IPSJ Transactions on Bioinformatics Vol. 1 23–34 (Nov. 2008)

Original Paper

Analyses and Algorithms for Predecessor and Control Problems for Boolean Networks of Bounded Indegree

Tatsuya Akutsu,[†1] Morihiro Hayashida,[†1]
Shu-Qin Zhang,[†2] Wai-Ki Ching[†3]
and Michael K. Ng[†4]

We study the predecessor and control problems for Boolean networks (BNs). The predecessor problem is to determine whether there exists a global state that transits to a given global state in a given BN, and the control problem is to find a sequence of 0-1 vectors for control nodes in a given BN which leads the BN to a desired global state. The predecessor problem is useful both for the control problem for BNs and for analysis of landscape of basins of attractions in BNs. In this paper, we focus on BNs of bounded indegree and show some hardness results on the computational complexity of the predecessor and control problems. We also present simple algorithms for the predecessor problem that are much faster than the naive exhaustive search-based algorithm. Furthermore, we show some results on distribution of predecessors, which leads to an improved algorithm for the control problem for BNs of bounded indegree.

1. Introduction

Analysis of genetic networks is an important research topic in bioinformatics. Mathematical models of genetic networks are usually required for such analysis and thus many models have been proposed and applied. Among these, the *Boolean network* (BN, in short) model has received much attention [14]. BN is a very simple model: each node (e.g., gene) takes either 0 (inactive) or 1 (active) and the states of nodes change synchronously according to regulation rules given as Boolean functions.

Extensive studies have been done on average case analysis of the number and length of attractors in randomly generated BNs [9),14),21)], where attractors correspond to steady-states. Recently, several methods have been developed for efficiently finding or enumerating attractors in BNs [8),11),12),24)], whereas it is known that finding a singleton attractor (i.e., a fixed point) is NP-hard [2),16)]. Devloo, et al. developed a method using transformation to a constraint satisfaction problem [8)]. Garg, et al. developed a method based on Binary Decision Diagrams (BDDs) [11)]. Irons developed a method that makes use of small subnetworks [12)]. However, theoretical analysis of the average case complexity was not performed in these works. We recently developed algorithms for identifying singleton attractors and small attractors and analyzed the average case time complexities of these algorithms [24)].

Finding a sequence of control actions for BNs is another important problem on BNs, which is abbreviated as BN-CONTROL in this paper. Inspired from works on control of the probabilistic Boolean network (PBN, in short) model [7),18)], we studied BN-CONTROL [3)]. We showed that BN-CONTROL is NP-hard even in considerably restricted cases [3)]. However, we may be able to develop algorithms that are much faster than exhaustive search based algorithms. Though we have not yet fully succeeded to develop such algorithms, we encountered the problem of finding a global state transiting to a given global state, which is known as the *predecessor problem* for BNs [5),6)] and is abbreviated as BN-PREDECESSOR in this paper. In other words, BN-PREDECESSOR is to find an input node to a specified node in a state transition diagram of a BN. It should be noted that the problem is trivial once a state transition diagram is constructed. However, the number of nodes of a state transition diagram is 2^n where n is the number of nodes in a BN. Therefore, faster algorithms should be developed.

BN-PREDECESSOR has some potential applications. If the target state does not have a predecessor state, there does not exist a control sequence. Thus, BN-PREDECESSOR may be used for pre-processing for the control problem. BN-PREDECESSOR may also be useful for identifying the basin of attraction by enumerating predecessor states recursively [23)], which has a potential application to design of BNs with a prescribed attractor structure [19)]. Besides, BN-

†1 Bioinformatics Center, Institute for Chemical Research, Kyoto University
†2 School of Mathematical Sciences, Fudan University
†3 Department of Mathematics, The University of Hong Kong
†4 Department of Mathematics, Hong Kong Baptist University
⋆1 Several results in this paper are included in a preliminary conference version appeared in Proc. Fifth IEEE International Workshop on Genomic Signal Processing and Statistics, 2007.

PREDECESSOR may also be useful for analyzing self-organized criticality in BNs [6].

For the predecessor problem, some studies have been done. Barrett, et al. studied the computational complexity of the predecessor problem for BNs and other discrete dynamical systems [5]. They showed that BN-PREDECESSOR is NP-hard even for BNs with planar graph structures, whereas they presented a polynomial time algorithm for BNs of bounded tree-width. Coppersmith also studied the computational complexity of BN-PREDECESSOR and distribution of predecessors [6]. She showed that BN-PREDECESSOR for BNs with maximum indegree K can be reduced to K-SAT, where K-SAT denotes the Boolean satisfiability problem for a set of clauses each of which consists of at most K-literals. She also showed that as n grows, the ratio of global states having at least one predecessors converges to $1/e$ and 0 for general BNs and for BNs with maximum indegree K, respectively.

In this paper, we study BN-PREDECESSOR, its variant, and BN-CONTROL with focusing on cases where the maximum indegee is bounded by some constant K. It is to be noted that many real networks are considered to have small maximum indegrees and most of existing studies on BNs also focused on cases of bounded indegree. We show that BN-PREDECESSOR is NP-hard even for BNs with $K = 3$. We also show that the 2-predecessor problem, which is a problem of finding a predecessor of a predecessor, is NP-hard even for BNs with $K = 2$. Besides, we show that BN-CONTROL is NP-hard even for $K = 2$, which strengthens a previous NP-hardness result on BN-CONTROL [3]. Next, based on our previous algorithms for identifying singleton attractors [24], we develop algorithms for identifying all predecessors, all of which are much faster than the naive enumeration algorithm. We also show results of computational experiments on these algorithms. Then, based on studies by Coppersmith [6], we show some results on distributions of predecessors and predecessors of predecessors. Furthermore, by making use of some of these results, we develop an improved algorithm for BN-CONTROL for bounded indegree. Finally, we conclude with future work.

2. Preliminaries

In this section, we briefly review the Boolean network model [14], BN-PREDECESSOR [5,6], and a control problem on BN (BN-CONTROL, in short) [3].

2.1 Boolean Network and BN-PREDECESSOR

A BN consists of a set of n nodes V and n Boolean functions F, where $V = \{v_1, \ldots, v_n\}$ and $F = \{f_1, \ldots, f_n\}$. In general, V and F correspond to a set of genes and a set of gene regulatory rules, respectively. Each node takes either 0 or 1 at each discrete time t, where 1 (resp. 0) means that the corresponding gene expresses (resp. does not express) at time t. The state of v_i at time t is denoted by $v_i(t)$. The *global state* of a BN (or simply the *state* of a BN) at time step t is denoted by the vector $\mathbf{v}(t) = [v_1(t), \ldots, v_n(t)]$. A regulation rule for each node is given in the form of a Boolean function and the states of nodes change synchronously. A node v_i has k_i incoming nodes $v_{i_1}, \ldots, v_{i_{k_i}}$ and the state of v_i at time $t+1$ is determined by $v_i(t+1) = f_i(v_{i_1}(t), \ldots, v_{i_{k_i}}(t))$, where f_i is a Boolean function with k_i input variables. The number k_i is called the *indegree* of node v_i. We also write $v_i(t+1) = f_i(\mathbf{v}(t))$ to denote the regulation rule for v_i and $\mathbf{v}(t+1) = \mathbf{f}(\mathbf{v}(t))$ to denote the regulation rule for the whole BN. A specific global state can be written as an n-dimensional binary vector $[b_1, \ldots, b_n]$. If we consider all 2^n possible states and compute their respective next states, a list of 2^n one-step state transitions can be obtained. These 2^n transitions fully characterize the dynamics of a BN and the table representing these 2^n transitions is called the *state transition table*. We can also associate a directed graph called *state transition diagram* in which a set of nodes is the set of all possible 2^n global states, and there exists a directed edge from $\mathbf{u}$ to $\mathbf{v}$ if and only if $\mathbf{v} = \mathbf{f}(\mathbf{u})$ holds. It is not difficult to see from the definition of BN that each node has exactly one outgoing edge.

Starting from an initial global state, a BN will eventually reach a set of global states, called an *attractor* (a directed cycle in the state transition diagram). An attractor consisting of only one global state is called a *singleton attractor*. That is, $\mathbf{v}$ is a singleton attractor if $\mathbf{v} = \mathbf{f}(\mathbf{v})$. Otherwise, it is called a *cyclic attractor* with period p if it consists of p global states. The set of all global states that eventually evolve into the same attractor is called the *basin of attraction*.

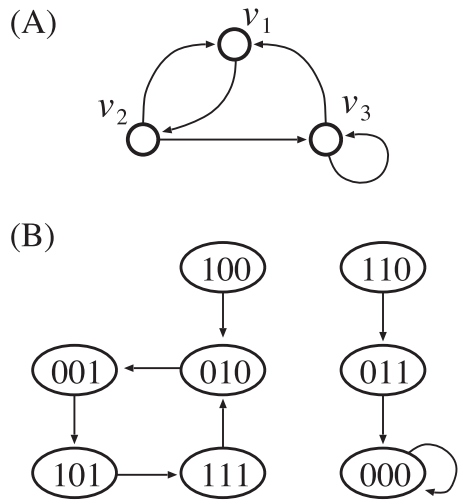


Fig. 1 Example of (A) Boolean network and (B) state transition diagram, where $v_1(t+1) = \overline{v_2(t)} \wedge v_3(t)$, $v_2(t+1) = v_1(t)$ and $v_3(t+1) = v_2(t) \oplus v_3(t)$.

Different basins of attraction correspond to different connected components in the state transition diagram, and each connected component contains exactly one directed cycle.

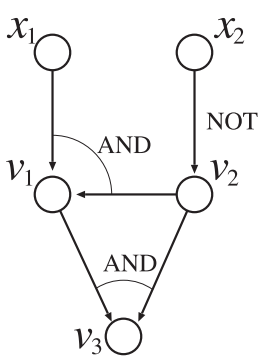
For a global state $\mathbf{v}$, a global state $\mathbf{u}$ is called a *predecessor* of $\mathbf{v}$ if $\mathbf{v} = \mathbf{f}(\mathbf{u})$. That is, $\mathbf{u}$ is a predecessor of $\mathbf{v}$ if there is an edge from $\mathbf{u}$ to $\mathbf{v}$ in the state transition diagram of a given BN. Then, BN-PREDECESSOR is defined as follows [5),6)].

Definition 1 (BN-PREDECESSOR) [5),6)]
Suppose that a BN (V, F) and a global state $\mathbf{v}^1$ are given. Then, the problem is to find a global state $\mathbf{v}^0$ such that $\mathbf{v}^1 = \mathbf{f}(\mathbf{v}^0)$. If there does not exist such a global state, "None" should be the output.

An example of a BN is given in **Fig. 1** along with the corresponding state transition diagram. In this case, there is one singleton attractor and one cyclic attractor with period 4. In BN-PREDECESSOR, either 100 or 111 should be output for 010, and "none" should be output for 100.

2.2 Control of Boolean Network

In BN-CONTROL [3)], there are two types of nodes: *internal nodes* and *external nodes*, where internal nodes correspond to usual nodes (i.e., genes) in BN and external nodes correspond to control nodes. Let a set V of $n+m$ nodes be $V = \{v_1, \ldots, v_n, v_{n+1}, \ldots, v_{n+m}\}$, where $v_1, \ldots, v_n$ are internal nodes and $v_{n+1}, \ldots, v_{n+m}$ are external nodes. For convenience, we use x_i to denote an external node v_{n+i}. Then, $v_i(t+1)$ for $i = 1, \ldots, n$ are determined by $v_i(t+1) = f_i(v_{i_1}(t), \ldots, v_{i_{k_i}}(t))$, where each v_{i_k} is either an internal node or an external node. Here, we let $\mathbf{v}(t) = [v_1(t), \ldots, v_n(t)]$ and $\mathbf{x}(t) = [x_1(t), \ldots, x_m(t)]$. We can describe the dynamics of a BN by $\mathbf{v}(t+1) = \mathbf{f}(\mathbf{v}(t), \mathbf{x}(t))$, where $\mathbf{x}(t)$'s are determined externally. Then, BN-CONTROL is defined as follows (see also **Fig. 2**).



$\mathbf{v}^1$ (initial)	0	0	0
$\mathbf{v}^M$ (target)	1	0	1

t	v_1	v_2	v_3	x_1	x_2
0	0	0	0	1	0
1	0	1	0	1	0
2	1	1	0	1	1
3	1	0	1	0	0

Fig. 2 BN-CONTROL is, given initial and desired states of internal nodes (v_1, v_2, v_3), to compute a sequence of states of external nodes (x_1, x_2) that leads to the desired state.

Definition 2 (BN-CONTROL) [3)]
Suppose that for a BN, we are given an initial state of the network (for internal nodes) $\mathbf{v}^0$ and the desired state of the network $\mathbf{v}^M$ at the M-th time step. Then, the problem is to find a sequence of 0-1 vectors $\langle \mathbf{x}(0), \ldots, \mathbf{x}(M) \rangle$ such that $\mathbf{v}(0) = \mathbf{v}^0$ and $\mathbf{v}(M) = \mathbf{v}^M$. If there does not exist such a sequence, "None" should be the output.

As mentioned in Introduction, Datta, et al. studied control of PBN [7)]. They proposed a dynamic programming algorithm, which can also be applied to BN. Here, we briefly review their method in the context of BN [4)].

We use a table $D[b_1, \ldots, b_n, t]$, where each entry takes either 0 or 1.

$D[b_1, \ldots, b_n, t]$ takes 1 if there exists a desired control sequence beginning from a state $[b_1, \ldots, b_n]$ at time t. This table is computed from $t = M$ to $t = 0$ by using a dynamic programming procedure based on the following recurrence:

$$D[b_1, \ldots, b_n, M] = \begin{cases} 1, & \text{if } [b_1, \ldots, b_n] = \mathbf{v}^M, \\ 0, & \text{otherwise,} \end{cases}$$

$$D[b_1, \ldots, b_n, t-1] = \begin{cases} 1, & \text{if there exists } (\mathbf{a}, \mathbf{x}) \text{ such that} \\ & D[a_1, \ldots, a_n, t] = 1 \text{ and } \mathbf{a} = \mathbf{f}(\mathbf{b}, \mathbf{x}), \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathbf{b} = [b_1, \ldots, b_n]$ and $\mathbf{a} = [a_1, \ldots, a_n]$. Then, there exists a desired control sequence if and only if $D[(\mathbf{v}^0)_1, \ldots, (\mathbf{v}^0)_n, 0] = 1$ holds, where $\mathbf{v}_i$ denotes the ith element of a vector $\mathbf{v}$.

In this method, the size of table $D[b_1, \ldots, b_n, t]$ is clearly $O(M \cdot 2^n)$. Moreover, we should examine pairs of $O(2^n)$ internal states and $O(2^m)$ external states for each time t. Thus, it requires $O(nM \cdot 2^{m+n})$ time even if the maximum indegree of a BN is bounded by a constant K because additional $O(n)$ factor is required to compute $\mathbf{f}(\mathbf{b}, \mathbf{x})$.

3. Hardness Results

In this section, we present some hardness results on the computational complexity of BN-PREDECESSOR, its variant, and BN-CONTROL.

As mentioned in Introduction, Barrett, et al. proved that BN-PREDECESSOR is NP-hard even for BNs having planar structures [5]. However, it is not very clear from their results whether BN-PREDECESSOR is NP-hard for BNs with $K = 3$. Although the following result is almost obvious and may be implied by some result in 5), we show it here since it gives at least a very simple proof.

Proposition 3.1 BN-PREDECESSOR is NP-hard for $K = 3$.

Proof. We use a simple polynomial time reduction from 3-SAT [10] to BN-PREDECESSOR (see also **Fig. 3**).

Let $y_1, \ldots, y_N$ be Boolean variables (i.e., 0-1 variables). Let $c_1, \ldots, c_L$ be clauses over $y_1, \ldots, y_N$, where each clause is a disjunction (OR) of at most three literals. It should be noted that a literal is a variable or its negation (NOT). Then, 3-SAT is to ask whether or not there exists an assignment of 0-1 values to $y_1, \ldots, y_N$ which satisfies all the clauses (i.e., the values of all clauses are 1).

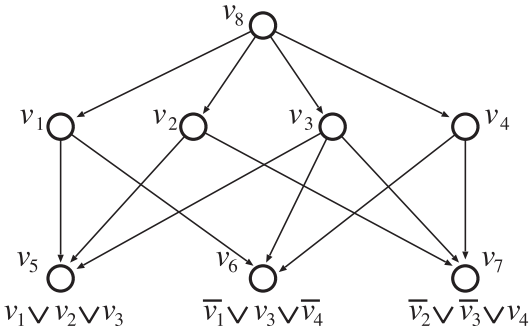


Fig. 3 Example of a reduction from 3-SAT to BN-PREDECESSOR for $K = 3$. An instance of 3-SAT $\{y_1 \vee y_2 \vee y_3, \overline{y_1} \vee y_3 \vee \overline{y_4}, \overline{y_2} \vee \overline{y_3} \vee y_4\}$ is transformed into this Boolean network.

From an instance of 3-SAT, we construct an instance of BN-PREDECESSOR. We let $V = \{v_1, \ldots, v_{N+L+1}\}$, where each v_i for $i = 1, \ldots, N$ corresponds to y_i, each v_{N+j} for $j = 1, \ldots, L$ corresponds to c_j, and v_{N+L+1} is a special node which does not have any input variables and takes always value 1 (i.e., $v_{N+L+1}(t) = 1$ for all t). For each v_i $(i = 1, \ldots, N)$, we assign the following function:

$v_i(t+1) = v_{N+L+1}(t)$.

Let $c_j = g_j(y_{j_1}, y_{j_2}, y_{j_3})$. That is, c_j is a disjunction of literals of y_{j_1}, y_{j_2} and y_{j_3} (e.g., $c_j = y_{j_1} \vee \overline{y_{j_2}} \vee y_{j_3}$). Then, for each v_{N+j} $(j = 1, \ldots, L)$, we assign the following function:

$v_{N+j}(t+1) = g_j(v_{j_1}(t), v_{j_2}(t), v_{j_3}(t))$.

Finally, we let $\mathbf{v}^1 = [1, 1, \ldots, 1]$ (i.e., $v_i(1) = 1$ for all $i = 1, \ldots, N+L+1$).

Then, it is straight-forward to see that $\mathbf{v}^0$ corresponds to a satisfying assignment for 3-SAT. That is, there exists a predecessor $\mathbf{v}^0$ if and only if there exists a satisfying assignment for 3-SAT. Since the above reduction can be done in polynomial time, we have the proposition. □

Coppersmith showed that BN-PREDECESSOR for BNs with maximum indegree K can be reduced to K-SAT in polynomial time. Since it is well known that 2-SAT is solved in linear time [10], BN-PREDECESSOR for $K = 2$ can be solved in linear time.

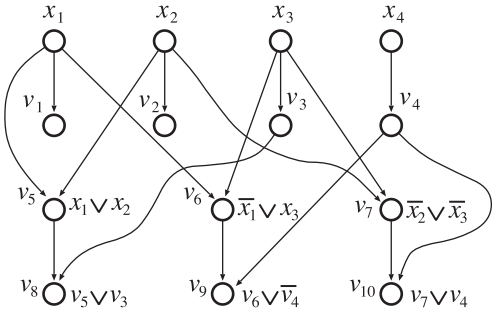


Fig. 4 Example of a reduction from 3-SAT to BN-CONTROL for $K = 2$ and $M = 2$. The same instance of 3-SAT as in Fig. 3 is transformed into this Boolean network.

As for BN-CONTROL, we proved that it is NP-hard even for BNs with $K = 3$ [3)]. Here, we strengthen this hardness result as below.

Theorem 3.2 BN-CONTROL is NP-hard for $K = 2$ and $M \geq 2$.

Proof. We modify the reduction used in the proof of Proposition 3.1 (see also **Fig. 4**).

Let $y_1, \ldots, y_N$ be Boolean variables (i.e., 0-1 variables). Let $c_1, \ldots, c_L$ be a set of clauses over $y_1, \ldots, y_N$. From this instance of 3-SAT, we construct an instance of BN-CONTROL. We let $V = \{v_1, \ldots, v_{N+2L}, x_1, \ldots, x_N\}$ where $x_1, \ldots, x_N$ are external nodes. Each v_i $(i = 1, \ldots, N)$ corresponds to y_i, v_{N+i} and v_{N+L+i} $(i = 1, \ldots, L)$ correspond to c_i, and each x_i corresponds to y_i. Next, we assume without loss of generality that clause c_i is represented as $g_i(f_i(y_{i_1}, y_{i_2}), y_{i_3})$. For example, if $c_i = x \vee \overline{y} \vee z$, we have $f_i(x, y) = x \vee \overline{y}$ and $g_i(w, z) = w \vee z$. Then, for each v_i, we assign the following function:

$$\begin{aligned} v_i(t+1) &= x_i(t), \\ v_{N+i}(t+1) &= f_i(x_{i_1}(t), x_{i_2}(t)), \\ v_{N+L+i}(t+1) &= g_i(v_{N+i}(t), v_{i_3}(t)). \end{aligned}$$

Finally we let $\mathbf{v}^0$ and $\mathbf{v}^M$ as follows.

$$\begin{aligned} v_i^0 &= 0, \quad \text{for all } i, \\ v_i^M &= 1, \quad \text{for } i = 1, \ldots, N, \\ v_{N+i}^M &= f_i(1,1), \quad \text{for } i = 1, \ldots, L, \\ v_{N+L+i}^M &= 1, \quad \text{for } i = 1, \ldots, L. \end{aligned}$$

Now, we show the correctness of the reduction. Suppose that there exists an assignment of Boolean values $b_1, \ldots, b_N$ to $y_1, \ldots, y_N$ which satisfies all clauses $c_1, \ldots, c_L$. Then, for all $i = 1, \ldots, N$, we let

$x_i(0) = b_i$,
$x_i(1) = 1$,

where $x_i(2)$ can be set arbitrary (e.g., $x_i(2) = 1$). Then, we can see that $\mathbf{v}(2) = \mathbf{v}^M$ holds since $v_{N+i}(1) = f_i(b_{i_1}, b_{i_2})$ holds for all $i = 1, \ldots, N$.

Suppose that there exists a control sequence $\langle \mathbf{x}(0), \mathbf{x}(1), \mathbf{x}(2) \rangle$ for which $\mathbf{v}(2) = \mathbf{v}^M$ is satisfied. Then, $x_i(1) = 1$ must hold since $v_i(2) = v_i^M = 1$ holds for $i = 1, \ldots, N$. Moreover, $y_i = x_i(0)$ $(i = 1, \ldots, N)$ must satisfy all clauses c_j since $v_{N+L+i}(2) = g_i(f_i(x_{i_1}(0), x_{i_2}(0)), x_{i_3}(0))$ holds.

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

We can generalize the concept of predecessor to k-predecessors [12)]. $\mathbf{u}$ is called a *k-predecessor* of $\mathbf{v}$ if k times applications of $\mathbf{f}$ to $\mathbf{u}$ yield $\mathbf{v}$. That is, $\mathbf{u}$ is a k-predecessor of $\mathbf{v}$ if

$$\mathbf{v} = \overbrace{\mathbf{f}(\mathbf{f}(\cdots}^{k}(\mathbf{u})\cdots))$$

holds. Clearly, a usual predecessor is equivalent to a 1-predecessor. We define BN-k-PREDECESSOR to be a problem of finding a k-predecessor of a given global state in a given BN. By modifying the reduction in Theorem 3.2, we have:

Theorem 3.3 BN-2-PREDECESSOR is NP-hard even for $K = 2$.

Proof. We modify the reduction in Theorem 3.2. We add one additional node v_0 to V, where v_0 is a constant node such that $v_0(t) = 1$ for all t. For each x_i (which is an internal node in this case), we assign Boolean function $x_i(t+1) = v_0(t)$. Then, we let $\mathbf{v}_{N+i} = f_i(1,1)$ for $i = 1, \ldots, L$ and $\mathbf{v}_i = 1$ for all other $v_i \in V$. Then, it is straight-forward to see that there exists $\mathbf{u}$ such that $\mathbf{v} = \mathbf{f}(\mathbf{f}(\mathbf{u}))$ if and only if 3-SAT has a satisfiable assignment. □

By integrating the above results and the results in [3),5),6),22),24)], we have a clear picture on how computational complexity changes depending on the maximum indegree K for several related problems. **Table 1** summarizes these results where BN-ATTRACTOR denotes the problem of deciding whether or not there exists a singleton attractor in a given BN [24)].

Table 1 Computational complexities of BN-PREDECESSOR and related problems.

	BN-ATTRACTOR	BN-PREDECESSOR	BN-2-PREDECESSOR	BN-CONTROL
$K = 2$	NP-hard [Ref. 22)]	P [Ref. 6)]	NP-hard [this paper]	NP-hard [this paper]
$K \geq 3$	NP-hard [Ref. 2)]	NP-hard [Ref. 5) and this paper]	NP-hard	NP-hard [Ref. 3)]

4. Recursive Algorithms for BN-PREDECESSOR

The predecessor problem can be solved by constructing a state transition diagram or examining all possible global states. However, the number of global states is 2^n and thus it is impossible to construct the diagram unless n is small. Therefore, algorithms which do not examine all possible global states are required and some studies have been done as mentioned in Introduction. Barrett, et al. developed polynomial time algorithms for BN-PREDECESSOR and related problems for networks with bounded tree-width [5]. However, it seems that real networks do not have very small tree-width and thus their algorithms are not practical. Coppersmith showed a polynomial time reduction from BN-PREDECESSOR with maximum indegree K to K-SAT [6]. Since many SAT solvers have been developed, this reduction-based method might be useful. However, it is difficult to perform theoretical analysis on practical SAT solvers. Of course, we can give some guarantees on the worst case time complexity using theoretical algorithms for K-SAT [13),20)], However, such algorithms are complicated and may not be practical. Therefore, it is worthy to develop simple practical algorithms for BN-PREDECESSOR that have some theoretical guarantees on the computational complexity.

In our previous work [24)], we developed a very simple algorithm (called the *basic recursive algorithm*) for identifying all singleton attractors along with its variants. In that algorithm, a partial solution (i.e., a partial global state) is extended one by one according to a given node ordering that leads to a complete solution (i.e., a singleton attractor). If it is found that a partial solution cannot be extended to a complete solution, the next partial solution is examined. The pseudocode of this algorithm is given below [24)], where this procedure is invoked with $m = 1$.

Table 2 Theoretically estimated average case time complexities of basic, outdegree-based, and BFS-based algorithms for the singleton attractor detection problem [24)]. The same results should hold for the modified algorithms for BN-PREDECESSOR.

K	2	3	4	5	6
basic	1.35^n	1.43^n	1.49^n	1.53^n	1.57^n
outdegree-based	1.19^n	1.27^n	1.34^n	1.41^n	1.45^n
BFS-based	1.16^n	1.27^n	1.35^n	1.41^n	1.45^n

Procedure *IdentSingletonAttractor*(v, m)
  if $m = n + 1$
  then Output $v_1(t), v_2(t), \cdots, v_n(t)$, **return**;
  for $b = 0$ **to** 1 **do**
    $v_m(t) := b$;
    if it is found that $f_j(\mathbf{v}(t)) \neq v_j(t)$ for some $j \leq m$
    then continue
    else *IdentSingletonAttractor*$(v, m + 1)$;
  return;

We obtained variants of this algorithm by sorting nodes before invoking the recursive procedure [24)]. In particular, we used the orderings of nodes according to the outdegree and BFS (breadth-first search). For these algorithms, we obtained theoretical estimates of the average case time complexity (**Table 2**) [24)], where some approximations were introduced to obtain these results and thus they are not very exact.

We can modify these algorithms for BN-PREDECESSOR. For that purpose, we only need to modify the part of

  it is found that $f_j(\mathbf{v}(t)) \neq v_j(t)$

to

  it is found that $f_j(\mathbf{v}(t)) \neq \mathbf{v}_j^1$.

It should be noted that the modified algorithms can identify all predecessors and thus it is still useful even for the case of $K = 2$. Since both algorithms are almost identical, the same results as in Table 2 should hold for the modified algorithms.

Here we briefly give an analysis only for the basic recursive algorithm, which

is almost identical to that in 24). Assume that we have tested the first m out of n nodes, where $m \geq K$. For all $j \leq m$, $f_j(\mathbf{v}(t)) \neq \mathbf{v}_j^1$ holds with probability

$$P(f_j(\mathbf{v}(t)) \neq \mathbf{v}_j^1) \;=\; 0.5 \cdot \frac{\left(\begin{array}{c} m \\ k_j \end{array} \right)}{\left(\begin{array}{c} n \\ k_j \end{array} \right)} \;\approx\; 0.5 \cdot \left(\frac{m}{n}\right)^{k_j} \;\geq\; 0.5 \cdot \left(\frac{m}{n}\right)^{K},$$

where we assume that Boolean functions of k_j ($< K$) inputs are selected at uniformly random. If $f_j(\mathbf{v}(t)) \neq \mathbf{v}_j^1$ is found for some $j \leq m$, the algorithm cannot proceed to the next recursive level. Therefore, the probability that the algorithm examines the $(m+1)$th node is no more than

$$[1 - P(f_j(\mathbf{v}(t)) \neq \mathbf{v}_j^1)]^m \;\approx\; \left[1 - 0.5 \cdot \left(\frac{m}{n}\right)^K\right]^m.$$

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

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

Here we let $s = \frac{m}{n}$ and

$F(s) = [2^s \cdot (1 - 0.5 \cdot s^K)^s]^n = [(2 - s^K)^s]^n$.

Then, the average case time complexity is estimated by computing the maximum value of $(2 - s^K)^s$ since the other factors can be ignored. By simple numerical calculations, we obtain the results shown in the first row of Table 2.

In order to verify these theoretical estimates, we performed computational experiments. We examined the basic algorithm and the outdegree-based algorithm for BN-PREDECESSOR with $K = 3$ and $K = 4$. For each K and n, we took the average CPU time over 100 random $\mathbf{v}^1$'s over 100 random BNs, where these computational experiments were performed on a LINUX PC with Xeon 3 GHz Dual-Core CPUs and 8 GB RAM. We verified through these computational experiments that the results similar to Table 2 hold. The results are shown in **Fig. 5** and **Fig. 6**. There are some gaps between theoretical estimates and experimental results. However, it is to be noted that the complexities of Table 2 were obtained for large n (by introducing some approximations) whereas only small size networks were examined in our experiment. Thus, it seems that gaps will become smaller as n grows. It is also seen from Fig. 5 and Fig. 6 that our proposed algorithms are much faster than the naive algorithm that examines all possible states (2^n states).

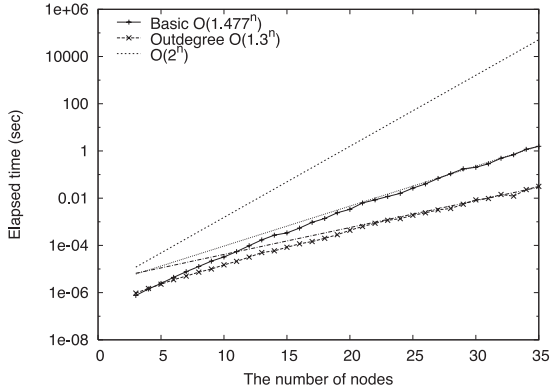


Fig. 5 Elapsed time (in seconds) by the proposed algorithms for random Boolean networks with $K = 3$.

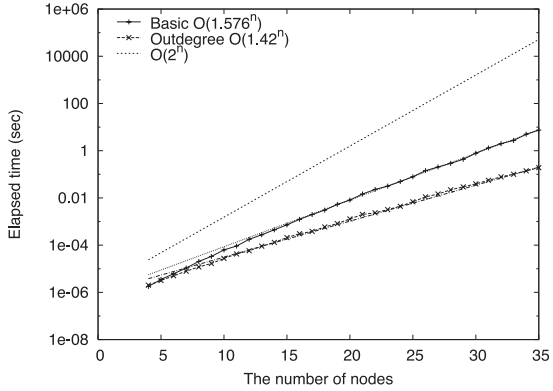


Fig. 6 Elapsed time (in seconds) by the proposed algorithms for random Boolean networks with $K = 4$.

5. Results on Distribution of Predecessors

We have studied so far the complexity issue on BN-PREDECESSOR. In this section, we study distribution of predecessors. First, we present an almost trivial result.

Proposition 5.1 For any Boolean network, the average number of predecessors per global state is 1.

Proof. The number of outgoing edges from each node in a state transition diagram is 1. Thus, the total number of edges in a state transition diagram is 2^n where n is the number of nodes in BN. Since there exist 2^n global states, the average number of incoming edges (i.e., predecessors) is 1. □

Based on this result, we expected that recursive application of BN-PREDECESSOR leads to a fast algorithm for BN-CONTROL. However, we empirically found that there exist many predecessors if it exists and thus the above attempt was not successful. This motivated us to study distribution of predecessors.

We begin with BNs with no constraint since it is easier to analyze. Coppersmith showed that the expected number of global states with no predecessors converges to $2^n/e$ as n grows [6], where all possible BNs with n nodes are randomly given and e is the base of natural log. This can be shown as follows. Since each global state has exactly one outgoing edge, the probability that $\mathbf{u}$ is a predecessor of $\mathbf{v}$ is $1/N$ for each $\mathbf{u}$, where $N = 2^n$. Thus, the probability that each global state $\mathbf{v}$ has no predecessor is $(1 - \frac{1}{N})^N$, which approaches to $1/e$ as N grows.

We extend this result for distribution of 2-predecessors. That is, we estimate the number of global states $\mathbf{v}$ for which there is no $\mathbf{w}$ such that $\mathbf{v} = \mathbf{f}(\mathbf{f}(\mathbf{w}))$. First, we consider the probability that there exists only one $\mathbf{u}$ such that $\mathbf{v} = \mathbf{f}(\mathbf{u})$ and $\mathbf{u}$ has no predecessor. This probability is calculated by

$$(N-1) \cdot \left(\frac{1}{N}\right) \cdot \left(1 - \frac{2}{N}\right)^{N-1}$$

since $\mathbf{u}$ must be different from $\mathbf{v}$ (otherwise $\mathbf{v} = \mathbf{f}(\mathbf{v}) = \mathbf{f}(\mathbf{f}(\mathbf{v})) = \cdots$ would hold) and global states except $\mathbf{u}$ should not go to $\mathbf{u}$ or $\mathbf{v}$. Next, we consider the probability that there exist exactly two predecessors of $\mathbf{v}$ and none of them has a predecessor. This probability is calculated by

$$\left(\begin{array}{c} N-1 \\ 2 \end{array}\right) \cdot \left(\frac{1}{N}\right)^2 \cdot \left(1 - \frac{3}{N}\right)^{N-2}.$$

In this way, the number of global states $\mathbf{v}$ for which there is no $\mathbf{w}$ such that $\mathbf{v} = \mathbf{f}(\mathbf{f}(\mathbf{w}))$ is calculated by

$$\sum_{i=1}^{N-2} \left(\begin{array}{c} N-1 \\ i \end{array}\right) \cdot \left(\frac{1}{N}\right)^i \cdot \left(1 - \frac{i+1}{N}\right)^{N-i}.$$

This probability is approximated to $e^{-1}(e^{e^{-1}} - 1)$ since

$$\left(\begin{array}{c} N-1 \\ i \end{array}\right) \cdot \left(\frac{1}{N}\right)^i \cdot \left(1 - \frac{i+1}{N}\right)^{N-i} \approx \frac{e^{-(i+1)}}{i!}$$

holds for small i. Therefore, the number of global states having no 2-predecessor is approximated by $Ne^{-1}(e^{e^{-1}} - 1)$.

It is reasonable to try to extend this result for $\mathbf{v} = \mathbf{f}^k(\mathbf{w})$. However, it seems quite difficult to do so even for $k = 3$.

Now, we consider BNs with maximum indegree K. Coppersmith showed that the probability that a randomly chosen global state has a predecessor is bounded by $(1-2^{-2^K})^n$, which approaches to 0 as n grows [6]. This bound can be shown as follows. Suppose that $\mathbf{v} = [1, 1, \cdots, 1]$. If the Boolean function always outputting 0 is assigned to some node, $\mathbf{v}$ has no predecessor. Since there are 2^{2^K} Boolean functions with K inputs, the probability that such a function is assigned to none of the nodes is $(1 - 2^{-2^K})^n$, where we assume that a Boolean function with K inputs is assigned to each node uniformly at random.

Based on her idea, we estimate a lower bound of the expected number of predecessors for a global state having at least one predecessor. Hereafter, we let $H = 2^{2^K}$, and we call a node for which a constant Boolean function (i.e., a function always outputting 1 or always outputting 0) is assigned a *constant node*.

Proposition 5.2 Suppose that for each node, K input nodes are randomly selected and then a Boolean function is randomly selected from 2^{2^K} possible Boolean functions (including constant Boolean functions). Then, the expected number of global states having no predecessor is greater than

$$2^n \cdot \left(\frac{2^L - 1}{2^L}\right)$$

where $L = \frac{n}{2H} = \frac{n}{2^{2^K+1}}$.

Proof. Since a constant function is assigned to each node with probability $\frac{2}{H}$, the expected number of constant nodes is $\mu = 2n/H$. Here, we use the Chernoff bound [17], which states that the probability that the sum of independent Poisson trails with mean value μ is less than $(1-\delta)\mu$ is less than $\exp(-\mu\delta^2/2)$. By setting $\delta = 3/4$, the probability that the number of constant nodes is less than $\mu/4 = n/2H$ is

$$\exp\left(-\mu\left(\frac{3}{4}\right)^2/2\right) \;=\; \exp\left(-\frac{9n}{16H}\right).$$

Consider the case where the number of constant nodes is at least $n/2H$. We can assume without loss of generality that the constant function 0 is assigned to each of the first $L = n/2H$ nodes. Then, $\mathbf{v}$ has no predecessors if $\mathbf{v}_i = 1$ holds for at least one i in $\{1, 2, \ldots, L\}$. Thus, in this case, the number of global states having no predecessor is at least $2^n \cdot \left(\frac{2^L-1}{2^L}\right)$.

Such a case occurs with probability at least $1 - \exp\left(-\frac{9n}{16H}\right)$. Therefore, the expected number of global states having no predecessor is at least

$$\left(1 - e^{-\frac{9n}{16H}}\right) \cdot 2^n \cdot \left(\frac{2^L - 1}{2^L}\right) \;>\; 2^n \cdot \frac{2^L - 2}{2^L}$$

for sufficiently large n because

$$\begin{aligned}\left(1 - e^{-\frac{9n}{16H}}\right) \cdot \left(2^{n/2H} - 1\right) &= 2^{n/2H} - 1 + e^{-\frac{9n}{16H}} - e^{-\frac{9n}{16H}} \cdot 2^{n/2H} \\ &> 2^{n/2H} - 1 - e^{-\frac{9n}{16H}} \cdot 2^{n/2H} \\ &> 2^{n/2H} - 1 - \left(\frac{2^{1/2}}{e^{9/16}}\right)^{n/H} \\ &> 2^{n/2H} - 2\end{aligned}$$

holds for $n > H$. □

Therefore, the expected number of global states having predecessors is at most $\frac{2 \cdot 2^n}{2^L}$. Thus, once a global state has a predecessor, it is expected to have $\frac{2^L}{2} \;=\; 2^{(n/(2^{2^K+1}))-1}$ or more predecessors.

Proposition 5.3 Suppose that for each node, K input nodes are randomly selected and then a Boolean function is randomly selected from 2^{2^K} possible Boolean functions. Then, once a global state has a predecessor, it is expected to have $2^{n/(2^{(2^K+1)}-1)}$ or more predecessors.

It should be noted that the above number becomes very large as n grows. Thus, this estimation explains the reason why recursive application of BN-PREDECESSOR does not lead to a fast algorithm for BN-CONTROL.

6. An Improved Algorithm for BN-CONTROL

As mentioned in Sections 1 and 2, BN-CONTROL is NP-hard but can be solved in $O(nM2^{m+n})$ time for BNs of bounded indegree. Though some practically faster algorithm was proposed [15], no improvement has been done on the theoretical computational complexity. Here, we show an improved algorithm for BN-CONTROL whose average case time complexity is $O(nM2^{m+\beta n})$ for BNs of bounded indegree K, where β (< 1) depends on K.

The idea of the improved algorithm is quite simple but non-trivial. As shown in Section 5, we can assume without loss of generality that the constant function 0 is assigned to each of the first $n/2H$ nodes with high probability. Then, we can ignore these nodes and thus we can only consider $2^{n-\frac{n}{2H}}$ states for internal nodes, instead of 2^n states.

As in Section 5, we let $L = n/2H$. For a global state $\mathbf{v}$, $\mathbf{v_0}$ denotes the global state defined by

$$[\overbrace{0, 0, \ldots, 0}^{L}, \mathbf{v}_{L+1}, \mathbf{v}_{L+2}, \ldots, \mathbf{v}_n].$$

Then, the following proposition follows from the definition of $\mathbf{v_0}$.

Proposition 6.1 Suppose that the constant function 0 is assigned to each of the first L nodes in a BN with external nodes. Then, $\mathbf{f}(\mathbf{v}, \mathbf{x}) = \mathbf{f}(\mathbf{v_0}, \mathbf{x})$ holds for all $\mathbf{v}$.

Based on this proposition, we can replace $D[b_1, \ldots, b_n, t]$ in the original dynamic programming algorithm with $D'[c_1, \ldots, c_{n-L}, t]$. For a 0-1 vecter $\mathbf{c} = [c_1, \ldots, c_{n-L}]$ of length $n - L$, we let $\mathbf{0} \cdot \mathbf{c}$ denote the 0-1 vector of length n defined by

$[\overbrace{0,0,\ldots,0}^{L},c_1,c_2,\ldots,c_{n-L}]$.

Then, the recurrence for the improved dynamic programming algorithm is given by:

$$D'[c_1,\ldots,c_{n-L},M] = \begin{cases} 1, & \text{if } \mathbf{0}\cdot\mathbf{c} = \mathbf{v}^M, \\ 0, & \text{otherwise,} \end{cases}$$

$$D'[c_1,\ldots,c_{n-L},t-1] = \begin{cases} 1, & \text{if there exists } (\mathbf{a},\mathbf{x}) \text{ such that} \\ & D'[a_1,\ldots,a_{n-L},t] = 1 \text{ and } \mathbf{0}\cdot\mathbf{a} = \mathbf{f}(\mathbf{0}\cdot\mathbf{c},\mathbf{x}), \\ 0, & \text{otherwise,} \end{cases}$$

It is straight-forward to see the correctness of the improved algorithm. In this algorithm, we assumed that there exist at least $L = \frac{n}{2H}$ constant nodes. However, in some cases, we may have much smaller number of constant nodes. Thus, when the number of constant nodes is less than $\frac{n}{2H}$, we use the original dynamic programming algorithm. By using this combination, we have the following.

Theorem 6.2 Suppose that for each node, K input nodes are randomly selected and then a Boolean function is randomly selected from 2^{2^K} possible Boolean functions. Then, BN-CONTROL for bounded indegree K can be solved in $O(nM2^{m+(1-(1/2^{(2^K+1)}))n})$ time on the average.

Proof. We bound the probability P_1 that the number of constant nodes is less than $n/2H$ (recall $H = 2^{2^K}$). Since a constant function is assigned to each of n nodes with probability $2/H$ independently, the expected number of constant nodes is $\mu = 2n/H$. By setting $\delta = 3/4$ in the Chernoff bound, we have

$$P_1 \;<\; \exp\left(-\frac{2n}{H}\cdot\left(\frac{3}{4}\right)^2\cdot\frac{1}{2}\right) \;=\; \exp\left(-\frac{9n}{16H}\right).$$

Therefore, the average case time complexity is bounded by

$$\begin{aligned} & O\left(nM2^m\cdot\left(e^{-\frac{9n}{16H}}\cdot 2^n + (1-e^{-\frac{9n}{16H}})\cdot 2^{n-\frac{n}{2H}}\right)\right) \\ =\; & O\left(nM2^m\cdot\left(e^{-\frac{9n}{16H}}\cdot 2^n + 2^{n-\frac{n}{2H}}\right)\right) \\ =\; & O\left(nM2^m2^n\cdot\left((e^{-9/16})^{n/H} + (2^{-1/2})^{n/H}\right)\right) \\ =\; & O\left(nM2^m2^n\cdot\left((2^{-1/2})^{n/H}\right)\right) \\ =\; & O(nM2^{m+(1-(1/2H))n}). \end{aligned}$$

□

Even for the case of $K = 2$, the bound given by Theorem 6.2 is $O(nM2^{m+0.969n})$ and thus the improvement is quite small. However, it is the first result (even in the average case) that breaks $O(2^n)$ factor.

The result of Theorem 6.2 might be slightly improved by adjusting L and δ carefully. However, the anlysis would be much harder.

7. Concluding Remarks

In this paper, we studied BN-PREDECESSOR, its variant, and BN-CONTROL with focusing on bounded indegree cases. We showed some hardness results, and presented simple practical algorithms for BN-PREDECESSOR. We also analyzed distribution of predecessors, which leaded to an improved algorithm for BN-CONTROL and also explained why recursive application of BN-PREDECESSOR does not lead to a fast algorithm for BN-CONTROL. However, BN-PREDECESSOR is still useful for BN-CONTROL since it can be used for pre-processing. If it is found that there exists no predecessor, we can conclude that there exists no feasible control actions leading to the desired state.

We developed algorithms for BN-PREDECESSOR, based on our previous algorithms for identification of attractors [24]. Though these algorithms are simple and practical to some extent, better algorithms would be obtained by using reduction to SAT or by using techniques employed in other methods for identification of attractors. In particular, it might be possible to develop randomized expected polynomial time algorithms for BN-PREDECESSOR since such algorithms are known for K-SAT problems [1]. Therefore, development of randomized expected polynomial time algorithms for BN-PREDECESSOR is left as a future work.

We presented an improved algorithm for BN-CONTROL. However, this improved algorithm is based on the assumption that all Boolean functions, which include constant functions, are assigned uniformly at random. However, constant

functions may not appear so frequently in real networks. Therefore, average case analysis in more realistic situations is an important future work. Of course, improvement of the worst case time complexity for BN-CONTROL is another important future work.

Though BN may be too simple as a model of genetic networks, studies on BN may provide some insights to other models. As least, negative results should hold for more general models. Some ideas in positive results may also be useful for designing algorithms for more general models. Therefore, extension of the proposed algorithms for more general models is also an important future work.

Acknowledgments TA was partially supported by a Grant-in-Aid "Systems Genomics" from MEXT and by the Cell Array Project from NEDO, Japan. WKC was partially supported by HK RCG Grant No.7017/07P and HKU CRCG grants. MKN was partially supported by RGC 7046/03P, 7035/04P, 7035/ 05P and HKBU FRGs.

References

1) Achlioptas, D., Naor, A. and Peres, Y.: Rigorous location of phase transitions in hard optimization problems, *Nature*, Vol.435, pp.759–764 (2005).
2) Akutsu, T., Kuhara, S., Maruyama, O. and Miyano, S.: A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions, *Genome Informatics*, Vol.9, pp.151–160 (1998).
3) Akutsu, T., Hayashida, M., Ching, W-K. and Ng, M.K.: Control of Boolean networks: Hardness results and algorithms for tree-structured networks, *J. Theoret. Biol.*, Vol.244, pp.670–679 (2007).
4) Akutsu, T., Hayashida, M. and Tamura, T.: Algorithms for inference, analysis and control of Boolean networks, *Lect. Notes Comput. Sci.*, Vol.5147, pp.1–15 (2008).
5) Barrett, C., Hunt III, H.B., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E. and Thakur, M.: Predecessor existence problems for finite discrete dynamical systems, *Theoret. Comput. Sci.*, Vol.386, pp.3–37 (2007).
6) Coppersmith, S.N.: Complexity of the predecessor problem in Kauffman networks, *Phys. Rev. E*, Vol.75, 051108 (2007).
7) Datta, A., Choudhary, A., Bittner, M.L. and Dougherty, E.R.: External control in Markovian genetic regulatory networks, *Machine Learning*, Vol.52, pp.169–191 (2003).
8) Devloo, V., Hansen, P. and Labbé, M.: Identification of all steady states in large networks by logical analysis, *Bull. Math. Biol.*, Vol.65, pp.1025–1051 (2003).
9) Drossel, B., Mihaljev, T. and Greil, F.: Number and length of attractors in a critical kauffman model with connectivity one, *Phys. Rev. Lett.*, Vol.94, 088701 (2005).
10) Garey, M.R. and Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York (1979).
11) Garg, A., Xenarios, I., Mendoza, L. and DeMicheli, G.: An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments, *Lect. Notes Bioinfomatics*, Vo.4453, pp.62–76 (2007).
12) Irons, D.J.: Improving the efficiency of attractor cycle identification in Boolean networks, *Physica D*, Vol.217, pp.7–21 (2006).
13) Iwama, K. and Tamaki, S.: Improved upper bounds for 3-SAT, *Proc. 15th ACM-SIAM Symp. Discrete Algoriths*, pp.328–329, ACM Press (2004).
14) Kauffman, S.A.: *The Origins of Order: Self-organization and Selection in Evolution*, Oxford Univ. Press, New York (1993).
15) Langmead, C.J. and Jha, S.K.: Symbolic approaches for finding control strategies in Boolean networks, *Proc. 6th Asia-Pacific Bioinformatics Conference*, pp.307–319, Imperial College Press (2008).
16) Milano, M. and Roli, A.: Solving the satisfiability problem through Boolean networks, *Lect. Notes Art. Intell.*, Vol.1792, pp.72–93 (2000).
17) Motwani, R. and Raghavan, P.: *Randomized Algorithms*, Cambridge University Press, New York (1995).
18) Pal, R., Datta, A., Bittner, M.L. and Dougherty, E.R.: Intervention in context-sensitive probabilistic Boolean networks, *Bioinformatics*, Vol.21, pp.1211–1218 (2005).
19) Pal, R., Ivanov, I., Datta, A., Bittner, M.L. and Dougherty, E.R.: Generating Boolean networks with a prescribed attractor structure, *Bioinformatics*, Vol.21, pp.4021–4025 (2005).
20) Rolf, D.: Improved bound for the PPSZ/Schöning-Algorithm for 3-SAT, *J. Satisfiability, Boolean Modeling and Computation*, Vol.1, pp.111–122 (2006).
21) Samuelsson, B. and Troein, C.: Superpolynomial growth in the number of attractors in kauffman networks, *Phys. Rev. Lett.*, Vol.90, 098701 (2003).
22) Tamura, T. and Akutsu, T.: Detecting a singleton attractor in a Boolean network utilizing SAT algorithms, *IEICE Trans. Fundamentals.*, to appear.
Preliminary version has appeared as: An $O(1.787^n)$-time algorithm for detecting a singleton attractor in a Boolean network consisting of AND/OR nodes, *Lect. Notes Comput. Sci.*, Vol.4639, pp.494–505 (2007).
23) Wuensche, W.: Basins of attraction in cellular automata, *Complexity*, Vol.5, pp.19–25 (2001).
24) Zhang, S-Q., Hayashida, M., Akutsu, T., Ching, W-K. and Ng, M.K.: Algorithms for finding small attractors in Boolean networks, *EURASIP J. Bioinform. Syst. Biol.*, Vol.2007, 20180 (2007).

(Received June 11, 2008)
(Accepted July 7, 2008)
(Released November 28, 2008)

(Communicated by *Tetsuo Shibuya*)

Tatsuya Akutsu received his M.Eng. degree in Aeronautics in 1996 and a Dr.Eng. degree in Information Engineering in 1989 both from the University of Tokyo, Japan. From 1989 to 1994, he was with Mechanical Engineering Laboratory, Japan. He was an associate professor in Gunma University from 1994 to 1996 and in Human Genome Center, the University of Tokyo from 1996 to 2001 respectively. He joined Bioinformatics Center, Institute for Chemical Research, Kyoto University, Japan as a professor in Oct. 2001. His research interests include bioinformatics and discrete algorithms. He is a member of ACM, IEICE, ISCB and JSBi.

Morihiro Hayashida received his M.Sci. degree in Information Science from the University of Tokyo in 2002 and his Ph.D. degree in Informatics from Kyoto University in 2005. He is currently an assistant professor in Bioinformatics Center, Institute for Chemical Research, Kyoto University. His current research interests include issues related to protein function prediction and bioinformatics.

Shu-Qin Zhang received her Master degree in Operations Research and Control Theory from Beijing Institute of Technology, China in 2003 and the Ph.D. degree in applied mathematics from the University of Hong Kong, Hong Kong in 2007. She then joined School of Mathematical Sciences, Fudan University, China as an assistant professor. Her main research interests include bioinformatics and scientific computing.

Wai-Ki Ching is a lecturer at the Department of Mathematics, the University of Hong Kong. He obtained his B.Sc. (1991) and M. Phil. (1994) degrees in Mathematics from the University of Hong Kong. He then obtained his Ph.D. degree in Systems Engineering and Engineering Management (1998) from the Chinese University of Hong Kong. He was a visiting post-doc fellow at the Judge Business School of the Cambridge University (1999–2000), and was a lecturer at the University of Southampton (2000–2001). He has published over 150 papers in refereed journals, book chapters and conference proceedings. His research interests include data modelling, optimization algorithms, systems engineering and bioinformatics.

Michael K. Ng is a Professor in the Department of Mathematics at the Hong Kong Baptist University. He obtained his B.Sc. degree in 1990 and M.Phil. degree in 1992 at the University of Hong Kong, and Ph.D. degree in 1995 at Chinese University of Hong Kong. As an applied mathematician, Michael's main research areas include Bioinformatics, Data Mining, Operations Research and Scientific Computing. Michael has published and edited 5 books, published more than 160 journal papers. He has reviewed papers for more than 40 international journals. He currently serves on the editorial boards of several international journals.