An Improved Algorithm for Detecting a Singleton Attractor in a Boolean Network Consisting of AND/OR Nodes

Takeyuki Tamura and Tatsuya Akutsu

Bioinformatics Center, Institute for Chemical Research, Kyoto University,
Gokasho, Uji, Kyoto, Japan, 611-0011.
{tamura, takutsu}@kuicr.kyoto-u.ac.jp
http://sunflower.kuicr.kyoto-u.ac.jp/member.html.en

Abstract. Detection of a singleton attractor, which is also called a fixed point, is known to be NP-hard even for AND/OR BNs (i.e., BNs consisting of AND/OR nodes), where the Boolean network (BN) is a mathematical model of genetic networks and singleton attractors correspond to steady states. In our recent paper, we developed an $O(1.787^n)$ time algorithm for detecting a singleton attractor of a given AND/OR BN where n is the number of nodes. In this paper, we present an $O(1.757^n)$ time algorithm with which we succeeded in improving the above algorithm.

1 Introduction

Studying biological networks from an algebraic perspective is becoming more important in various areas such as bioinformatics, computational biology and systems biology. To analyze them, various kinds of mathematical models of biological networks have been proposed. Among them, the *Boolean network* (BN, in short), which is a model of genetic networks, has received much attention [2, 3, 6, 12, 13]. It is a very simple model: each node (e.g., gene) takes either 0 (inactive) or 1 (active) and the states of nodes change synchronously according to regulation rules given as Boolean functions [9, 20].

Stable states are called *attractors* (or fixed points) in a BN. Since stable states play an important role in biological systems, attractors have also received much attention. In particular, extensive studies have been done for analyzing the number and length of attractors [5, 13, 18]. Most of existing studies on attractors focus on average case features of random BNs with low indegree (connectivity). However, not much attention has been paid on analysis of attractors in a specific BN. In particular, to our knowledge, only several studies have been done on algorithms for detecting attractors in a given BN.

Detection of a singleton attractor (i.e., an attractor with period 1) is known to be NP-hard by a polynomial time reduction from SAT (the satisfiability problem of Boolean formulas in conjunctive normal form) [1]. Milano and Roli independently proposed a similar reduction [16]. Zhang et al. developed algorithms with guaranteed average case time complexity [23]. For example, it is shown that in the average case, one of the algorithms identifies all singleton attractors in $O(1.19^n)$ time for a random BN with maximum indegree two. However, these algorithms may take $O(2^n)$ or more time in the worst case even if there exist only a small number of singleton attractors. Recently, Leone et al. applied SAT algorithms to identify singleton attractors in a BN [15]. However, they did not focus on the time complexity issue. Tamura and Akutsu studied the time complexity of that approach and showed that detection of a singleton attractor for a BN with maximum indegree k can be reduced to $(k+1)$-SAT [21].

The attractor detection problem has a close relationship with the SAT problem, which is a well-known NP-complete problem, as mentioned above. Extensive studies have been done for developing $O(c^n)$ time algorithms with smaller c for k-SAT, where n is the number of variables and each clause in k-SAT consists of at most k literals. To our knowledge, the fastest algorithms for 3-SAT and 4-SAT developed by Iwama and Tamaki run in $O(1.324^n)$ time and in $O(1.474^n)$ time, respectively [10]. However, no $O((2-\epsilon)^n)$ $(\epsilon > 0)$ time algorithms are known for general SAT. On the other hand, Hirsh developed an $\tilde{O}(1.239^m)$ time algorithm for SAT with m-clauses [8], which was further improved to $\tilde{O}(1.234^m)$ time by Yamamoto [22], where $\tilde{O}(f(m))$ means $O(f(m)poly(m,n))$. However, these algorithms cannot be directly applied to our problem although we utilize the algorithm in [22] as a subroutine.

Recently, we presented an $O(1.787^n)$ time algorithm [21] for detecting a singleton attractor of a given AND/OR BN, in which a Boolean function assigned to each node is restricted to be a conjunction or disjunction of literals as shown in Fig.1 (a). This was the first result in which the computation time of the algorithm is $O((2-\epsilon)^n)$ $(\epsilon > 0)$ with non-restricted indegree although numbers and lengths of attractors of AND/OR BNs had been studied in [4, 7]. In this paper, we present an $O(1.757^n)$ time algorithm with which we succeeded in improving the above algorithm. The $O(1.787^n)$ time algorithm is based on an observation that there exist at most 3 possible assignments (among $2^2 = 4$ assignments) for two adjacent nodes and utilizes Yamamoto's algorithm as a subroutine, where details of the algorithm and analysis are involved. In this paper, we extend this algorithm and obtain an improved $O(1.757^n)$ time algorithm. In this improved algorithm, we make use of an observation that there exist at most 5 possible assignments (among $2^3 = 8$ assignments) for three adjacent nodes, in addition to the observation used in the $O(1.787^n)$ time algorithm. However, this improvement is far from straight-forward. It requires further ideas and much more involved analyses, to be presented in Sections 3 and 4.

2 Preliminaries

Here we briefly review BN and attractors. A BN $N(V,F)$ consists of a set of n nodes V and a set of n Boolean functions F, where $V = \{v_1, v_2, \ldots, v_n\}$ and $F = \{f_1, f_2, \ldots, f_n\}$. 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

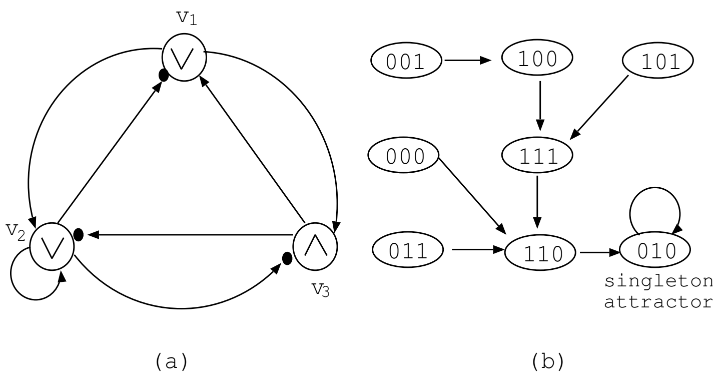


Fig. 1. (a) An example of AND/OR BN where $v_1(t+1) = \overline{v_2(t)} \vee v_3(t)$, $v_2(t+1) = v_1(t) \vee v_2(t) \vee \overline{v_3(t)}$ and $v_3(t+1) = v_1(t) \wedge \overline{v_2(t)}$ are satisfied. "$\wedge$", "$\vee$" and "$\bullet$" mean "AND", "OR" and "NOT" respectively. (b) The state transition of $[v_1, v_2, v_3]$.

at time t is given by the vector $v(t) = [v_1(t), v_2(t), \ldots, v_n(t)]$. This vector is referred as the *Gene Activity Profile* (GAP) of the network at time t, where $v_i(t) = 1$ means that the i-th gene is expressed and $v_i(t) = 0$ means that the i-th gene is not 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 $v_i(t+1) = f_i(v(t))$ for $i = 1, 2, \ldots, n$. When the state of gene v_i at time $t+1$ depends on the states of k_i genes at time t, the *indegree* of gene v_i is k_i and denoted by $id(v_i)$. These $id(v_i)$ ($=k_i$) genes are called *parents* of v_i. The number of genes that are directly influenced by gene v_i is called the *outdegree* of gene v_i and denoted by $od(v_i)$. Furthermore, these $od(v_i)$ genes are called *children* of v_i. The states of all genes are updated simultaneously 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 $v(t) = v(t+p)$. When $p = 1$, an attractor is called a *singleton attractor*. When $p > 1$, it is called a *cyclic attractor*.

For example, a BN where $v_1(t+1) = \overline{v_2(t)} \vee v_3(t)$, $v_2(t+1) = v_1(t) \vee v_2(t) \vee \overline{v_3(t)}$ and $v_3(t+1) = v_1(t) \wedge \overline{v_2(t)}$ is given in Fig. 1 (a). Note that "$\bullet$" means "NOT". The state transition of $[v_1, v_2, v_3]$ is as shown in Fig. 1 (b). $[0, 1, 0]$ is a singleton attractor since $v(t+1) = [0, 1, 0]$ when $v(t) = [0, 1, 0]$.

In this paper, we treat Boolean functions which can be represented by either $({v_{i_1}}^{a_1} \wedge {v_{i_2}}^{a_2} \wedge \cdots \wedge {v_{i_{k_i}}}^{a_{k_i}})^b$ or $({v_{i_1}}^{a_1} \vee {v_{i_2}}^{a_2} \vee \cdots \vee {v_{i_{k_i}}}^{a_{k_i}})^b$ where $v_{i_j}, a_j, b \in \{0, 1\}$. Note that a and b express whether or not negations exist. If every Boolean function of a BN satisfies the above condition, we call it *AND/OR Boolean network*. The number of nodes in AND/OR BN is obtained by counting "AND" and "OR". For example, in Fig. 1 (a), the AND/OR BN has 3 nodes. If no confusion arises, we treat an AND/OR BN as a directed graph as shown in Fig. 1 (a) and denote $N(V, E)$ where V is a set of nodes and E is a set of directed edges.

If a BN is acyclic and does not have self-loops, there is a polynomial time algorithm for detecting an attractor [1, 23]. In such a case, the number of attractors is only one and it is a singleton attractor. On the other hand, if a BN is acyclic and has self-loops, detection of an attractor is NP-hard [1]. In this paper, we allow that a BN has self-loops.

In our main algorithm for detecting a singleton attractor, there are steps, which we call *consistency checks*, to determine whether or not 0-1 assignments for nodes contradict 0-1 assignments for their parent nodes. That is, it checks whether or not a given (partial) 0-1 assignment for nodes is consistent with the definition of a singleton attractor. For example, in Fig. 1 (a), if $v_1(t) = 0$ and $v_2(t) = 0$ are assigned, the consistency check detects a contradiction since $v_1(t+1) = 1 \neq v_1(t)$. Note that consistency checks just detect change of values of nodes from time steps t to $t+1$ and then they do not contribute directly to reduce the computational time of our proposed algorithm. The following lemma shows that consistency checks can be done in ignorable time since our main algorithm takes an exponential time of n and $O(n^k a^n) \ll O((a+\epsilon)^n)$ holds for any $a > 1$ and $\epsilon > 0$, where k is a small positive integer.

Lemma 1. *[21] A consistency check for a GAP or a partial GAP can be done in $O(n^2)$ time.*

In this paper, we treat only singleton attractors. Since $v(t) = v(t+1)$ must hold for a singleton attractor, it suffices to consider only time step t. Thus, we omit t from here on.

As mentioned in Introduction, detection of a singleton attractor for a BN with maximum indegree k is reduced to $(k+1)$-SAT [21]. For example of $k = 2$, $v_1(t+1) = v_2(t) \wedge v_3(t)$ can be represented by 3-SAT as follows:

$$\begin{aligned} v_1(t+1) = v_2(t) \wedge v_3(t) &\Longleftrightarrow v_1 = v_2 \wedge v_3 \\ &\Longleftrightarrow (\overline{v_1} \vee (v_2 \wedge v_3)) \wedge (v_1 \vee (\overline{v_2 \wedge v_3})) \\ &\Longleftrightarrow (\overline{v_1} \vee v_2) \wedge (\overline{v_1} \wedge v_3) \wedge (v_1 \vee \overline{v_2} \vee \overline{v_3}). \end{aligned}$$

However, the computational time increases as k increases.

3 $O(1.774^n)$ time algorithm

In this section, we present an $O(1.774^n)$ time algorithm which detects a singleton attractor of a given AND/OR BN. The $O(1.757^n)$ time algorithm, which is to be shown in the next section, can be obtained by improving the analysis of this $O(1.774^n)$ time algorithm. Although the detection of a singleton attractor for a BN with maximum indegree k can be reduced to $(k+1)$-SAT [21], it cannot be directly applied to our problem since no $O((2-\epsilon)^n)$ $(\epsilon > 0)$ time algorithms are known for SAT with general k.

Let (V, E) denote the structure of a given BN. An edge $(u, v) \in E$ from u to v is called a *non-assigned edge* if no assignment has been done on any of u and v. The notation of (u, v) is treated as if it were undirected although it is actually directed in a given BN. It should be noted there exist at most 3 consistent assignments (among 4 possible assignments) on (u, v) even if there exist self-loops since either a conjunction of literals or a disjunction of literals is assigned to v [21]. For example in Fig. 2, $(k, l) = (0, 0), (0, 1), (1, 1)$ are consistent, but $(k, l) = (1, 0)$ is not consistent. Moreover, two undirected edges $(u, v), (v, w) \in E$ are called *non-assigned neighboring edges* if no assignment has been done on any of u, v and w. The notations of $(u, v), (v, w)$ are also treated as if they were undirected although they are actually directed in a given BN. It should also be noted there exist at most 5 consistent assignments (among 8 possible assignments) on (u, v, w). For example in Fig. 2, $(e, i, j) = (0, 0, 0)$,$(0, 0, 1), (1, 0, 0), (1, 0, 1), (1, 1, 1)$ are consistent, but $(e, i, j) = (0, 1, 0)$,$(0, 1, 1)$,$(1, 1, 0)$ are not consistent.

We show below a pseudo code of the algorithm, which is to be later explained using an example.

```
Begin
    /*STEP1*/
    for $s_1 = 1$ to $n$ do
        $v_{s_1}$ is non-assigned;
    /*STEP2*/
    for $s_1 = 1$ to $n$ do
        for $s_2 = 1$ to $n$ do
            for $s_3 = 1$ to $n$ do
                if $s_1 \neq s_2$ and $s_2 \neq s_3$ and $s_3 \neq s_1$ and $(v_{s_1}, v_{s_2}) \in E$ and
                $(v_{s_2}, v_{s_3}) \in E$ and $v_{s_1}$ is non-assigned and $v_{s_2}$ is non-assigned
                and $v_{s_3}$ is non-assigned
                then examine all possible assignments on $\{(v_{s_1}, v_{s_2}), (v_{s_2}, v_{s_3})\}$,
                which are at most 5 cases, recursively;
                $U$ = the set of nodes whose values were determined at this step;
    /*STEP3*/
    for $s_1 = 1$ to $n$ do
        for $s_2 = 1$ to $n$ do
            if $s_1 \neq s_2$ and $(v_{s_1}, v_{s_2}) \in E$ and $v_{s_1}$ is non-assigned and $v_{s_2}$ is
            non-assigned
            then examine all possible 3 assignments on $(v_{s_1}, v_{s_2})$ recursively;
            $X$ = the set of nodes whose values were determined at this step;
            $W = V - U - X$, $|U| = K$ and $|X| = L$;
    /*STEP4*/
    if $K > \alpha(n - L)$
    then examine all possible assignments on $W$ and then perform consistency
    check;
    else compute an appropriate assignment on $W$ by using Yamamoto's algo-
    rithm and then perform consistency check;
End
```

It is to be noted that the subgraph induced by W is a set of isolated nodes (with self-loops). Therefore, each node v in W is classified into the following types:

type I: the value of v is directly determined from assignment on $U + X$,
type II: the value of v is not directly determined from assignment on $U + X$,

where type I nodes consist of the following:

– The value of v is determined from the values of the input nodes to v,
– v is an input of AND node u and 1 is assigned to u,
– v is an input of OR node u and 0 is assigned to u.

Based on this fact, we can use $\tilde{O}(1.234^m)$ time SAT algorithm for m-clauses to compute an appropriate assignment on type II nodes of W in the following way, where $\tilde{O}(f(m))$ means $O(f(m)poly(m))$. Suppose that $v_{i_1}, \cdots, v_{i_p}$ in W are type II input nodes to node $u \in U + X$. We assume w.l.o.g. that u is an AND node to which 0 is already assigned (we can treat analogously the case where u is an OR node). Furthermore, we can assume w.l.o.g. that u is defined as $u = l_{i_1} \wedge l_{i_2} \wedge \cdots \wedge l_{i_p}$ where l_{i_j} is either v_{i_j} or $\overline{v_{i_j}}$. Then, the constraint of $l_{i_1} \wedge l_{i_2} \wedge \cdots \wedge l_{i_p} = 0$ can be rewritten as a SAT clause $\overline{l_{i_1}} \vee \overline{l_{i_2}} \vee \cdots \vee \overline{l_{i_p}}$. Therefore, we can use the SAT algorithm to find an assignment on W that leads to a singleton attractor.

From the above, it is straight-forward to see the correctness of the algorithm. Thus, we analyze the time complexity.

Lemma 2. *Recursive execution of STEP 2 generates $O(1.71^K)$ assignments.*

Proof. Since at most 5 assignments are examined per three nodes, the number of possible assignments generated at STEP 2 is bounded by $f(K)$ where $f(K)$ is defined by $f(3) = 5$ and $f(K) = 5 \cdot f(K - 3)$. Then, $f(K)$ is $O(5^{K/3})$, which is at most $O(1.710^K)$ ($=O(1.71^K)$). □

Lemma 3. *Recursive execution of STEP 3 generates $O(1.733^L)$ assignments.*

Proof. Since 3 assignments are examined per two nodes, the number of possible assignments generated at STEP 3 is bounded by $f(L)$ where $f(L)$ is defined by $f(2) = 3$ and $f(L) = 3 \cdot f(L - 2)$. Then, $f(L)$ is $O(3^{L/2})$, which is at most $O(1.733^L)$. □

Lemma 4. *If the former part of STEP 4 is executed, the total number of examined assignments is $O(2^{n-K-L} \cdot 1.71^K \cdot 1.733^L)$.*

Lemma 5. *If the latter part of STEP 4 is executed, the total number of examined assignments is $O(1.234^K \cdot 1.71^K \cdot 1.733^L)$.*

Proof. Assume that a SAT clause is constructed when STEP 3 is executed. There must be a directed edge which is terminated by either u or v and initialized by a node $a \in V$ whose value has not been determined yet. We can assume w.o.l.g. (a, u) has been a non-assigned edge before values of (u, v) are assigned. Therefore $\{(a, u), (u, v)\}$ are non-assigned neighboring edges at the beginning of STEP 3 and it contradicts the definition of STEP 2. Thus, no SAT-clauses are constructed in STEP 3. Since the number of constructed SAT clauses in STEP 2 is at most the number of nodes assigned in STEP 2, the lemma holds. □

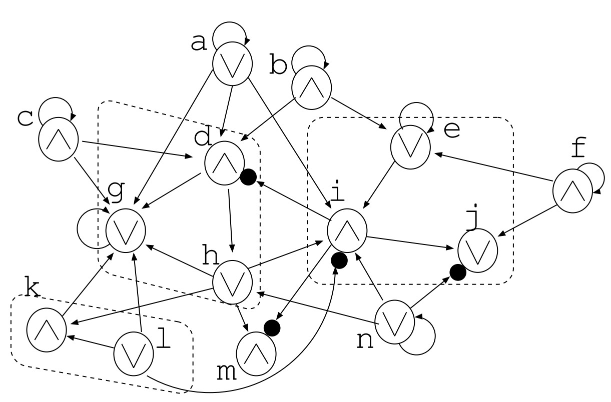


Fig. 2. Example for explaining our proposed algorithm. $\{(d,g),(g,h)\}$ and $\{(e,i),(i,j)\}$ are selected as non-assigned neighboring edges and (k,l) is selected as a non-assigned edge.

Theorem 1. *Detection of a singleton attractor can be done in* $O(1.774^n)$ *time for AND/OR BNs.*

Proof. Assume that L is obtained. If n is a large enough constant, then $2^{n-K-L} \cdot 1.71^K \cdot 1.733^L$ and $1.234^K \cdot 1.71^K \cdot 1.733^L$ are monotone decreasing and increasing function of K respectively. Therefore, the computation time of the proposed algorithm can be bounded by that of the case in which $1.234^K = 2^{n-K-L}$ holds. By solving $1.234^K = 2^{n-K-L}$, we obtain $K = 0.767n - 0.767L$. Therefore, by letting $\alpha = 0.767$, the computation time can be bounded by

$$\max_{0 \le L \le n} \{1.234^{0.767n-0.767L} \cdot 1.71^{0.767n-0.767L} \cdot 1.733^L\} \tag{1}$$

where $0 \le K + L \le n$ must hold. However, $0 \le K + L \le n$ always holds for any L $(0 \le L \le n)$ since $K + L = 0.767n + 0.233L$ holds. Since $(1.234 \cdot 1.71)^{0.767} = 1.773 > 1.733$, (1) is a monotone decreasing function of L if n is a large enough constant. Therefore, (1) takes the maximum value when $L = 0$. Thus, since the computation time of the proposed algorithm can be bounded by assigning $L = 0$ to (1), $1.234^{0.767n} \cdot 1.71^{0.767n} < 1.774^n$ is obtained as the upper bound. □

Example 1. In an example shown in Fig. 2, suppose that $(d,g,h) = (0,1,0)$ and $(e,i,j) = (1,0,1)$ are assigned at STEP 2 and $(k,l) = (0,0)$ is assigned at STEP 3. In STEP 2, SAT clauses $(\overline{a} \vee \overline{b} \vee \overline{c})$, $(a \vee c)$, $(b \vee f)$, $(\overline{a} \vee \overline{n})$, $(f \vee \overline{n})$ are constructed by d, g, e, i, j respectively. Note that d, g, h, e, i, j, k, l are not included in SAT clauses since they are assigned either 0 or 1 directly. Since m and n are determined as 0 by $h = 0$, they are type I nodes. On the other hand, a, b, c, f are Type II nodes. In STEP 3, no SAT clauses are constructed (See Lemma 5). If the former part of STEP 4 is executed, all possible assignments for a, b, c, f, which has 2^4 cases, are examined. Otherwise the SAT problem is solved by Yamamoto's algorithm [22].

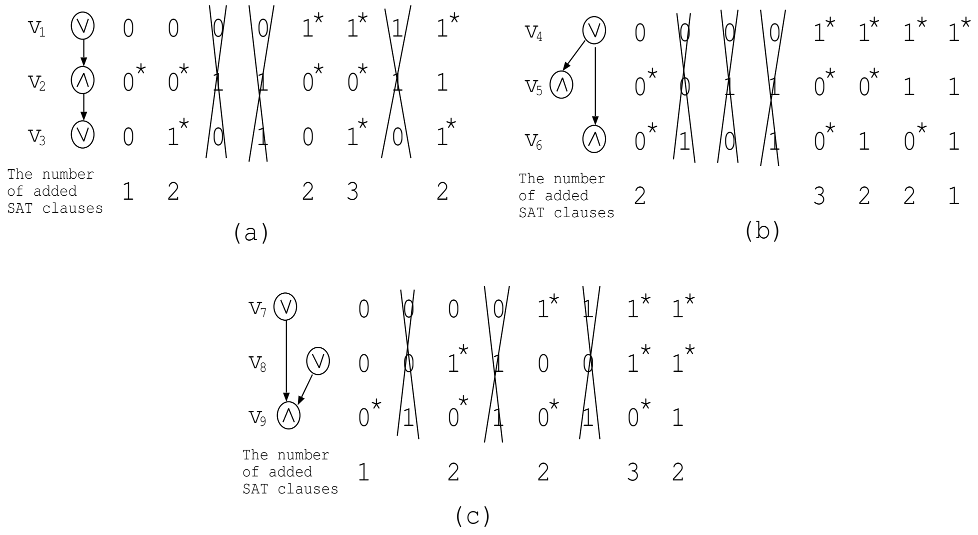


Fig. 3. Three types of non-assigned neighboring edges. By examining all cases, it is seen that the numbers of possible assignments and added SAT clauses are at most 5 and 10 (=1+2+2+2+3) respectively for each type. Note that "*" indicates that the corresponding node adds a SAT clause.

4 Improved Analysis

In this section, we present an improved analysis of the algorithm in Section 3. We show that the algorithm works in $O(1.757^n)$ time. Though this improved analysis is based on the idea used in the improved analysis of our previous algorithm, the analysis given here is much more involved and is far from a straight-forward extension of [19].

For example, in Fig. 3 (a), since v_1 and v_3 are "$\vee$" and v_2 is "$\wedge$", the possible assignments for $[v_1, v_2, v_3]$ are $[0,0,0]$, $[0,0,1]$, $[1,0,0]$, $[1,0,1]$ and $[1,1,1]$. Note that $[0,1,0]$, $[0,1,1]$, $[1,1,0]$ do not satisfy the condition of a singleton attractor. Suppose that $[v_1, v_2, v_3]$=$[0,0,0]$ is assigned. Then, values of parents of v_1 are determined uniquely. Similarly, values of parents of v_3 are also determined uniquely. However, values of parents of v_2 are not determined but a SAT clause which is a disjunction of values of parent nodes of v_2 is constructed. In such a case, we say that v_2 *adds* a SAT clause.

By applying the above discussion to any non-assigned neighboring edges, the numbers of added SAT clauses can be bounded for each case. For example, in Fig. 3 (a), numbers of added SAT clauses by $[v_1, v_2, v_3] = [0,0,0]$, $[0,0,1]$, $[1,0,0]$, $[1,0,1]$ $[1,1,1]$ are 1, 2, 2, 3 and 2 respectively. Similarly, in Fig. 3 (b), the possible assignments for $[v_4, v_5, v_6]$ are $[0,0,0]$, $[1,0,0]$, $[1,0,1]$, $[1,1,0]$ and $[1,1,1]$ and numbers of added SAT clauses by them are 2, 3, 2, 2 and 1 respectively. Furthermore, in Fig. 3 (c), the possible assignments for $[v_7, v_8, v_9]$ are $[0,0,0]$, $[0,1,0]$, $[1,0,0]$, $[1,1,0]$ and $[1,1,1]$ and numbers of added SAT clauses by them are 1, 2, 2, 3 and 2 respectively. Although there are assignments which add less numbers of SAT clauses, by examining all cases it is seen that the worst case is as follows:

– one of the five assignments adds one clause.
– three of the five assignments add two clauses.
– one of the five assignments adds three clause.

From Lemma 2, the number of cases generated in STEP 2 is $O(5^{\frac{K}{3}})$ ($\leq O(1.71^K)$). For each case of them, the number of added SAT clauses is determined according to which one of five assignments is selected in each non-assigned neighboring edges. For example, in Fig. 3, if $[v_1, v_2, v_3] = [0, 0, 0]$, $[v_4, v_5, v_6] = [0, 0, 0]$ and $[v_7, v_8, v_9] = [0, 0, 0]$ are assigned, the total number of added SAT clauses is 4 (=1+2+1). Similarly, if $[v_1, v_2, v_3] = [1, 0, 1]$, $[v_4, v_5, v_6] = [1, 0, 0]$ and $[v_7, v_8, v_9] = [1, 1, 0]$ are assigned, the total number of added SAT clauses is 9 (=3+3+3). Then, the number of cases where one clause is added i times and three clauses are added j times is

$$\sum_{i=0}^{\frac{K}{3}} \sum_{j=0}^{\frac{K}{3}-i} 3^{(\frac{K}{3}-i-j)} \cdot {}_{\frac{K}{3}}C_i \cdot {}_{\frac{K}{3}-i}C_j$$

since the number of cases where two clauses are added is $3^{\frac{K}{3}-i-j}$. Moreover, the total number of added SAT clauses in this case is $\frac{2K}{3} - i + j$. Therefore, the computation time in the case where the latter part of STEP 4 is executed is bounded by

$$g(K, L) = 1.733^L \cdot \sum_{i=0}^{\frac{K}{3}} \sum_{j=0}^{\frac{K}{3}-i} 1.234^{(\frac{2K}{3}-i+j)} \cdot 3^{(\frac{K}{3}-i-j)} \cdot {}_{\frac{K}{3}}C_i \cdot {}_{\frac{K}{3}-i}C_j.$$

To estimate $g(K, L)$, we show the following lemmas. Let β and γ be constants where $0 \leq \beta \leq \frac{1}{3}$ and $0 \leq \gamma \leq \frac{1}{3} - \beta$ hold.

Lemma 6. *${}_{\frac{K}{3}}C_{\beta K}$ is $O\left(\left\{\frac{\frac{1}{3}}{\beta^{3\beta}\cdot(\frac{1}{3}-\beta)^{1-3\beta}}\right\}^{\frac{K}{3}}\right)$ and ${}_{\frac{K}{3}-\beta K}C_{\gamma K}$ is $O\left(\left\{\frac{(\frac{1}{3}-\beta)^{1-3\beta}}{\gamma^{3\gamma}\cdot(\frac{1}{3}-\beta-\gamma)^{1-3\beta-3\gamma}}\right\}^{\frac{K}{3}}\right)$.*

Proof. From Stirling's formula, ${}_{\frac{K}{3}}C_{\beta K}$ is $O(p(K, \beta))$ where

$$p(K, \beta) = \frac{(\frac{K}{3})^{\frac{K}{3}}}{(\beta K)^{\beta K} \cdot (\frac{K}{3} - \beta K)^{\frac{K}{3}-\beta K}} = \left\{\frac{\frac{K}{3}}{(\beta K)^{3\beta} \cdot (\frac{K}{3} - \beta K)^{1-3\beta}}\right\}^{\frac{K}{3}}$$

$$= \left\{\frac{\frac{1}{3}}{\beta^{3\beta} \cdot (\frac{1}{3} - \beta)^{1-3\beta}}\right\}^{\frac{K}{3}}$$

Similarly, ${}_{\frac{K}{3}-\beta K}C_{\gamma K}$ is $O(q(K,\beta,\gamma))$ where

$$\begin{aligned} q(K,\beta,\gamma) &= \frac{(\frac{K}{3}-\beta K)^{\frac{K}{3}-\beta K}}{(\gamma K)^{\gamma K}\cdot(\frac{K}{3}-\beta K-\gamma K)^{\frac{K}{3}-\beta K-\gamma K}} \\ &= \left\{\frac{(\frac{K}{3}-\beta K)^{1-3\beta}}{(\gamma K)^{3\gamma}\cdot(\frac{K}{3}-\beta K-\gamma K)^{1-3\beta-3\gamma}}\right\}^{\frac{K}{3}} \\ &= \left\{\frac{(\frac{1}{3}-\beta)^{1-3\beta}}{\gamma^{3\gamma}\cdot(\frac{1}{3}-\beta-\gamma)^{1-3\beta-3\gamma}}\right\}^{\frac{K}{3}} \end{aligned}$$

□

To estimate terms including ${}_{\frac{K}{3}}C_{\beta K}$ and ${}_{\frac{K}{3}-\beta K}C_{\gamma K}$, we divide β and γ into N and $2N$ intervals respectively.

Lemma 7. *Suppose that $\frac{i-1}{3N} \le \beta \le \frac{i}{3N}$, $\frac{j-1}{6N} \le \gamma \le \frac{j}{6N}$ and N is a positive even integer.*

1. *If $i \le \frac{N}{2}$, then ${}_{\frac{K}{3}}C_{\beta K} \le {}_{\frac{K}{3}}C_{\frac{iK}{3N}}$ holds. Otherwise ${}_{\frac{K}{3}}C_{\beta K} \le {}_{\frac{K}{3}}C_{\frac{(i-1)K}{3N}}$ holds.*
2. *If $j \le N-i+1$, then ${}_{\frac{K}{3}-\beta K}C_{\gamma K} \le {}_{\frac{K}{3}-\frac{(i-1)K}{3N}}C_{\frac{jK}{6N}}$ holds. Otherwise ${}_{\frac{K}{3}-\beta K}C_{\gamma K} \le {}_{\frac{K}{3}-\frac{(i-1)K}{3N}}C_{\frac{(j-1)K}{6N}}$ holds.*

Proof. 1. If K is a constant, then ${}_{\frac{K}{3}}C_{\beta K}$ is convex upward with β and takes the maximum value when $\beta = \frac{1}{6}$. By solving $\frac{i}{3N} = \frac{1}{6}$, we obtain $i = \frac{N}{2}$ (See also Fig. 4 (a)). Note that $\frac{K}{3}$ and $\frac{N}{2}$ are integers from their definitions.

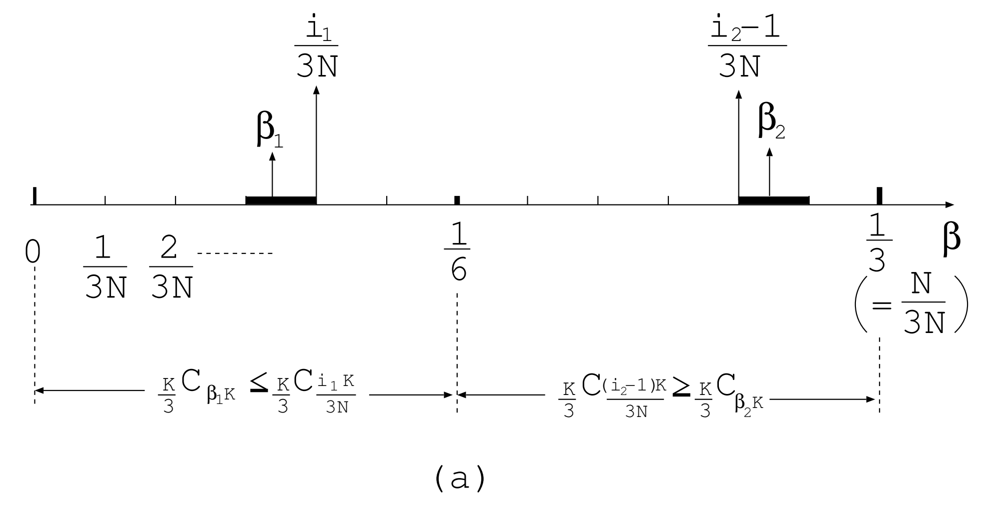
2. From the assumption, ${}_{\frac{K}{3}-\beta K}C_{\gamma K} \le {}_{\frac{K}{3}-\frac{i-1}{3N}K}C_{\gamma K}$ holds. If K is a constant, then ${}_{\frac{K}{3}-\frac{i-1}{3N}K}C_{\gamma K}$ is convex upward with γ and takes the maximum value when $\gamma = \frac{1}{2}(\frac{1}{3}-\frac{i-1}{3N})$. Note that i and N are constants. By solving $\frac{1}{2}(\frac{1}{3}-\frac{i-1}{3N}) = \frac{j}{6N}$, we obtain $j = N-i+1$ (See also Fig. 4 (b)).

□

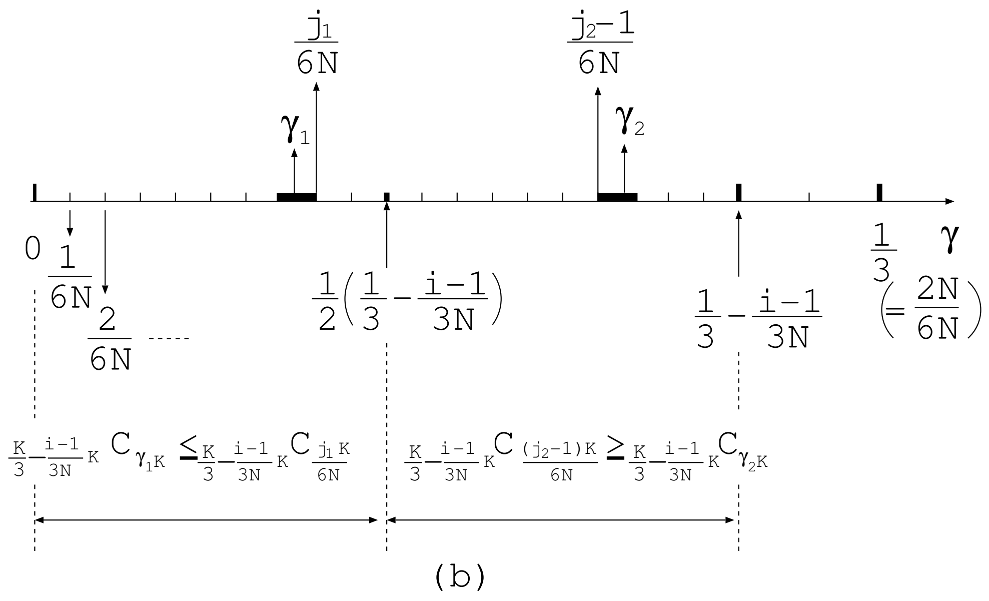
Theorem 2. *Detection of a singleton attractor can be done in $O(1.757^n)$ time for AND/OR BNs.*

Proof.

$$\begin{aligned} g(K,L) &= 1.733^L \cdot 1.234^{\frac{2K}{3}} \cdot 3^{\frac{K}{3}} \cdot \sum_{i=0}^{\frac{K}{3}} (1.234\cdot 3)^{-i} \cdot {}_{\frac{K}{3}}C_i \cdot \sum_{j=0}^{\frac{K}{3}-i}\left(\frac{1.234}{3}\right)^j \cdot {}_{\frac{K}{3}-i}C_j \\ &< 1.733^L \cdot 1.234^{\frac{2K}{3}} \cdot 3^{\frac{K}{3}} \cdot \sum_{i=0}^{\frac{K}{3}} 0.2702^i \cdot {}_{\frac{K}{3}}C_i \cdot \sum_{j=0}^{\frac{K}{3}-i} 0.4114^j \cdot {}_{\frac{K}{3}-i}C_j \end{aligned}$$



$\frac{i_1}{3N}$
$\frac{i_2-1}{3N}$
β_1
β_2
0
$\frac{1}{3N}$
$\frac{2}{3N}$
$\frac{1}{6}$
$\frac{1}{3}$
$\left(=\frac{N}{3N}\right)$
β
${}_{\frac{K}{3}}C_{\beta_1 K} \le {}_{\frac{K}{3}}C_{\frac{i_1 K}{3N}}$
${}_{\frac{K}{3}}C_{\frac{(i_2-1)K}{3N}} \ge {}_{\frac{K}{3}}C_{\beta_2 K}$
(a)



$\frac{j_1}{6N}$
$\frac{j_2-1}{6N}$
γ_1
γ_2
0
$\frac{1}{6N}$
$\frac{2}{6N}$
$\frac{1}{2}\left(\frac{1}{3}-\frac{i-1}{3N}\right)$
$\frac{1}{3}-\frac{i-1}{3N}$
$\frac{1}{3}$
$\left(=\frac{2N}{6N}\right)$
γ
${}_{\frac{K}{3}-\frac{i-1}{3N}K}C_{\gamma_1 K} \le {}_{\frac{K}{3}-\frac{i-1}{3N}K}C_{\frac{j_1 K}{6N}}$
${}_{\frac{K}{3}-\frac{i-1}{3N}K}C_{\frac{(j_2-1)K}{6N}} \ge {}_{\frac{K}{3}-\frac{i-1}{3N}K}C_{\gamma_2 K}$
(b)

Fig. 4. (a)If $\beta \le \frac{1}{6}$, then ${}_{\frac{K}{3}}C_{\beta K} \le {}_{\frac{K}{3}}C_{\frac{iK}{3N}}$ holds. Otherwise ${}_{\frac{K}{3}}C_{\beta K} \le {}_{\frac{K}{3}}C_{\frac{(i-1)K}{3N}}$ holds. (b)If $j \le \frac{1}{2}(\frac{1}{3}-\beta)$, then ${}_{\frac{K}{3}-\beta K}C_{\gamma K} \le {}_{\frac{K}{3}-\frac{(i-1)K}{3N}}C_{\frac{jK}{6N}}$ holds. Otherwise ${}_{\frac{K}{3}-\beta K}C_{\gamma K} \le {}_{\frac{K}{3}-\frac{(i-1)K}{3N}}C_{\frac{(j-1)K}{6N}}$ holds.

$$
\begin{aligned}
&< 1.733^L \cdot (1.234^2 \cdot 3)^{\frac{K}{3}} \cdot \Big(\frac{K}{3N} \cdot \frac{K}{6N}\Big) \cdot \\
&\Bigg\{ \sum_{i=1}^{\frac{N}{2}} f_1\Big(\frac{i-1}{3N}\Big) \cdot f_2\Big(\frac{i}{3N}\Big) \sum_{j=1}^{N-i} f_3\Big(\frac{j-1}{6N}\Big) \cdot f_4\Big(\frac{i-1}{3N}, \frac{j}{6N}\Big) \\
&+ \sum_{i=1}^{\frac{N}{2}} f_1\Big(\frac{i-1}{3N}\Big) \cdot f_2\Big(\frac{i}{3N}\Big) \sum_{j=N-i+1}^{2(N-i)} f_3\Big(\frac{j-1}{6N}\Big) \cdot f_4\Big(\frac{i-1}{3N}, \frac{j-1}{6N}\Big) \\
&+ \sum_{i=\frac{N}{2}+1}^{N} f_1\Big(\frac{i-1}{3N}\Big) \cdot f_2\Big(\frac{i-1}{3N}\Big) \sum_{j=1}^{N-i} f_3\Big(\frac{j-1}{6N}\Big) \cdot f_4\Big(\frac{i-1}{3N}, \frac{j}{6N}\Big) \\
&+ \sum_{i=\frac{N}{2}+1}^{N} f_1\Big(\frac{i-1}{3N}\Big) \cdot f_2\Big(\frac{i-1}{3N}\Big) \sum_{j=N-i+1}^{2(N-i)} f_3\Big(\frac{j-1}{6N}\Big) \cdot f_4\Big(\frac{i-1}{3N}, \frac{j-1}{6N}\Big) \Bigg\} \\
&= 1.733^L \cdot (1.234^2 \cdot 3)^{\frac{K}{3}} \cdot poly(K) \cdot h(K)
\end{aligned}
$$

where

$$
f_1(i) = (0.2702^{3i})^{\frac{K}{3}}, \quad f_2(i) = \left\{ \frac{\frac{1}{3}}{i^{3i} \cdot (\frac{1}{3} - i)^{1-3i}} \right\}^{\frac{K}{3}},
$$

$$
f_3(j) = (0.4114^{3j})^{\frac{K}{3}}, \quad f_4(i,j) = \left\{ \frac{(\frac{1}{3} - i)^{1-3i}}{j^{3j} \cdot (\frac{1}{3} - i - j)^{1-3i-3j}} \right\}^{\frac{K}{3}}
$$

hold from Lemma 6. By setting $N = 10000$, it can be confirmed that $h(K) < O(1.683^{\frac{K}{3}})$. Although this confirmation can be done manually, we used a computer since it requires a vast amount of routine works. Note that larger N yields a smaller upper bound of $h(K)$. However, $O(1.683^{\frac{K}{3}})$ is almost not improved by N which is larger than 10000. Thus, if the latter part of STEP 4 is executed, the computation time of the proposed algorithm is $O((1.234^2 \cdot 3 \cdot 1.683)^{\frac{K}{3}} \cdot 1.733^L) < O(1.974^K \cdot 1.733^L)$. Similar to the proof of the previous theorem, assume that L is obtained. If n is a large enough constant, then $2^{n-K-L} \cdot 1.71^K \cdot 1.733^L$ and $1.974^K \cdot 1.733^L$ are monotone decreasing and increasing functions of K respectively. Therefore, the computation time of the proposed algorithm can be bounded by that of the case in which $1.974^K = 2^{n-K-L} \cdot 1.71^K$ holds. By solving this equation, we obtain $K = 0.8286n - 0.8286L$. Therefore, by letting $\alpha = 0.8286$, the computation time can be bounded by

$$
\max_{0 \le L \le n} \{1.974^{0.8286n - 0.8286L} \cdot 1.733^L\} \tag{2}
$$

where $0 \le K + L \le n$ must hold. However, $0 \le K + L \le n$ always holds for any L $(0 \le L \le n)$ since $K + L = 0.8286n + 0.1714L$ holds. Since $1.974^{0.8286} = 1.757 > 1.733$, (2) is a monotone decreasing function of L if n is a large enough constant. Therefore, (2) takes the maximum value when $L = 0$. Thus, since the computation time of the proposed algorithm can be bounded by assigning $L = 0$ to (2),

$$O(1.974^{0.8286n}) < O(1.757^n)$$

is obtained as the upper bound. □

5 Conclusion and future works

We improved the computation time of the algorithm for detecting a singleton attractor in a given AND/OR BN from $O(1.787^n)$ [21] to $O(1.757^n)$. Readers may think that further improvement is possible by making use of 4 or more adjacent nodes (in addition to 2 and 3 adjacent nodes). However, it is unclear whether such a simple idea leads to an improvement. At least, algorithm and analysis would be quite involved. Thus, improvement of the proposed algorithm is left as an open problem. Extension of the proposed algorithm to the enumeration problem (i.e., efficient and output-sensitive enumeration of all singleton attractors) is also left as an open problem.

For the singleton attractor detection problem, every BN can be transformed into an AND/OR BN although additional nodes are needed as discussed in [21]. If the number of additional nodes is less than $0.229n$, the computation time of our algorithm is still $O((2-\epsilon)^n)$ $(\epsilon > 0)$ for general BNs because $1.757^{n+0.229n} < 2^n$. This value (0.229n) was also improved from that of [21]. Since canalizing functions and nested canalizing functions are known to be good models for regulatory rules of eukaryotic genes [14, 19], the number of such additional nodes are considered to be not large for real biological networks when compared to the case where Boolean functions are assigned to nodes purely at random. It also deserves to mention that the class of nested canalizing functions is equal to that of unate cascade functions [11]. An experimental comparison of proposed algorithms is also one of our future works.

Although this paper focused on the Boolean network as a biological network model, the proposed techniques might be useful for designing algorithms which find steady states in other models [17] as already discussed in [21]. Application and extension of the proposed techniques to other types of biological networks are important future works.

References

1. T. Akutsu, S. Kuhara, O. Maruyama, S. Miyano, A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions, *Genome Informatics*, 9:151–160, 1998.
2. T. Akutsu, S. Miyano, S. Kuhara, Inferring qualitative relations in genetic networks and metabolic pathways, *Bioinformatics*, 16:727–734, 2000.
3. R. Albert, A-L. Barabasi, Dynamics of complex systems: Scaling laws for the period of Boolean networks, *Physical Review Letters*, 84:5660–5663, 2000.

4. J. Aracena, J. Demongeot, E. Goles, Fixed points and maximal independent sets in AND-OR networks, *Discrete Applied Mathematics*, 138:277-288, 2004.
5. B. Drossel, T. Mihaljev and F. Greil, Number and length of attractors in a critical Kauffman model with connectivity one, *Physical Review Letters*, 94:088701, 2005.
6. L. Glass, S. A. Kauffman, The logical analysis of continuous, nonlinear biochemical control networks, *Journal of Theoretical Biology*, 39:103–129, 1973.
7. E. Goles, G. Hernandez, Dynamical behavior of Kauffman networks with AND-OR gates, *Journal of Biological Systems*, 8(2):151–175, 2000.
8. E. A. Hirsch, New worst-case upper bounds for SAT, *Journal of Automated Reasoning*, 24:397–420, 2000.
9. S. Huang, Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery, *Journal of Molecular Medicine*, 77(6):469–480, 1999.
10. K. Iwama, S. Tamaki, Improved upper bounds for 3-SAT, *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*, pp. 328, 2004.
11. A. S. Jarrah, B. Raposa, R. Laubenbacher, Nested canalyzing, unate cascade, and polynomial functions *Physica D*, 233(2):167–174, 2007.
12. S. Kauffman, Metabolic stability and epigenesis in randomly connected genetic nets, *Journal of Theoretical Biology*, 22:437–467, 1968.
13. S. Kauffman, *The Origin of Order*: Self-organization and selection in evolution, Oxford Univ. Press, New York, 1993.
14. S. Kauffman, C, Peterson, B. Samuelsson, C. Troein, Random Boolean network models and the yeast transcriptional network, *Proceedings of the National Academy of Sciences*, 100(25):14796-14799, 2003.
15. M. Leone, A. Pagnani, G. Parisi, O. Zagordi, Finite size corrections to random Boolean networks, *cond-mat*/0611088, 2006.
16. M. Milano, A. Roli, Solving the satisfiability problem through Boolean networks, *Lecture notes in Artificial Intelligence*, 1792:72–93, 2000.
17. A. Mochizuki, An analytical study of the number of steady states in gene regulatory networks, *J. Theoret. Biol.*, 236:291–310, 2005.
18. B. Samuelsson and C. Troein, Superpolynomial growth in the number of attractors in Kauffman networks, *Physical Review Letters*, 90:098701, 2003.
19. I. Shmulevich, S. Kauffman, Activities and sensitivities in Boolean network models, *Physical Review Letters*, 93(4):048701, 2004.
20. R. Somogyi, C. A. Sniegoski, Modeling the complexity of genetic networks: Understanding multigenic and pleitropic regulation, *Complexity*, 1(6):45–63, 1996.
21. T. Tamura, T. Akutsu, Detecting a Singleton Attractor in a Boolean Network Utilizing SAT Algorithms (submitted). Preliminary version has appeared as follow: T. Tamura, T. Akutsu, An $O(1.787^n)$-time Algorithm for Detecting a Singleton Attractor in a Boolean Network Consisting of AND/OR Nodes. *Proc. International Symposium on Fundamentals of Computation Theory*, pp. 494–505, 2007.
22. M. Yamamoto, An improved $\tilde{O}(1.234^m)$-time deterministic algorithm for SAT, *Proc. International Symposium on Algorithms and Computation*, pp. 644–653, 2005.
23. S. Zhang, M. Hayashida, T. Akutsu, W. Ching, M. K. Ng, Algorithms for finding small attractors in Boolean networks, *EURASIP Journal on Bioinformatics and Systems Biology*, 2007:20180, 2007.