algorithms by modifying the original algorithms. In the original algorithms, we update the minimum score whenever a new singleton attractor is found. Instead, in the modified algorithms, we compare the score of a new singleton attractor with a given threshold θ and output the corresponding assignment to external nodes as an approximate solution if the score is greater than θ. Of course, there may exist multiple attractors for each assignment to external nodes, and the minimum is taken (per assignment to external nodes) in the original algorithms. However, it is known that the expected number of singleton attractors is 1 [37, 38]. Thus, it is expected that we can obtain a good solution even

Input: a Boolean network (V, F) and score functions $S(v_i, a)$.
Output: 0-1 assignments to external nodes, which maximize the minimum score of attractors whose periods are p, where $p \leq x$. The score of an attractor is given as $\sum_{t=1}^{p} \sum_{v_i \in V} S(v_i(t), a)$.

Begin
  Define x-$parent(v_i)$: nodes which have length-x paths to v_i.
  Initialize $maxmin = -\infty$;
  for $z = 0$ **to** $2^m - 1$ **do**
    for $d = 1$ **to** m **do**
      $v_{e,m-d+1}(t) =$ the dth digit of the binary number representation of z.
    Initialize $min = \infty$; $g = 1$.
    Procedure $MinScoreAttractor(v, g)$
      If $g = n + 1$ and $\sum_{t=1}^{p} score(v_{e,1}(t), \ldots, v_{e,m}(t), v_{i,1}(t), \ldots, v_{i,n}(t)) < min$, **then**
        $min = \sum_{t=1}^{p} score(v_{e,1}(t), \ldots, v_{e,m}(t), v_{i,1}(t), \ldots, v_{i,n}(t))$;
      for $h = 0$ **to** 1 **do**
        $v_{i,g}(t) = h$;
        $flag = 0$;
        for $r = 1$ to x **do**
          $y = 1$;
          while $flag = 0$ and $y \leq g$ **do**
            if every r-$parent(v_{i,g}(t))$ is assigned and $v_{i,g}(t) \neq v_{i,g}(t + r)$ **then**
              $flag = 1$;
            $y = y + 1$;
        if $flag = 1$ **then continue**;
        else $MinScoreAttractor(v, g + 1)$;
    if $min \neq \infty$ and $maxmin < min$,
      then $maxmin = min$;
      for $d = 1$ **to** m **do**
        $v'_{e,d}(t) = v_{e,d}(t)$;
  if $maxmin \neq -\infty$, **then return** $\{v'_{e,1}(t), \ldots, v'_{e,m}(t)\}$;
  else return null.
End

Algorithm 2: Pseudocode of ExternalAhead for ACP(x).

if we stop the algorithms as soon as a singleton attractor whose score is greater than θ is found. How to select θ is also an important issue in these heuristic algorithms. If we know appropriate θ in advance, we can simply use such θ. Otherwise, we may examine several values of θ from lower to upper. For each θ, we manually inspect the solution and we stop further examinations if the solution is satisfactory.

Since there is no performance guarantee on the proposed heuristic approach, we examined it by means of computational experiments. We implemented one of the proposed heuristic algorithms assuming that $S(v_i, a)$ is distributed in $[-1, 1]$ uniformly. Furthermore, let us call the following property *selectivity*: When v_i is to be assigned, if $S(v_i, 0) > S(v_i, 1)$ holds, $v_i = 0$ is examined in advance of examining $v_i = 1$. On the other hand, if $S(v_i, 0) < S(v_i, 1)$ holds, $v_i = 1$ is examined in advance of examining $v_i = 0$. Note that the results in Tables 6 and 8 were not with selectivity.

Since OutdLastOne was the fastest among our proposed algorithms for SACP, we implemented OutdLastOne with selectivity and $\theta = \phi, -0.1, 0, 0.1$, where $\theta = \phi$ means that a threshold is not used. As a result, empirical time complexities for each α and θ are obtained as shown in Figure 8 and Table 12, and we can confirm that using a smaller threshold yields better time complexities than using a bigger threshold or not using a threshold. Furthermore, from Tables 14 and 15, it is seen that the average number of singleton attractors in a BN is less than 1 with $K = 2, 3$. Therefore, it is reasonable that the proposed algorithm stops as soon as it finds a singleton attractor whose score is greater than θ. Tables 14 and 15 also show the average and standard deviations of c for each case. It is seen that s_2 is very close to s_1 when $\theta = 0.1$. On the other hand, s_2 is much smaller than s_1 when $\theta = -0.1$. However, it often occurs that the algorithm cannot find desired singleton attractors when $\theta = 0.1$. For example, from Table 14, when $K = 2$, $\alpha = 0.1$, and $\theta = -0.1$, it is seen that the algorithm can always find desired singleton attractors if they exist. On the other hand, when the algorithm is applied to 100 random BNs with $K = 2$, $\alpha = 0.1$, $\theta = 0.1$, it can find desired singleton attractors only for 14 BNs although 64 of 100 BNs include singleton attractors.

We also implemented ExternalAhead with selectivity and a threshold for SACP. As shown in Table 16, empirical time complexities for ExternalAhead were much larger than those

of OutdLastOne with $\theta = -0.1, 0, 0.1$ and $K = 2$. It is seen that assigning internal nodes first and utilizing the notion of "LastOne" are also effective for SACP with a threshold.

5. Conclusion

In this paper, we have presented fast algorithms to find a 0-1 assignment for external nodes of a BN, which maximizes the minimum score of singleton attractors. We performed theoretical and experimental analyses for these proposed algorithms, which showed good agreements between their theoretical results and empirical results. It was also suggested that assigning internal nodes in advance of external nodes was the fastest. Furthermore, we have implemented some heuristic algorithms although theoretical analysis has not been performed. One of our future works is to extend our algorithms to a problem where it is not given which nodes are external. Furthermore, for practical use, it is important to develop a method for controlling steady states of a continuous model of biological networks. Although BN is not a continuous model, the idea based on combinatorial models may be utilized in the analysis of continuous models as in [38]. Therefore, it is also our important future work to develop a method for extending our model to continuous one.

Acknowledgments

Wai-Ki Ching supports in part by Research HK RCG Grant no. 7017/07P and HKU CRCG grants.

References

[1] H. Kitano, "Computational systems biology," *Nature*, vol. 420, no. 6912, pp. 206–210, 2002.

[2] H. Kitano, "Cancer as a robust system: implications for anticancer therapy," *Nature Reviews Cancer*, vol. 4, no. 3, pp. 227–235, 2004.

[3] A. Isidori, *Nonlinear Control Systems II*, Springer, Berlin, Germany, 1995.

[4] H. Nijmeijer and A. J. van der Schaft, *Nonlinear Dynamical Control Systems*, Springer, New York, NY, USA, 1990.

[5] K. Takahashi and S. Yamanaka, "Induction of pluripotent stem cells from mouse embryonic and adult fibroblast cultures by defined factors," *Cell*, vol. 126, no. 4, pp. 663–676, 2006.

[6] K. Takahashi, K. Tanabe, M. Ohnuki, et al., "Induction of pluripotent stem cells from adult human fibroblasts by defined factors," *Cell*, vol. 131, no. 5, pp. 861–872, 2007.

[7] J. Yu, M. A. Vodyanik, K. Smuga-Otto, et al., "Induced pluripotent stem cell lines derived from human somatic cells," *Science*, vol. 318, no. 5858, pp. 1917–1920, 2007.

[8] W.-K. Ching, M. M. Ng, E. S. Fung, and T. Akutsu, "On construction of stochastic genetic networks based on gene expression sequences," *International Journal of Neural Systems*, vol. 15, no. 4, pp. 297–310, 2005.

[9] H. de Jong, "Modeling and simulation of genetic regulatory systems: a literature review," *Journal of Computational Biology*, vol. 9, no. 1, pp. 67–103, 2002.

[10] P. Smolen, D. A. Baxter, and J. H. Byrne, "Mathematical modeling of gene networks," *Neuron*, vol. 26, no. 3, pp. 567–580, 2000.

[11] H. Matsuno, Y. Tanaka, H. Aoshima, A. Doi, M. Matsui, and S. Miyano, "Biopathways representation and simulation on hybrid functional Petri net," *In Silico Biology*, vol. 3, no. 3, pp. 389–404, 2003.

[12] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 437–467, 1969.

[13] S. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, no. 5215, pp. 177–178, 1969.

[14] S. A. Kauffman, *The Origins of Order: Self-Organization and Selection in Evolution*, Oxford University Press, New York, NY, USA, 1993.

[15] D. Thieffry, A. M. Huerta, E. Pérez-Rueda, and J. Collado-Vides, "From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in *Escherichia coli*," *BioEssays*, vol. 20, no. 5, pp. 433–440, 1998.

[16] S. Huang, "Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery," *Journal of Molecular Medicine*, vol. 77, no. 6, pp. 469–480, 1999.

[17] M. Aldana, "Boolean dynamics of networks with scale-free topology," *Physica D*, vol. 185, no. 1, pp. 45–66, 2003.

[18] B. Drossel, T. Mihaljev, and F. Greil, "Number and length of attractors in a critical Kauffman model with connectivity one," *Physical Review Letters*, vol. 94, no. 8, Article ID 088701, 4 pages, 2005.

[19] B. Samuelsson and C. Troein, "Superpolynomial growth in the number of attractors in Kauffman networks," *Physical Review Letters*, vol. 90, no. 9, Article ID 098701, 4 pages, 2003.

[20] V. Devloo, P. Hansen, and M. Labbé, "Identification of all steady states in large networks by logical analysis," *Bulletin of Mathematical Biology*, vol. 65, no. 6, pp. 1025–1051, 2003.

[21] A. Garg, I. Xenarios, L. Mendoza, and G. DeMicheli, "An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments," in *Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology (RECOMB '07)*, vol. 4453 of *Lecture Notes in Computer Science*, pp. 62–76, Springer, Oakland, Calif, USA, April 2007.

[22] D. J. Irons, "Improving the efficiency of attractor cycle identification in Boolean networks," *Physica D*, vol. 217, no. 1, pp. 7–21, 2006.

[23] S.-Q. Zhang, M. Hayashida, T. Akutsu, W.-K. Ching, and M. K. Ng, "Algorithms for finding small attractors in Boolean networks," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2007, Article ID 20180, 13 pages, 2007.

[24] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano, "A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions," *Genome Informatics*, vol. 9, pp. 151–160, 1998.

[25] M. Milano and A. Roli, "Solving the satisfiability problem through Boolean networks," in *Proceedings of the 6th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*, vol. 1792 of *Lecture Notes in Artificial Intelligence*, pp. 72–83, Bologna, Italy, September 2000.

[26] A. Datta, A. Choudhary, M. L. Bittner, and E. R. Dougherty, "External control in Markovian genetic regulatory networks," *Machine Learning*, vol. 52, no. 1-2, pp. 169–191, 2003.

[27] A. Datta, A. Choudhary, M. L. Bittner, and E. R. Dougherty, "External control in Markovian genetic regulatory networks: the imperfect information case," *Bioinformatics*, vol. 20, no. 6, pp. 924–930, 2004.

[28] R. Pal, I. Ivanov, A. Datta, M. L. Bittner, and E. R. Dougherty, "Generating Boolean networks with a prescribed attractor structure," *Bioinformatics*, vol. 21, no. 21, pp. 4021–4025, 2005.

[29] 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, no. 2, pp. 261–274, 2002.

[30] R. Pal, A. Datta, M. L. Bittner, and E. R. Dougherty, "Intervention in context-sensitive probabilistic Boolean networks," *Bioinformatics*, vol. 21, no. 7, pp. 1211–1218, 2005.

[31] R. Pal, A. Datta, and E. R. Dougherty, "Optimal infinite-horizon control for probabilistic Boolean networks," *IEEE Transactions on Signal Processing*, vol. 54, no. 6, part 2, pp. 2375–2387, 2006.

[32] T. Akutsu, M. Hayashida, W.-K. Ching, and M. K. Ng, "Control of Boolean networks: hardness results and algorithms for tree structured networks," *Journal of Theoretical Biology*, vol. 244, no. 4, pp. 670–679, 2007.

[33] M. K. Ng, S.-Q. Zhang, W.-K. Ching, and T. Akutsu, "A control model for Markovian genetic regulatory networks," in *Transactions on Computational Systems Biology V*, vol. 4070 of *Lecture Notes in Computer Science*, pp. 36–48, Springer, Berlin, Germany, 2006.

[34] B. Faryabi, A. Datta, and E. R. Dougherty, "On approximate stochastic control in genetic regulatory networks," *IET Systems Biology*, vol. 1, no. 6, pp. 361–368, 2007.

[35] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[36] N. Guelzim, S. Bottani, P. Bourgine, and F. Képès, "Topological and causal structure of the yeast transcriptional regulatory network," *Nature Genetics*, vol. 31, no. 1, pp. 60–63, 2002.

[37] I. Harvey and T. Bossomaier, "Time out of joint: attractors in asynchronous random Boolean networks," in *Proceedings of the 4th European Conference on Artificial Life (ECAL '97)*, pp. 67–75, MIT Press, Brighton, UK, July 1997.

[38] A. Mochizuki, "An analytical study of the number of steady states in gene regulatory networks," *Journal of Theoretical Biology*, vol. 236, no. 3, pp. 291–310, 2005.