Hindawi Publishing Corporation
EURASIP Journal on Bioinformatics and Systems Biology
Volume 2007, Article ID 20180, 13 pages
doi:10.1155/2007/20180

Research Article

Algorithms for Finding Small Attractors in Boolean Networks

Shu-Qin Zhang,[1] Morihiro Hayashida,[2] Tatsuya Akutsu,[2] Wai-Ki Ching,[1] and Michael K. Ng[3]

[1] *Advanced Modeling and Applied Computing Laboratory, Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong*
[2] *Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Kyoto 611-0011, Japan*
[3] *Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong*

Received 29 June 2006; Revised 24 November 2006; Accepted 13 February 2007

Recommended by Edward R. Dougherty

A Boolean network is a model used to study the interactions between different genes in genetic regulatory networks. In this paper, we present several algorithms using gene ordering and feedback vertex sets to identify singleton attractors and small attractors in Boolean networks. We analyze the average case time complexities of some of the proposed algorithms. For instance, it is shown that the outdegree-based ordering algorithm for finding singleton attractors works in $O(1.19^n)$ time for $K = 2$, which is much faster than the naive $O(2^n)$ time algorithm, where n is the number of genes and K is the maximum indegree. We performed extensive computational experiments on these algorithms, which resulted in good agreement with theoretical results. In contrast, we give a simple and complete proof for showing that finding an attractor with the shortest period is NP-hard.

1. INTRODUCTION

The advent of DNA microarrays and oligonucleotide chips has significantly sped up the systematic study of gene interactions [1–4]. Based on microarray data, different kinds of mathematical models and computational methods have been developed, such as Bayesian networks, Boolean networks and probabilistic Boolean networks, ordinary and partial differential equations, qualitative differential equations, and other mathematical models [5]. Among all the models, the *Boolean network* model has received much attention. It was originally introduced by Kauffman [6–9] and reviews can be found in [10–12]. In a Boolean network, gene expression states are quantized to only two levels: 1 (expressed) and 0 (unexpressed). Although such binary expression is very simple, it can retain meaningful biological information contained in the real continuous-domain gene expression profiles. For instance, it can be applied to separation between types of gliomas and types of sarcomas [13].

In a Boolean network, genes interact through some logical rules called *Boolean functions*. The state of a target gene is determined by the states of its regulating genes (input genes) and its Boolean function. Given the states of the input genes, the Boolean function transforms them into an output, which is the state of the target gene. Although the Boolean network model is very simple, its dynamic process is complex and can yield insight to the global behavior of large genetic regulatory networks [14].

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*. There can be one or many states for each attractor. An attractor having only one state is called a singleton attractor. Otherwise, it is called a cyclic attractor.

There are two different interpretations for the function of attractors. One intuition that follows Kauffman is that one attractor should correspond to a cell type [11]. Another interpretation of attractors is that they correspond to the cell states of growth, differentiation, and apoptosis [10]. Cyclic attractors should correspond to cell cycles (growth) and singleton attractors should correspond to differentiated or apoptosis states. These two interpretations are complementary since one cell type can consist of several neighboring attractors and each of them corresponds to different cellular functional states [15].

The number and length of attractors are important features of networks. Extensive studies have been done for analyzing them. Starting from [11], a fast increase of the number

of attractors has been seen in [16–19]. Many studies have also been done on the mean length of attractors [11, 17], although there is no conclusive result.

It is also important to identify attractors of a given Boolean network. In particular, identification of all singleton attractors is important because singleton attractors correspond to steady states in Boolean networks and have close relation with steady states in other mathematical models of biological networks [10, 20–23]. As mentioned before, Huang wrote that singleton attractors correspond to differentiation and apoptosis states of a cell [10]. Devloo et al. transforms the problem of finding steady states for some types of biological networks to a constraint satisfaction problem [20]. The resulting constraint satisfaction problem is very close to the problem of identification of singleton attractors in Boolean networks. Mochizuki introduced a general model of genetic networks based on nonlinear differential equations [21]. He analyzed the number of steady states in that model, where steady states are again closely related to singleton attractors in Boolean networks. Zhou et al. proposed a Bayesian-based approach to constructing probabilistic genetic networks [23]. Pal et al. proposed algorithms for generating Boolean networks with a prescribed attractor structure [22]. These studies focus on singleton attractors and it is mentioned that real-world attractors are most likely to be singleton attractors, rather than cyclic attractors.

Therefore, it is meaningful to identify singleton attractors. Of course, these can be done by examining all possible states of a Boolean network. However, it would be too time consuming even for small n, since 2^n states have to be examined. Of course, if we want to find any one (not necessarily singleton) attractor, we may find it by following the trajectory to the attractor beginning from a randomly selected state. If the basin of attraction is large, the possibility to find the corresponding attractor would be high. However, it is not guaranteed that a singleton attractor can be found. In order to find a singleton attractor, a lot of trajectories may be examined. Indeed, Akutsu et al. proved in 1998 that finding a singleton attractor is NP-hard [24]. Independently, Milano and Roli showed in 2000 that the satisfiability problem can be transformed into the problem of finding a singleton attractor [25], which provides a proof of NP-hardness of the singleton attractor problem. Thus, it is not plausible that the singleton attractor problem can be solved efficiently (i.e., polynomial time) in all cases. However, it may be possible to develop algorithms that are fast in practice and/or in the average case. Therefore, this paper studies algorithms for identifying singleton attractors that are fast in many practical cases and have concrete theoretical backgrounds.

Some studies have been done on fast identification of singleton attractors. Akutsu et al. proposed an algorithm for finding singleton attractors based on a feedback vertex set [24]. Devloo et al. proposed algorithms for finding steady states of various biological networks using constraint programming [20], which can also be applied to identification of singleton attractors in Boolean networks. In particular, the algorithms proposed by Devloo et al. are efficient in practice. However, there are no theoretical results on the efficiency of their algorithms. Thus, we aim at developing algorithms that are fast in practice and have a theoretical guarantee on their efficiency (more precisely, the average case time complexity).

In this paper, we propose several algorithms for identifying all singleton attractors. We first present a basic recursive algorithm. In this algorithm, a partial solution is extended one by one according to a given gene ordering that leads to a complete solution. If it is found that a partial solution cannot be extended to a complete solution, the next partial solution is examined. This algorithm is quite similar to the backtracking method employed in [20]. The important difference of this paper from [20] is that we perform some theoretical analysis of the average case time complexity. For example, we show that the basic recursive algorithm works in $O(1.23^n)$ time in the average case under the condition that Boolean networks with maximum indegree 2 are given uniformly at random. It should be noted that $O(1.23^n)$ is much smaller than $O(2^n)$, though it is not polynomial.

Next, we develop improved algorithms using the outdegree-based ordering and the breadth-first search (BFS) based ordering. For these algorithms, we perform theoretical analysis of the average case time complexity, which shows that these are better than the basic recursive algorithm. Moreover, we examine the algorithm based on feedback vertex sets (FVS) and its combination with the outdegree-based ordering, where the idea of use of FVS was previously proposed in our previous work [24]. We also perform computational experiments using these algorithms, which show that the FVS-based algorithm with the outdegree-based gene ordering is the most efficient in practice among these algorithms. Then, we extend the gene-ordering-based algorithms for finding cyclic attractors with short periods along with theoretical analysis and computational experiments. Though we do not have strong evidence that small attractors are more important than those with long periods, it seems that cell cycles correspond to small attractors and large attractors are not so common (with the exception of circadian rhythms) in real biological networks. As a minimum, these extensions show that application of the proposed techniques is not limited to the singleton attractor problem.

As mentioned before, NP-hardness results on finding a singleton attractor (or the smallest attractor) were already presented in [24, 25]. However, both papers appeared as conference papers, the detailed proof is not given in [24], and the transformation given in [25] is a bit complicated. Therefore, we describe a simple and complete proof. We believe that it is worthy to include a simple and complete proof in this paper.

Finally, we conclude with future work.

2. ANALYSIS OF ALGORITHMS USING GENE ORDERING FOR FINDING SINGLETON ATTRACTORS

In this section, we present algorithms using gene ordering for identification of singleton attractors along with theoretical analysis of the average case time complexity. Experimental results will be given later along with those of FVS-based

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	0	1	1
0	0	1	1	0	1
0	1	0	1	1	0
0	1	1	0	1	1
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

methods. Before presenting the algorithms, we briefly review the Boolean network model.

2.1. Boolean network and attractor

A Boolean network $G(V, F)$ consists of a set of n nodes (vertices) V and n Boolean functions F, where

$$V = \{v_1, v_2, \ldots, v_n\},$$
$$F = \{f_1, f_2, \ldots, f_n\}. \quad (1)$$

In general, V and F correspond to a set of genes and a set of gene regulatory rules, respectively. Let $v_i(t)$ represent the state of v_i at time t. The overall expression level of all the genes in the network at time step t is given by the following vector:

$$v(t) = [v_1(t), v_2(t), \ldots, v_n(t)]. \quad (2)$$

This vector is referred to as the *Gene Activity Profile* (GAP) of the network at time t, where $v_i(t) = 0$ means that the ith gene is not expressed and $v_i(t) = 1$ means that it is expressed. Since $v(t)$ ranges from $[0, 0, \ldots, 0]$ (all entries are 0) to $[1, 1, \ldots, 1]$ (all entries are 1), there are 2^n possible states. The regulatory rules among the genes are given as follow:

$$v_i(t+1) = f_i(v_{i_1}(t), v_{i_2}(t), \ldots, v_{i_{k_i}}(t)), \quad i = 1, 2, \ldots, n. \quad (3)$$

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 gene v_i. The maximum indegree of a Boolean network is defined as

$$K = \max_i \{k_i\}. \quad (4)$$

The number of genes that are directly affected by gene v_i is called the *outdegree* of gene v_i. The states of all genes are updated synchronously according to the corresponding Boolean functions.

A consecutive sequence of GAPs $v(t), v(t+1), \ldots, v(t+p)$ is called an *attractor* with period p if $v(t) = v(t+p)$. An attractor with period 1 is called a *singleton attractor* and an attractor with period > 1 is called a *cyclic attractor*.

Table 1 gives an example of a truth table of a Boolean network. Each gene will update its state according to the states of some other genes in the previous step. The state transitions of this Boolean network can be seen in Figure 1. The system will eventually evolve into two attractors. One attractor is $[0, 1, 1]$, which is a singleton attractor, and the other one is

$$[1,0,1] \longrightarrow [1,0,0] \longrightarrow [0,1,0] \longrightarrow [1,1,0] \longrightarrow [1,0,1], \quad (5)$$

which is a cyclic attractor with period 4.

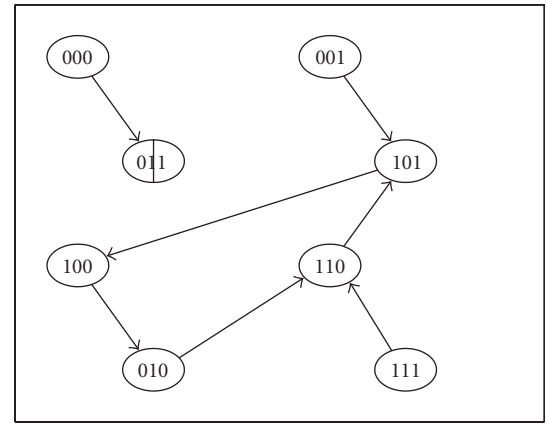

Figure 1: State transitions of the Boolean network shown in Table 1.

Input: a Boolean network $G(V, F)$
Output: all the singleton attractors
  Initialize $m := 1$;
  Procedure *IdentSingletonAttractor*(v, m)
    if $m = n + 1$ **then** Output $v_1(t), v_2(t), \ldots, v_n(t)$, **return**;
    for $b = 0$ **to** 1 **do** $v_m(t) := b$;
      if it is found that $v_j(t+1) \neq v_j(t)$ for some $j \leq m$ **then**
        continue;
      else *IdentSingletonAttractor*$(v, m+1)$;
    return.

Algorithm 1

2.2. Basic recursive algorithm

The number of singleton attractors in a Boolean network depends on the regulatory rules of the network. If the regulatory rules are given as $v_i(t+1) = v_i(t)$ for all i, the number of singleton attractors is 2^n. Thus, it would take $O(2^n)$ time in the worst case if we want to identify all the singleton attractors. On the other hand, it is known that the average number of singleton attractors is 1 regardless of the number of genes n and the maximum indegree K [21]. Therefore, it is useful to develop algorithms for identifying all singleton attractors without examining all 2^n states (in the average case).

For that purpose, we propose a very simple algorithm, which is referred to as the *basic recursive algorithm* in this paper. In the algorithm, a partial GAP (i.e., profile with m ($< n$) genes) is extended one by one towards a complete GAP (i.e.,

singleton attractor), 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. The pseudocode of the algorithm is given as shown in Algorithm 1.

The algorithm extends a partial GAP by one gene at a time. At the mth recursive step, the states of the first $m-1$ genes are determined. Then, the algorithm extends the partial GAP by adding $v_m(t) = 0$. If $v_j(t+1) = v_j(t)$ holds or the value of $v_j(t+1)$ is not determined for all $j = 1, \ldots, m$, the algorithm proceeds to the next recursive step. That is, if there is a possibility that the current partial GAP can be extended to a singleton attractor, it goes to the next recursive step. Otherwise, it extends the partial GAP by adding $v_m(t) = 1$ and executes a similar procedure. After examining $v_m(t) = 0$ and $v_m(t) = 1$, the algorithm returns to the previous recursive step. Since the number of singleton attractors is small in most cases, it is expected that the algorithm does not examine many partial GAPs with large m. The average case time complexity is estimated as follows.

Suppose that Boolean networks with maximum indegree K are given uniformly at random. Then the average case time complexity of the algorithm for $K = 1$ to $K = 10$ is given in the first row of Table 2.

Theoretical analysis

Assume that we have tested the first m out of n genes, where $m \geq K$. For all $i \leq m$, $v_i(t) \neq v_i(t+1)$ holds with probability

$$P(v_i(t) \neq v_i(t+1)) = 0.5 \cdot \left(\frac{{}_mC_{k_i}}{{}_nC_{k_i}}\right) \approx 0.5 \cdot \left(\frac{m}{n}\right)^{k_i} \geq 0.5 \cdot \left(\frac{m}{n}\right)^{K}. \quad (6)$$

If $v_i(t) \neq v_i(t+1)$ does not hold, the algorithm can continue. Therefore, the probability that the algorithm examines the $(m+1)$th gene is not more than

$$\left[1 - P(v_i(t) \neq v_i(t+1))\right]^m = \left[1 - 0.5 \cdot \left(\frac{m}{n}\right)^{K}\right]^m. \quad (7)$$

Thus, the number of recursive calls executed for the first m genes is at most

$$f(m) = 2^m \cdot \left[1 - 0.5 \cdot \left(\frac{m}{n}\right)^{K}\right]^m. \quad (8)$$

Let $s = m/n$, and $f(s) = [2^s \cdot (1 - 0.5 \cdot s^K)^s]^n = [(2 - s^K)^s]^n$. The average case time complexity is estimated by the maximum value of $f(s)$. Though an additional $O(nm)$ factor is required, it can be ignored since $O(n^2 a^n) \ll O((a+\epsilon)^n)$ holds for any $a > 1$ and $\epsilon > 0$.

Since the time complexity should be a function with respect to n, we only need to compute the maximum value of the function $g(s) = (2 - s^K)^s$. With simple numerical calculations, we can get its maximum value for fixed K. Then, the average case time complexity of the algorithm can be estimated as $O((\max(g))^n)$. We list the time complexity from $K = 1$ to 10 in the first row of Table 2. As K gets larger, the complexity increases.

2.3. Outdegree-based ordering algorithm

In the basic recursive algorithm, the original ordering of genes was used. If we sort the genes according to their outdegree (genes are ordered from larger outdegree to smaller outdegree), it is expected that values of $v_j(t+1)$ for a larger number of genes are determined at each recursive step than those determined for the basic recursive algorithm, and thus a lower number of partial GAPs are examined. This intuition is justified by the following theoretical analysis.

Suppose that Boolean networks with maximum indegree K are given uniformly at random. After reordering all genes according to their outdegrees from largest to smallest, the average case time complexity of the algorithm for $K = 1$ to $K = 10$ is given in the second row of Table 2.

Theoretical analysis

We assume (without loss of generality) w.l.o.g. that the indegrees of all genes are K. If the input genes for any gene are randomly selected from all the genes, the outdegree of genes follows the Poisson distribution with mean approximately λ. In this case, $\lambda = K$ holds since the total indegree must be equal to the total outdegree. Thus, λ and K are confused in the following. The probability that a gene has outdegree k is

$$P(k) = \frac{\lambda^k \exp(-\lambda)}{k!}. \quad (9)$$

We reorder the genes according to their outdegrees from largest to smallest. Assume that the first m genes have been tested and gene m is the uth gene among the genes with outdegree l. Then

$$m - u = n \cdot \sum_{k=l+1}^{\infty} \frac{\lambda^k \exp(-\lambda)}{k!} \quad (10)$$

and therefore

$$n - m = n \cdot \sum_{k=0}^{l} \frac{\lambda^k \exp(-\lambda)}{k!} - u. \quad (11)$$

The total outdegree of these $n - m$ genes is

$$n \cdot \sum_{k=0}^{l} \frac{\lambda^k \exp(-\lambda)}{k!} \cdot k - u \cdot l. \quad (12)$$

The total outdegree for the first m genes is

$$\begin{aligned} &\lambda n - \left(n \cdot \sum_{k=0}^{l} \frac{\lambda^k \exp(-\lambda)}{k!} \cdot k - u \cdot l\right) \\ &= \lambda n - \lambda n \cdot \sum_{k=0}^{l-1} \frac{\lambda^k \exp(-\lambda)}{k!} + u \cdot l \\ &= \lambda n - \lambda\left(n - (m-u) - n \cdot \frac{\lambda^l \exp(-\lambda)}{l!}\right) + u \cdot l \\ &= \lambda m + \lambda n \cdot \frac{\lambda^l \exp(-\lambda)}{l!} + u(l - \lambda). \end{aligned} \quad (13)$$

Thus, for $i \le m$, we have

$$\begin{aligned} &P(v_i(t) \neq v_i(t+1)) \\ &\quad = 0.5 \cdot \left[\frac{\lambda m + \lambda n \cdot (\lambda^l \exp(-\lambda)/l!) + u(l-\lambda)}{\lambda n}\right]^{\lambda} \\ &\quad = 0.5 \cdot \left[\frac{m}{n} + \frac{\lambda^l \exp(-\lambda)}{l!} + \frac{(l-\lambda)u}{\lambda n}\right]^{\lambda}. \end{aligned} \tag{14}$$

The number of recursive calls executed for the first m genes is

$$f(m) = 2^m \cdot \left[1 - 0.5 \cdot \left(\frac{m}{n} + \frac{\lambda^l \exp(-\lambda)}{l!} + \frac{(l-\lambda)u}{\lambda n}\right)^{\lambda}\right]^m. \tag{15}$$

Letting $s = m/n$, $f(m)$ can be rewritten as

$$\begin{aligned} f(m) &= \left[2^s \cdot \left(1 - 0.5 \cdot \left(s + \frac{\lambda^l \exp(-\lambda)}{l!} + \frac{(l-\lambda)u}{\lambda n}\right)^{\lambda}\right)^s\right]^n \\ &= \left[\left(2 - \left(s + \frac{\lambda^l \exp(-\lambda)}{l!} + \frac{(l-\lambda)u}{\lambda n}\right)^{\lambda}\right)^s\right]^n. \end{aligned} \tag{16}$$

As in Section 2.2, we estimate the maximum value of $g(s)$ where it is defined here as $g(s) = [2 - (s + \lambda^l \exp(-\lambda)/l! + (l-\lambda)u/\lambda n)^{\lambda}]^s$. We also must consider the relationship between l and λ.

(1) If $l > \lambda$,

$$g(s) \le \left[2 - \left(s + \frac{\lambda^l \exp(-\lambda)}{l!}\right)^{\lambda}\right]^s = g_1(s). \tag{17}$$

Since $\lambda^l \exp(-\lambda)/l!$ tends to zero if l is large, we only need to examine several small values of l. The upper bound of $g(s)$ can be obtained by computing the maximum value of $g_1(s)$ with some numerical methods. However, we should be careful so that

$$P(k \ge l+1) \le s \le P(k \ge l) \tag{18}$$

holds. That is, it should be guaranteed that the maximum value obtained is for the gene with outdegree l.

(2) If $l = \lambda$,

$$g(s) = \left[2 - \left(s + \frac{\lambda^l \exp(-\lambda)}{l!}\right)^{\lambda}\right]^s. \tag{19}$$

Similar to above, we can get an upper bound for $g(s)$.

(3) If $l < \lambda$,

$$g(s) = \left[2 - \left(s + \frac{\lambda^l \exp(-\lambda)}{l!} + \frac{(l-\lambda)u}{\lambda n}\right)^{\lambda}\right]^s. \tag{20}$$

Since gene m is the uth gene among the genes with outdegree l,

$$u \le n \cdot \frac{\lambda^l \exp(-\lambda)}{l!}. \tag{21}$$

Thus,

$$\begin{aligned} g(s) &\le \left[2 - \left(s + \frac{\lambda^l \exp(-\lambda)}{l!} + \frac{(l-\lambda)}{\lambda n} \cdot n \cdot \frac{\lambda^l \exp(-\lambda)}{l!}\right)^{\lambda}\right]^s \\ &= \left[2 - \left(s + \frac{\lambda^l \exp(-\lambda)}{l!} + (l-\lambda) \cdot \frac{\lambda^{l-1} \exp(-\lambda)}{l!}\right)^{\lambda}\right]^s. \end{aligned} \tag{22}$$

There are only a few values that are less than λ. Using a method similar to the one above, we can get an upper bound for $g(s)$.

It should be noted that l must belong to exactly one of these three cases when $g(s)$ reaches its maximum value. Summarizing the three different cases above, we can get an approximation of the average case time complexity of the algorithm. The second row of Table 2 shows the time complexity of the algorithm for $K = 1$ to $K = 10$. As in Section 2.2, the complexity increases as K increases.

We remark that the difference between this improved algorithm and the basic recursive algorithm lies only in that we need to sort all the genes according to their outdegrees from largest to smallest before executing the main procedure of the basic recursive algorithm.

2.4. Breadth-first search-based ordering algorithm

Breadth-first search is a general technique for traversing a graph. It visits all the nodes and edges of a graph in a manner that all the nodes at depth (distance from the root node) d are visited before visiting nodes at depth $d + 1$. For example, suppose that node a has outgoing edges to nodes b and c, b has outgoing edges to nodes d and e, and c has outgoing edges to nodes f and g, where other edges (e.g., an edge from d to f) can exist. In this case, nodes are visited in the order of a, b, c, d, e, f. In this way, all of the nodes are totally ordered according to the visiting order. The algorithm for implementing BFS can be found in many text books. The computation time for BFS on a graph with n nodes and m edges is $O(n+m)$. If we use this BFS-based ordering, as in the case of outdegree-based ordering, it is expected that values of $v_j(t+1)$ for a larger number of genes are determined at each recursive step, and thus, lower numbers of partial GAPs are examined. We can estimate the average case time complexity as follows.

Suppose that Boolean networks with maximum indegree K are given uniformly at random. After reordering all genes according to the BFS-ordering, the average case time complexity of the algorithm for $K = 1$ to $K = 10$ is given in the third row of Table 2.

Theoretical analysis

As in Section 2.3, we assume w.l.o.g. that all n genes have the same indegree K. Suppose that we have tested m genes. Since the input genes of the ith gene must be among the first $K \cdot i + 1$ genes, whether $v_i(t+1) = v_i(t)$ or not can be determined before visiting the $(K \cdot i + 2)$th gene. According to

Table 2: Theoretical time complexities of basic, outdegree-based, and BFS-based algorithms.

K	1	2	3	4	5	6	7	8	9	10
Basic	1.23^n	1.35^n	1.43^n	1.49^n	1.53^n	1.57^n	1.60^n	1.62^n	1.65^n	1.67^n
Outdegree-based	1.09^n	1.19^n	1.27^n	1.34^n	1.41^n	1.45^n	1.48^n	1.51^n	1.56^n	1.57^n
BFS-based	$\approx O(n)$	1.16^n	1.27^n	1.35^n	1.41^n	1.45^n	1.50^n	1.53^n	1.56^n	1.58^n

the determination pattern of states of m genes, we consider 3 cases.

(1) The states of the first $\llcorner (m-1)/K \lrcorner$ genes are determined and they must satisfy $v_i(t+1) = v_i(t)$, where $\llcorner a \lrcorner$ denotes the standard floor function. Then, we have

$$P(v_i(t) = v_{i+1}(t)) = 0.5, \quad i \le \llcorner \frac{m-1}{K} \lrcorner. \tag{23}$$

(2) For any gene i between the $\llcorner m/K \lrcorner$th gene and the $\llcorner (n-1)/K \lrcorner$th gene, whether $v_i(t+1)$ is equal to $v_i(t)$ can be determined before examining the $(m + j \cdot K)$th gene, where $j = 1, 2, \ldots, \llcorner (n-m)/K \lrcorner$. Then, we have

$$P(v_i(t) \neq v_{i+1}(t)) = 0.5 \cdot \left(\frac{m}{m + j \cdot K}\right)^K, \quad \llcorner \frac{m}{K} \lrcorner \le i \le \llcorner \frac{n-1}{K} \lrcorner. \tag{24}$$

The algorithm can continue for any gene i with probability

$$1 - P(v_i(t) \neq v_{i+1}(t)) = 1 - 0.5 \cdot \left(\frac{m}{m + j \cdot K}\right)^K, \quad \llcorner \frac{m}{K} \lrcorner \le i \le \llcorner \frac{n-1}{K} \lrcorner. \tag{25}$$

(3) From the $\llcorner n/K \lrcorner$th gene to the mth gene, the input genes to them can be any gene; thus

$$P(v_i(t) \neq v_{i+1}(t)) = 0.5 \cdot \left(\frac{m}{n}\right)^K, \quad \llcorner \frac{n-1}{K} \lrcorner \le i \le m. \tag{26}$$

Here, the algorithm can continue for each gene with probability

$$1 - P(v_i(t) \neq v_{i+1}(t)) = 1 - 0.5 \cdot \left(\frac{m}{n}\right)^K, \quad \llcorner \frac{n-1}{K} \lrcorner \le i \le m. \tag{27}$$

The probability that the algorithm can be executed for all m genes is

$$\begin{aligned} &\left(\prod_{i=1}^{\llcorner (m-1)/K \lrcorner} P(v_i(t) = v_{i+1}(t))\right) \\ &\quad \cdot \left(\prod_{i=\llcorner (m-1)/K \lrcorner}^{\llcorner (n-1)/K \lrcorner} (1 - P(v_i(t) \neq v_{i+1}(t)))\right) \\ &\quad \cdot \left(\prod_{i=\llcorner (n-1)/K \lrcorner}^{m} (1 - P(v_i(t) \neq v_{i+1}(t)))\right) \\ &= 0.5^{\llcorner (m-1)/K \lrcorner} \cdot \left(\prod_{i=\llcorner (m-1)/K \lrcorner}^{\llcorner (n-1)/K \lrcorner} \left[1 - 0.5 \cdot \left(\frac{m}{m + i \cdot K}\right)^K\right] \right. \\ &\qquad \left. \cdot \left[1 - 0.5 \cdot \left(\frac{m}{n}\right)^K\right]^{m - \llcorner (n-1)/K \lrcorner}\right). \end{aligned} \tag{28}$$

Then, the total number of recursive calls is

$$\begin{aligned} f(m) &= 2^m \cdot 0.5^{\llcorner (m-1)/K \lrcorner} \\ &\quad \cdot \left(\prod_{i=\llcorner (m-1)/K \lrcorner}^{\llcorner (n-1)/K \lrcorner} \left[1 - 0.5 \cdot \left(\frac{m}{m + i \cdot K}\right)^K\right] \right. \\ &\qquad \left. \cdot \left[1 - 0.5 \cdot \left(\frac{m}{n}\right)^K\right]^{m - \llcorner (n-1)/K \lrcorner}\right) \\ &\le 2^m \cdot 0.5^{\llcorner (m-1)/K \lrcorner} \cdot \left[1 - 0.5 \cdot \left(\frac{m}{n}\right)^K\right]^{m - \llcorner (m-1)/K \lrcorner} \\ &= \left[2 - \left(\frac{m}{n}\right)^K\right]^{m - \llcorner (m-1)/K \lrcorner} \\ &= \left[2 - \left(\frac{m}{n}\right)^K\right]^{[(m - \llcorner (m-1)/K \lrcorner)/n] \cdot n} \\ &\approx \left[2 - \left(\frac{m}{n}\right)^K\right]^{(m/n)(1-1/K) \cdot n}. \end{aligned} \tag{29}$$

Let $s = m/n$ and $g(s) = (2 - s^K)^{s(1-1/K)}$. Using numerical methods, we can get the maximum value of g. From $K = 1$ to $K = 10$, the upper bound of the average case time complexity of the algorithm is in the third row of Table 2.

It is to be noted that in the estimation of the upper bound of $f(m)$, we overestimated the probability that genes belong to the second case, and thus the upper bound obtained here is not tight. More accurate time complexities can be estimated from the results of computational experiments.

3. FINDING SINGLETON ATTRACTORS USING FEEDBACK VERTEX SET

In this section, we present algorithms based on the feedback vertex set and the results of computational experiments on all of our proposed algorithms for identification of singleton attractors. The algorithms in this section are based on a simple and interesting property on acyclic Boolean networks although they can be applied to general Boolean networks with cycles. Though an algorithm based on the feedback vertex set was already proposed in our previous work [24], some improvements (ordering based on connected components and ordering based on outdegree) are achieved in this section.

3.1. Acyclic network

As to be shown in Section 5, the problem of finding a singleton attractor in a Boolean network is NP-hard. However, we have a positive result for acyclic networks as follows.

Proposition 1. *If the network is acyclic, there exists a unique singleton attractor. Moreover, the unique attractor can be computed in polynomial time.*

Proof. In an acyclic network, there exists at least one node without incoming edges. Such nodes should have fixed Boolean values. The values of the other nodes are uniquely determined from these nodes by the nth time step in polynomial time. Since the state of any node does not change after the nth step, there exists only one singleton attractor. □

As shown below, this property is also useful for identifying singleton attractors in cyclic networks.

3.2. Algorithm

In the basic recursive algorithm, we must consider truth assignments to all the nodes in the network. On the other hand, Proposition 1 indicates that if the network is acyclic, the truth values of all nodes are uniquely determined from the values of the nodes with no incoming edges. Thus, it is enough to examine truth assignments only to the nodes with no incoming edges, if we can decompose the network into acyclic graphs. Such a set of nodes is called a *feedback vertex set* (FVS). The problem of finding a minimum feedback vertex set is known to be NP-hard [26]. Some algorithms which approximate the minimum feedback vertex set have been developed [27]. However, such algorithms are usually complicated. Thus, we use a simple greedy algorithm (shown in Algorithm 2) for finding a (not necessarily minimum) feedback vertex set, where a similar algorithm was already presented in [24]. In our proposed algorithm, nodes in FVS are ordered according to the connected components of the original network in order to reduce the number of iterations. In other words, nodes in the same connected component are ordered sequentially.

Then, we modify the procedure *IdentSingletonAttractor*(v, m) for FVS as shown in Algorithm 3.

Input: a Boolean network $G(V, F)$
Output: an ordered feedback vertex set $\mathcal{F} = \{v_1^{(\mathrm{FVS})}, \ldots, v_M^{(\mathrm{FVS})}\}$
  Procedure *FindFeedbackVertexSet*
    let $\mathcal{F} := \emptyset$, $M := 1$;
    let C:= (all the connected components of G);
    for each connected component $C' \in C$ **do**
      let V':= (a set of vertices in C');
      while $V' \neq \emptyset$ **do**
        let $v_M^{(\mathrm{FVS})}$:=(a vertex selected randomly from V');
        remove $v_M^{(\mathrm{FVS})}$ and vertices whose truth values can be fixed only from $\mathcal{F}$ in V';
        increment M.

Algorithm 2

Input: a Boolean network $G(V, F)$ and an ordered feedback vertex set $\mathcal{F} = \{v_1^{(\mathrm{FVS})}, \ldots, v_M^{(\mathrm{FVS})}\}$
Output: all the singleton attractors
  Initialize $m := 1$;
  Procedure *IdentSingletonAttractorWithFVS*(v, m)
    if $m = M + 1$ **then** Output $v_1(t), v_2(t), \ldots, v_n(t)$, **return**;
    for $b = 0$ **to** 1 **do** $v_m^{(\mathrm{FVS})}(t) := b$;
      propagate truth values of $\{v_1^{(\mathrm{FVS})}(t), \ldots, v_m^{(\mathrm{FVS})}(t)\}$ to all possible $v(t)$ except $\mathcal{F}$;
      compute $\{v_1^{(\mathrm{FVS})}(t+1), \ldots, v_m^{(\mathrm{FVS})}(t+1)\}$ from $v(t)$;
      if it is found that $v_j^{(\mathrm{FVS})}(t+1) \neq v_j^{(\mathrm{FVS})}(t)$ for some $j \leq m$ **then**
        continue;
      else *IdentSingletonAttractorWithFVS*$(v, m+1)$;
    return.

Algorithm 3

Furthermore, we can combine the outdegree-based ordering with FVS. In *FindFeedbackVertexSet*, we select a node randomly from a connected component. When combined with the outdegree-based ordering, we can instead select the node with the maximum outdegree in a connected component.

3.3. Computational experiments

In this section, we evaluate the proposed algorithms by performing a number of computational experiments on both random networks and scale-free networks [28].

3.3.1. Experiments on random networks

For each K ($K = 1, \ldots, 10$) and each n ($n = 1, \ldots, 20$), we randomly generated 10 000 Boolean networks with maximum indegree K and took the average values. All of these computational experiments were done on a PC with Opteron

Table 3: Empirical time complexities of basic, outdegree, BFS, feedback vertex set, and FVS + outdegree algorithms.

K	1	2	3	4	5	6	7	8	9	10
Basic	1.27^n	1.39^n	1.46^n	1.53^n	1.57^n	1.60^n	1.63^n	1.67^n	1.69^n	1.70^n
Outdegree	1.14^n	1.23^n	1.30^n	1.37^n	1.42^n	1.47^n	1.51^n	1.54^n	1.56^n	1.59^n
BFS	1.09^n	1.16^n	1.24^n	1.31^n	1.37^n	1.42^n	1.45^n	1.49^n	1.52^n	1.53^n
Feedback	1.10^n	1.28^n	1.39^n	1.47^n	1.53^n	1.56^n	1.60^n	1.64^n	1.66^n	1.68^n
FVS + Outdegree	1.05^n	1.13^n	1.21^n	1.29^n	1.35^n	1.41^n	1.46^n	1.49^n	1.52^n	1.55^n

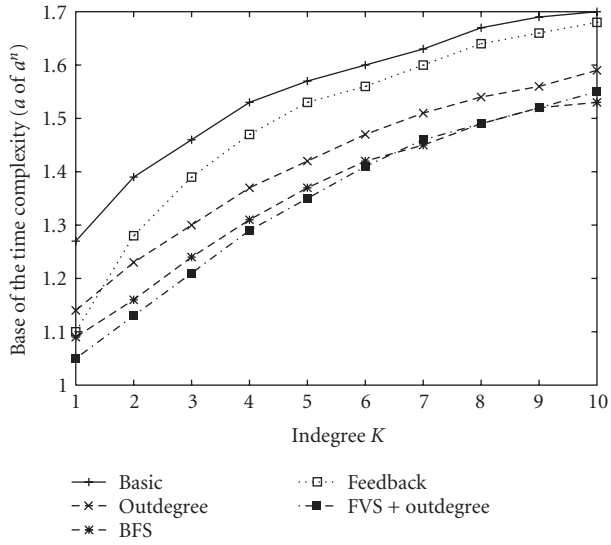

Figure 2: Base of the empirical time complexity (a^n's a value) of the proposed algorithms for finding singleton attractors.

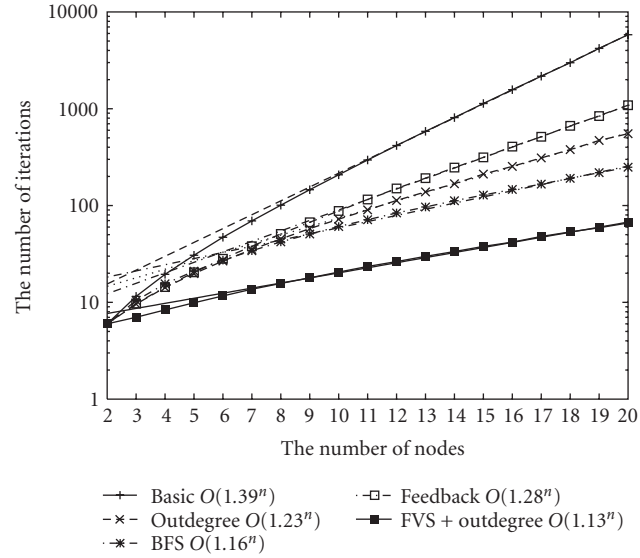

Figure 3: Number of iterations done by the proposed algorithms for $K = 2$.

2.4 GHz CPUs and 4 GB RAM running under the Linux (version 2.6.9) operating system, where the gcc compiler (version 3.4.5) was used with optimization option -O3.

Table 3 shows the empirical time complexity of each proposed method for each K. We used a tool for GNUPLOT to fit the function $b \cdot a^n$ to the experimental results. The tool uses the nonlinear least-squares (NLLS) Marquardt-Levenberg algorithm. Figure 2 is a graphical representation of the result of Table 3. It is seen that the FVS + Outdegree method is the fastest in most cases.

Figure 3 is an example to show the average number of iterations with respect to the number of genes for $K = 2$. Figure 4 shows the average computation time with respect to the number of genes when $K = 2$, where similar results were obtained for other values of K.

The time complexities estimated from the results of computational experiments are a little different from those obtained by theoretical analysis. However, this is reasonable since, in our theoretical analysis, we assumed that the number of genes is very large, we made some approximations, and there were also small numerical errors in computing the maximum values of $g(s)$.

3.3.2. Experiments on scale-free networks

It is known that many real biological networks have the scale-free property (i.e., the degree distribution approximately follows a power-law) [28]. Furthermore, it is observed that in gene regulatory networks, the outdegree distribution follows a power-law and the indegree distribution follows a Poisson distribution [29]. Thus, we examined networks with scale free topology.

We generated scale-free networks with a power-law outdegree distribution ($\propto k^{-2}$) and a Poisson indegree distribution (with the average indegree 2) as follows. We first choose the number of outputs for each gene from a power-law distribution. That is, gene v_i has L_i outputs where all the L_i are drawn from a power-law distribution. Then, we choose the L_i outputs of each gene v_i randomly with uniform probability from n genes. Once each gene has been assigned with a set of outputs, the inputs of all genes are fully determined because v_j is an input of v_i if v_i is an output of v_j. Since L_i output genes are chosen randomly for each gene v_i, the indegree distribution should follow a Poisson distribution.

Figure 5 compares the outdegree-based algorithm, the BFS-based algorithm and the FVS + Outdegree algorithm for scale-free networks generated as above and for random networks with constant indegree 2, where the average CPU time

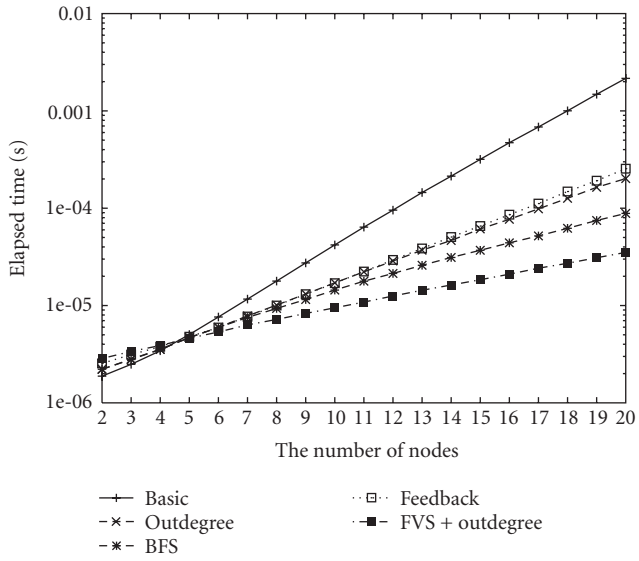

Figure 4: Elapsed time (in seconds) by the proposed algorithms for random networks with $K = 2$.

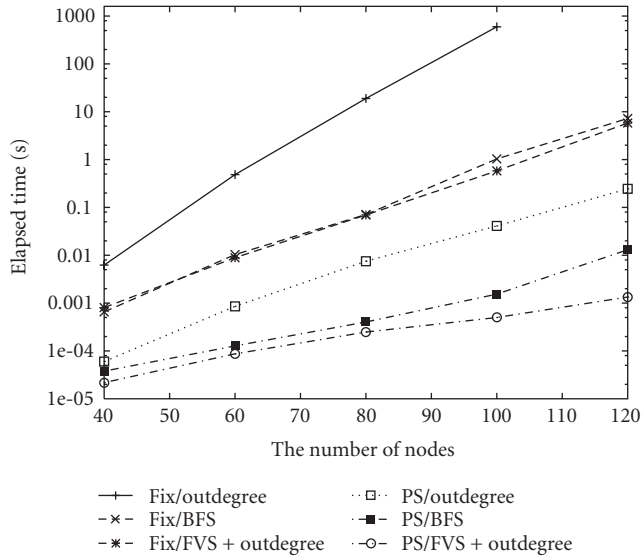

Figure 5: Elapsed time (in seconds) of some of the proposed algorithms for random networks with $K = 2$ (Fix) and scale-free networks (PS).

was taken over 100 networks for each case and a PC with Xeon 5160 3 GHz CPUs with 8 GB RAM was used. The result is interesting and we observed that all algorithms work much faster for scale-free networks than for random networks. This result is reasonable because scale-free networks have a much larger number of high degree nodes than random networks and thus heuristics based on the outdegree-based ordering or the BFS-based ordering should work efficiently. The average case time complexities estimated from this experimental result are as follows: $O(1.19^n)$ versus $O(1.09^n)$ for the outdegree-based algorithm, $O(1.12^n)$ versus $O(1.09^n)$ for the BFS-based algorithm, and $O(1.12^n)$ versus $O(1.05^n)$ for the FVS + Outdegree algorithm, where (random) versus (scale-free) is shown for each case. The average case complexities for random networks are better than those in Table 3 and are closer to the theoretical time complexities shown in Table 2. These results are reasonable because networks with much larger number of nodes were examined in this case.

It should be noted that Devloo et al. proposed constraint programming based methods for finding steady-states in some kinds of biological networks [20]. Their methods use a backtracking technique, which is very close to our proposed recursive algorithms, and may also be applied to Boolean networks. Their methods were applied to networks up to several thousand nodes with indegree = outdegree = 2. Since different types of networks were used, our proposed methods cannot be directly compared with their methods. Their methods include various heuristics and may be more useful in practice than our proposed methods. However, no theoretical analysis was performed on the computational complexity of their methods.

Input: a Boolean network $G(V, F)$ and a period p
Output: all of the small attractors with period p
  Initialize $m := 1$;
  Procedure *IdentSmallAttractor*(v, m)
    if $m = n + 1$ **then** Output $v_1(t), v_2(t), \ldots, v_n(t)$, **return**;
    for $b = 0$ **to** 1 **do** $v_m(t) := b$;
      for $p' = 0$ **to** $p-1$ **do** compute $v(t+p'+1)$ from $v(t+p')$;
        if it is found that $v_j(t+p) \neq v_j(t)$ for some $j \leq m$ **then**
          continue;
        else *IdentSmallAttractor*$(v, m + 1)$;
    return.

Algorithm 4

4. FINDING SMALL ATTRACTORS

In this section, we modify the gene-ordering-based algorithms presented in Section 2 to find cyclic attractors with short periods. We also perform a theoretical analysis and computational experiments.

4.1. Modifications of algorithms

The basic idea of our modifications is very simple. Instead of checking whether or not $v_i(t + 1) = v_i(t)$ holds, we check whether or not $v_i(t + p) = v_i(t)$ holds. The pseudocode of the modified basic recursive algorithm is given in Algorithm 4.

This procedure computes $v(t + p)$ from the truth assignments on the first m genes of $v(t)$. Values of some genes of $v(t + p)$ may not be determined because these genes may also depend on the last $(n - m)$ genes of $v(t)$. If either $v_j(t + p) = v_j(t)$ holds or the value of $v_j(t + p)$ is not determined for each $j = 1, \ldots, m$, the algorithm will continue to the next

recursive step. As in Section 2, we can combine this algorithm with the outdegree-based ordering and the BFS-based ordering.

In these algorithms, it is assumed that the period p is given in advance. However, the algorithms can be modified for identifying all cyclic attractors with period at most P. For that purpose, we simply need to execute the algorithms for each of $p = 1, 2, \ldots, P$. Though this method does not seem to be practical, its theoretical time complexity is still better than $O(2^n)$ for small P. Suppose that the average case time complexity for p is $O(T_p(n))$. Then, this simple method would take $O(\sum_{p=1}^{P} T_p(n)) \leq O(P \cdot T_P(n))$ time, which is still faster than $O(2^n)$ if $T_P(n) = o(2^n)$ and P is bounded by some polynomial of n.

4.2. Theoretical analysis

Before giving the experimental results, we perform a theoretical analysis on the modified basic recursive algorithm.

Suppose that Boolean networks with maximum indegree K are given uniformly at random. Then the average case time complexity of the modified basic recursive algorithm for period 1 to 5 and $K = 1$ to $K = 10$ is given in Table 4.

Theoretical analysis

Let the period of the attractor be p. We assume w.l.o.g. as before that the indegree of all genes is K. As in Section 2.2, we consider the first m genes among all n genes. Given the states of all m genes at time t, we need to know the states of all these genes at time $t+p$. The probability that $v_i(t) \neq v_i(t+p)$ holds for each $i \leq m$ is approximated by:

$$P(v_i(t) \neq v_i(t+p)) = 0.5 \cdot \left(\frac{m}{n}\right)^{K} \cdot \left(\frac{m}{n}\right)^{K^2} \cdots \left(\frac{m}{n}\right)^{K^p}, \tag{30}$$

where $(m/n)^K$ means that the K input genes to gene v_i at time $t+p-1$ are among the first m genes, $(m/n)^{K^2}$ means that at time $t+p-2$ the input genes to the K input genes to gene v_i are also in the first m genes, and so on.

Then, the probability that the algorithm examines some specific truth assignment on m genes is approximately given by

$$\begin{aligned}&[1 - P(v_i(t) \neq v_i(t+p))]^m \\ &\quad = \left[1 - 0.5 \cdot \left(\frac{m}{n}\right)^{K} \cdot \left(\frac{m}{n}\right)^{K^2} \cdots \left(\frac{m}{n}\right)^{K^p}\right]^m.\end{aligned} \tag{31}$$

Therefore, the number of total recursive calls executed for these m genes is

$$\begin{aligned}f(m) &= 2^m \cdot [1 - P(v_i(t) \neq v_i(t+p))]^m \\ &= 2^m \cdot \left[1 - 0.5 \cdot \left(\frac{m}{n}\right)^{K} \cdot \left(\frac{m}{n}\right)^{K^2} \cdots \left(\frac{m}{n}\right)^{K^p}\right]^m.\end{aligned} \tag{32}$$

As in Section 2.2, we can compute the maximum value of $f(m)$. The results are given in Table 4.

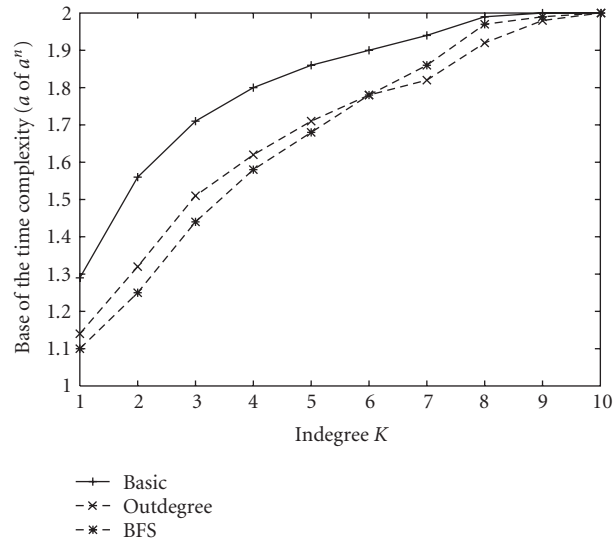

Figure 6: Base of the empirical time complexity (a^n's a value) of the proposed algorithms for finding cyclic attractors with period 2.

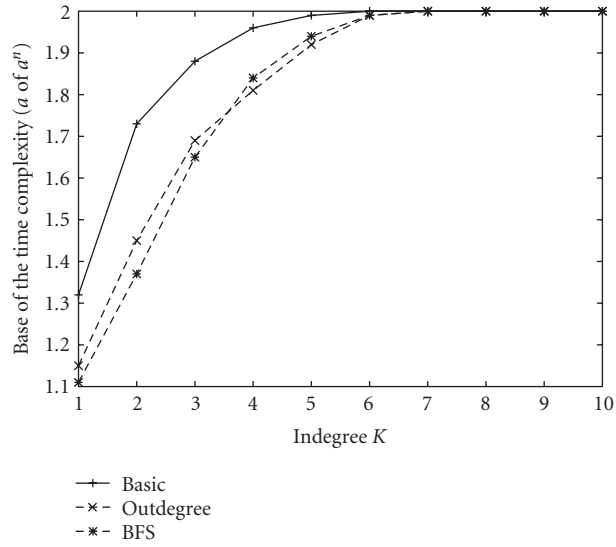

Figure 7: Base of the empirical time complexity (a^n's a value) of the proposed algorithms for finding cyclic attractors with period 3.

4.3. Computational experiments

Computational experiments were also performed to examine the time complexity of the algorithms for finding small attractors. The environment and parameters of the experiments were the same as in Section 3.3.1. Though FVS-based algorithms can also be modified for small attractors, they are not efficient for $p > 1$. Therefore, we only examined gene-ordering-based algorithms.

Figures 6 to 8 show the time complexity of the algorithms estimated from the results of computational experiments for $p = 2$ to $p = 4$ and for $K = 1$ to $K = 10$. When K is comparatively small, the outdegree-based ordering method is the

Table 4: Theoretical time complexities for the modified basic algorithm for finding small attractors with period p.

K	1	2	3	4	5	6	7	8	9	10
$p = 1$	1.23^n	1.35^n	1.43^n	1.49^n	1.53^n	1.57^n	1.60^n	1.62^n	1.65^n	1.67^n
$p = 2$	1.35^n	1.57^n	1.70^n	1.78^n	1.83^n	1.87^n	1.89^n	1.91^n	1.92^n	1.93^n
$p = 3$	1.43^n	1.72^n	1.86^n	1.92^n	1.95^n	1.97^n	1.97^n	1.98^n	1.99^n	1.99^n
$p = 4$	1.49^n	1.83^n	1.94^n	1.97^n	1.99^n	1.99^n	1.99^n	1.99^n	1.99^n	1.99^n
$p = 5$	1.53^n	1.90^n	1.97^n	1.99^n	1.99^n	1.99^n	1.99^n	1.99^n	1.99^n	1.99^n

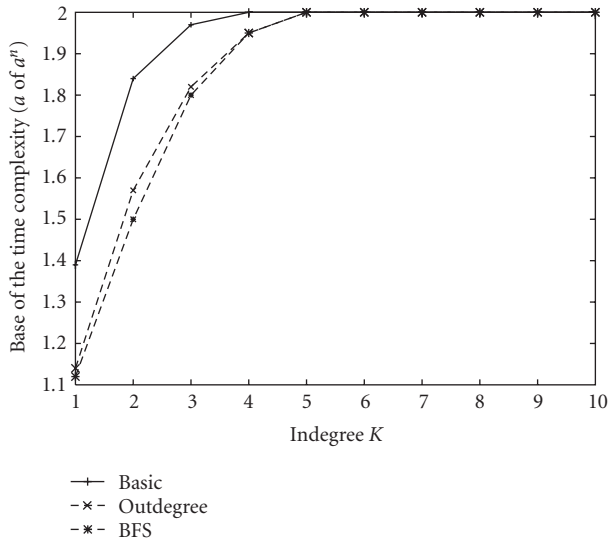

Figure 8: Base of the time complexity (a^n's a value) of the proposed algorithms for finding cyclic attractors with period 4.

most efficient. But when K increases, all the three methods perform the same, which is equivalent to the worst case in finding the attractors, that is $O(2^n)$. The results obtained from the numerical experiments for the modified basic recursive algorithm are consistent with the theoretical results presented in Section 4.2.

5. HARDNESS RESULT

As mentioned in Section 1, Akutsu et al. [24] and Milano and Roli [25] showed that finding a singleton attractor (or an attractor with the shortest period) is NP-hard. Those results justify our proposed algorithms which take exponential time in the worst case (and even in the average case). However, the proof is omitted in [24] and the proof in [25] is a bit complicated: Boolean functions assigned in the transformed Boolean network are much longer than those in the original satisfiability problem. Here we give a simpler and complete proof.

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

Proof. We show that deciding whether or not there exists a singleton attractor is NP-hard, from which the theorem follows since the singleton attractor is the attractor with the shortest period (if any such period exists).

We use a simple polynomial time reduction from 3SAT [26] to the singleton attractor problem.

Let $x_1, \ldots, x_N$ be Boolean variables (i.e., 0-1 variables). Let $c_1, \ldots, c_L$ be a set of clauses over $x_1, \ldots, x_N$, where each clause is a logical OR of at most three literals. It should be noted that a literal is a variable or its negation (logical NOT). Then, 3SAT is a problem of asking whether or not there exists an assignment of 0-1 values to $x_1, \ldots, x_N$ which satisfies all the clauses (i.e., the values of all clauses are 1).

From an instance of 3SAT, we construct an instance of the singleton attractor problem. We let the set of vertices (nodes) $V = \{v_1, \ldots, v_{N+L}\}$, where each v_i for $i = 1, \ldots, N$ corresponds to x_i and each v_{N+i} for $i = 1, \ldots, L$ corresponds to c_i. For each v_i such that $i \leq N$, we make the following assignment:

$$v_i(t+1) = v_i(t). \tag{33}$$

Suppose that $f_i(x_{i_1}, \ldots, x_{i_3})$ is a Boolean function assigned to c_i in 3SAT. Then, for each v_{N+i}, we assign the following function:

$$v_{N+i}(t+1) = f_i\left(v_{i_1}(t), v_{i_2}(t), v_{i_3}(t)\right) \vee \overline{v_{N+i}(t)}. \tag{34}$$

Figure 9 is an example of reduction from 3SAT to the singleton attractor problem.

Here, we show that 3SAT is satisfiable if and only if there exists a singleton attractor.

Suppose that there exists an assignment of Boolean values $b_1, \ldots, b_N$ to $x_1, \ldots, x_N$ which satisfies all clauses $c_1, \ldots, c_L$. Then, we let

$$v_i(0) = \begin{cases} b_i & \text{for } i = 1, \ldots, N, \\ 1 & \text{for } i = N+1, \ldots, N+L. \end{cases} \tag{35}$$

It is straight forward to see that $v(0) = (v_1(0), \ldots, v_{N+L}(0))$ is a singleton attractor (i.e., $v(0) = v(1)$).

Suppose that there exists a singleton attractor. Let $v(0) = (v_1(0), \ldots, v_{N+L}(0))$ be the state of the singleton attractor. Then, $v_{N+i}(0)$ must be 1 for all $i = 1, \ldots, L$. Otherwise (i.e., $v_{N+i}(0) = 0$), $v_{N+i}(1)$ would be 1 and it contradicts the assumption that $v(0)$ is a singleton attractor. Furthermore, $f_i(v_{i_1}(0), v_{i_2}(0), v_{i_3}(0)) = 1$ must hold. Otherwise, $v_{N+i}(1)$ would be 0 since the equations $v_{N+i}(0) = 1$ and $f_i(v_{i_1}(0), v_{i_2}(0), v_{i_3}(0)) = 0$ hold. This contradicts the assumption that $v(0)$ is a singleton attractor. Therefore, by assigning $v_i(0)$ to x_i for $i = 1, \ldots, N$, all the clauses are satisfied.

Since the reduction can trivially be done in polynomial time, we have the theorem. □

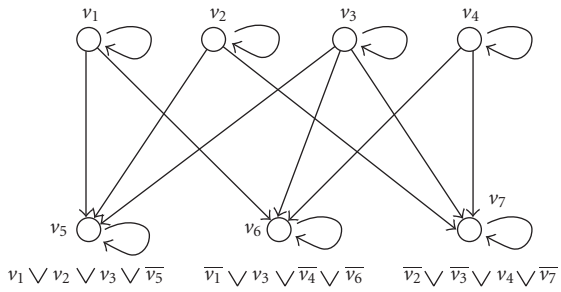

Figure 9: Example of a reduction from 3SAT to the singleton attractor problem. An instance of 3SAT $\{x_1 \vee x_2 \vee x_3, \overline{x_1} \vee x_3 \vee \overline{x_4}, \overline{x_2} \vee \overline{x_3} \vee x_4\}$ is transformed into this Boolean network.

Table 5: Comparison of time complexities for simulation of network behavior, identification of attractors, finding control strategies, and identification of networks. P means that the problem can be solved in polynomial time.

	Tree	Acyclic graph	General graph
Simulation of network	P	P	P
Identification of attractor	P	P	NP-hard
Finding control strategies	P	NP-hard	NP-hard
Identification of network	NP-hard	NP-hard	NP-hard
Identification of network (bounded indegree)	P	P	P

6. CONCLUSION

In this paper, we have presented fast algorithms for identifying singleton attractors and cyclic attractors with short periods. The proposed algorithms are much faster than the naive enumeration-based algorithm. However, the proposed algorithms cannot be applied to random networks with several hundreds or more genes. Moreover, it may not be faster than the constraint programming-based algorithms in [20]. However, the most important point of this work is that the average case time complexities of the ordering-based algorithms are analyzed and are shown to be better than $O(2^n)$. We hope that our work stimulates further development of faster algorithms and deeper theoretical analysis.

It is interesting that the results of computational experiments suggest that our proposed algorithms are much faster for scale-free networks than for random networks. However, we could not yet perform theoretical analysis for scale-free networks. Thus, theoretical analysis of the average case time complexity for scale-free networks (precisely, networks with a power-law outdegree distribution and a Poisson indegree distribution) is left as future work.

Although this paper focused on the Boolean network as a model of biological networks, the techniques proposed here may be useful for designing algorithms for finding steady states in other models and for theoretical analysis of such algorithms. For instance, Mochizuki performed theoretical analysis on the number of steady states in some continuous biological networks that are based on nonlinear differential equations [21]. However, the core part of the analysis is done in a combinatorial manner and is very close to that for Boolean networks. Thus, it may be possible to develop fast algorithms for finding steady states in such continuous network models. Application and extension of the proposed techniques to other types of biological networks are important future research topics.

Finally, it is interesting to compare the complexities of four problems for three classes of networks: simulation of network behavior (almost trivial), identification of attractors (this paper), identification of networks [30, 31], and finding control strategies [32] for trees, acyclic graphs, and general graphs. These four problems constitute a more complete picture of modeling genetic regulatory networks with a Boolean network. Simulation of a Boolean network is a trivial but important step to analyze the model. Attractors describe the long run behavior of the Boolean network system. Finding a control strategy is to consider how the system can be made to evolve desirably. Identification of genetic regulatory networks is the first step in obtaining the model from data. Table 5 shows complexities for various problems with several network structures. Although many works have been done for these problems, the computational complexity is still an important issue. It is also left as future work to study how to cope with high computational complexity (e.g., NP-hardness) of these problems.

ACKNOWLEDGMENTS

We thank anonymous reviewers for helpful comments. TA was partially supported by a Grant-in-Aid "Systems Genomics" from MEXT, Japan and by the Cell Array Project from NEDO, Japan. WKC was partially supported by Hung Hing Ying Physical Research Fund, HKU GRCC Grants nos. 10206647, 10206483, and 10206147. MKN was partially supported by RGC 7046/03P, 7035/04P, 7035/05P, and HKBU FRGs. S.-Q. Zhang and M. Hayashida contributed equally to this work.

REFERENCES

[1] J. E. Celis, M. Kruhøffer, I. Gromova, et al., "Gene expression profiling: monitoring transcription and translation products using DNA microarrays and proteomics," *FEBS Letters*, vol. 480, no. 1, pp. 2–16, 2000.

[2] T. R. Hughes, M. Mao, A. R. Jones, et al., "Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer," *Nature Biotechnology*, vol. 19, no. 4, pp. 342–347, 2001.

[3] R. J. Lipshutz, S. P. A. Fodor, T. R. Gingeras, and D. J. Lockhart, "High density synthetic oligonucleotide arrays," *Nature Genetics*, vol. 21, supplement 1, pp. 20–24, 1999.

[4] D. J. Lockhart and E. A. Winzeler, "Genomics, gene expression and DNA arrays," *Nature*, vol. 405, no. 6788, pp. 827–836, 2000.

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

[6] K. Glass and S. A. Kauffman, "The logical analysis of continuous, nonlinear biochemical control networks," *Journal of Theoretical Biology*, vol. 39, no. 1, pp. 103–129, 1973.

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

[8] S. A. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, no. 215, pp. 177–178, 1969.

[9] S. A. Kauffman, "The large scale structure and dynamics of genetic control circuits: an ensemble approach," *Journal of Theoretical Biology*, vol. 44, no. 1, pp. 167–190, 1974.

[10] 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.

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

[12] R. Somogyi and C. Sniegoski, "Modeling the complexity of genetic networks: understanding multigenic and pleiotropic regulation," *Complexity*, vol. 1, no. 6, pp. 45–63, 1996.

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

[14] 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.

[15] S. Huang, "Cell state dynamics and tumorigenesis in Boolean regulatory networks," *InterJournal Genetics*, MS: 416, http://www.interjournal.org/

[16] B. Drossel, "Number of attractors in random Boolean networks," *Physical Review E*, vol. 72, no. 1, Article ID 016110, 5 pages, 2005.

[17] 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.

[18] 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.

[19] J. E. S. Socolar and S. A. Kauffman, "Scaling in ordered and critical random Boolean networks," *Physical Review Letters*, vol. 90, no. 6, Article ID 068702, 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. 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.

[22] 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.

[23] X. Zhou, X. Wang, R. Pal, I. Ivanov, M. Bittner, and E. R. Dougherty, "A Bayesian connectivity-based approach to constructing probabilistic gene regulatory networks," *Bioinformatics*, vol. 20, no. 17, pp. 2918–2927, 2004.

[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, Springer, Bologna, Italy, September 1999.

[26] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, NY, USA, 1979.

[27] G. Even, J. Naor, B. Schieber, and M. Sudan, "Approximating minimum feedback sets and multicuts in directed graphs," *Algorithmica*, vol. 20, no. 2, pp. 151–174, 1998.

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

[29] 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.

[30] T. Akutsu, S. Miyano, and S. Kuhara, "Identification of genetic networks from a small number of gene expression patterns under the Boolean network model," in *Proceedings of the 4th Pacific Symposium on Biocomputing (PSB '99)*, vol. 4, pp. 17–28, Big Island of Hawaii, Hawaii, USA, January 1999.

[31] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano, "Identification of genetic networks by strategic gene disruptions and gene overexpressions under a Boolean model," *Theoretical Computer Science*, vol. 298, no. 1, pp. 235–251, 2003.

[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.