Algorithms for Inference, Analysis and Control of Boolean Networks

Tatsuya Akutsu, Morihiro Hayashida, and Takeyuki Tamura

Bioinformatics Center, Institute for Chemical Research, Kyoto University
Gokasho, Uji, Kyoto 611-0011, Japan
{takutsu,morihiro,tamura}@kuicr.kyoto-u.ac.jp

Abstract. Boolean networks (BNs) are known as a mathematical model of genetic networks. In this paper, we overview algorithmic aspects of inference, analysis and control of BNs while focusing on the authors' works. For inference of BN, we review results on the sample complexity required to uniquely identify a BN. For analysis of BN, we review efficient algorithms for identifying singleton attractors. For control of BN, we review NP-hardness results and dynamic programming algorithms for general and special cases.

1 Introduction

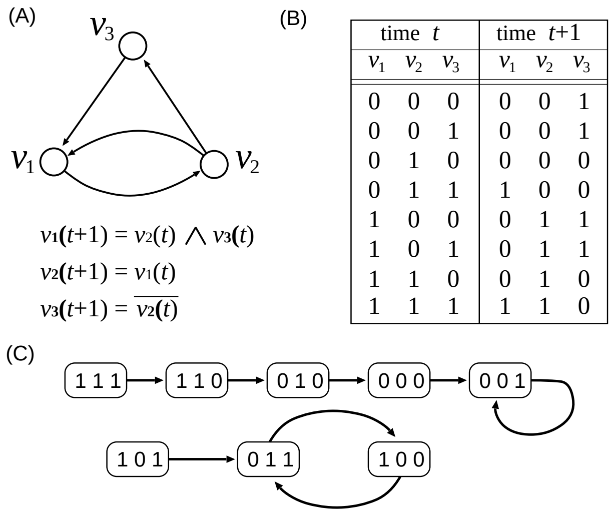
Mathematical analysis of biological networks is an important topic in bioinformatics, systems biology, and algebraic biology. For that purpose, various kinds of mathematical models have been proposed. Among them, the *Boolean network* (BN, in short) model has received much attention [16] as a model of genetic networks. 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. Though various aspects of BNs have been studied, this paper focuses on the following three problems.

Inference of BN: Given a part of the state transition table (which corresponds to time series data of gene expression), infer a BN that is consistent with given data.

Identification of Attractors: Given a BN, identify all attractors where attractors correspond to steady-states.

Control of BN: Given a BN with control nodes, its initial and target states, find a sequence of 0-1 vectors for control nodes which leads BN from the initial state to the target state.

These problems are considered to be fundamental and are interesting from an algorithmic viewpoint. The purpose of this review paper is not to give a comprehensive survey, but to explain the key ideas and proofs in algorithmic and mathematical results mainly obtained by the authors.



time t			time t+1		
v_1	v_2	v_3	v_1	v_2	v_3
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	1	1	0

Fig. 1. Example of a Boolean network. Dynamics of BN (A) is well-described by a state transition table (B) and by a state transition diagram (C).

2 Boolean Network

In this section, we briefly review BN [16].

A BN is represented by a set of *nodes* and a set of regulation rules for nodes, where each node corresponds to a gene if BN is regarded as a model of a genetic network. Each node takes either 0 or 1 at each discrete time t, where 1 (resp. 0) means that the corresponding gene is expressed (resp. not expressed) at time t. A regulation rule for each node is given in the form of a Boolean function and the states of nodes change synchronously. An example is given in Fig. 1. In this example, the state of node v_1 at time $t+1$ is determined by the logical AND of the states of nodes v_2 and v_3 at time t. The states of node v_2 and v_3 at time $t+1$ are determined by the state of node v_1 and the logical NOT of the state of node v_2 at time t, respectively. We use $x \wedge y$, $x \vee y$, $x \oplus y$, $\overline{x}$ to denote logical AND of x and y, logical OR of x and y, exclusive OR of x and y, and logical NOT of x, respectively. Dynamics of a BN is well-described by a state transition table and a state transition diagram shown in Fig. 1. For example, the third row of the table means that if the state of BN is $[0,1,0]$ at time t then the state will be $[0,0,0]$ at time $t+1$, and the arc from 111 to 110 in the diagram means that if the state of BN is $[1,1,1]$ at time t the state will be $[1,1,0]$ at time $t+1$.

Now we will give a formal definition of BN. A *Boolean network* $G(V,F)$ consists of a set $V = \{v_1,\ldots,v_n\}$ of nodes and a list $F = (f_1,\ldots,f_n)$ of *Boolean functions*, where a Boolean function $f_i(v_{i_1},\ldots,v_{i_k})$ with inputs from specified nodes $v_{i_1},\ldots,v_{i_k}$ is assigned to each node v_i. We use $IN(v_i)$ to denote the set of input nodes $v_{i_1},\ldots,v_{i_k}$ to v_i. Each node takes either 0 or 1 at each discrete time t, and the state of node v_i at time t is denoted by $v_i(t)$. Then, the state of node v_i at time $t+1$ is determined by

$$v_i(t+1) = f_i(v_{i_1}(t), \dots, v_{i_{k_i}}(t)).$$

Here we let $\mathbf{v}(t) = [v_1(t), \dots, v_n(t)]$, which is called a *Gene Activity Profile* (GAP) at time t. 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. We define the set of edges E by $E = \{(v_{i_j}, v_i) | v_{i_j} \in IN(v_i)\}$. Then, $G(V, E)$ is a directed graph representing the network topology of a BN. It is worthy to mention that an edge from v_{i_j} to v_i means that v_{i_j} directly affects expression of v_i. The number of input nodes to v_i is called the *indegree* of v_i. We use K to denote the *maximum indegree* of a BN, which plays an important role in both inference and analysis of BNs.

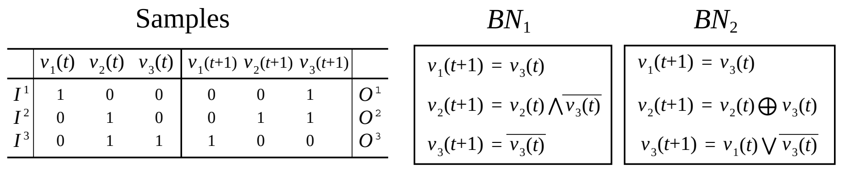
Though BNs are deterministic, many real biological systems contain stochastic subsystems. Thus, several probabilistic extensions of BN have been proposed, which include Noisy Boolean Networks [3] and Probabilistic Boolean Networks (PBNs) [28]. Since this paper focuses on BNs, readers interested in these models are referred to [3, 8, 25, 26, 28].

3 Inference of Boolean Networks

Due to the development of DNA microarray technology, it has been made possible to observe time series data of expression of several thousands of genes, and thus extensive studies have been done for inferring genetic networks using these time series data. In order to infer genetic networks, mathematical models of genetic networks are usually required. In 1998, Liang et al. developed the REVEAL algorithm for inference of BNs from gene expression data [20]. Independently, Akutsu et al. studied in 1998 algorithmic strategies for identification of Boolean-like networks using gene disruption and gene overexpression [4]. By combining these two, Akutsu et al. derived a fundamental result on the number of samples (gene expression profiles) that are required to uniquely identify a BN [1]. In this section, we review this result of the sample complexity along with some algorithmic issues.

3.1 Problem Definition and Simple Inference Algorithm

Let (I^j, O^j) $(j = 1, \dots, m)$ be a pair of expression profiles (i.e., 0-1 vectors) of $v_1, \dots, v_n$, where I^j corresponds to a GAP at time t and O^j corresponds to a GAP at time $t+1$. I_i^j (resp. O_i^j) denotes the expression (0 or 1) of gene v_i in I^j (resp. O^j). Each pair (I^j, O^j) is called a *sample*. We say that a node v_i in a BN $G(V, F)$ is *consistent* with a sample (I^j, O^j) if $O_i^j = f_i(I_{i_1}^j, \dots, I_{i_k}^j)$ holds. We say that $G(V, F)$ is *consistent* with (I^j, O^j) if all nodes are consistent with (I^j, O^j). For a set of samples $EX = \{(I^1, O^1), (I^2, O^2), \dots, (I^m, O^m)\}$, we say that $G(V, F)$ (resp. node v_i) is *consistent* with EX if $G(V, F)$ (resp. node v_i) is consistent with all (I^j, O^j) for $1 \leq j \leq m$. Then, the inference problem and the identification problem are defined as follows [1].


Samples
BN_1
BN_2

	$v_1(t)$	$v_2(t)$	$v_3(t)$	$v_1(t+1)$	$v_2(t+1)$	$v_3(t+1)$	
I^1	1	0	0	0	0	1	O^1
I^2	0	1	0	0	1	1	O^2
I^3	0	1	1	1	0	0	O^3

BN_1:

$v_1(t+1) = v_3(t)$

$v_2(t+1) = v_2(t) \wedge \overline{v_3(t)}$

$v_3(t+1) = \overline{v_3(t)}$

BN_2:

$v_1(t+1) = v_3(t)$

$v_2(t+1) = v_2(t) \oplus v_3(t)$

$v_3(t+1) = v_1(t) \vee \overline{v_3(t)}$

Fig. 2. Inference of BN. In this example, BNs consistent with given samples are not determined uniquely because both BN_1 and BN_2 are consistent with samples.

Definition 1. **[Inference of BN]**
Instance: *the number of nodes n and a set of samples $EX = \{(I^j, O^j) \mid j = 1, \ldots, m\}$,*
Problem: *decide whether or not there exists a BN of n nodes consistent with EX and output one if it exists.*

Definition 2. **[Identification of BN]**
Instance: *the number of nodes n and a set of samples $EX = \{(I^j, O^j) \mid j = 1, \ldots, m\}$,*
Problem: *decide whether or not there exists a unique BN of n nodes consistent with EX and output it if it exists.*

It is to be noted that both problems are very similar: the difference lies only in a point that the uniqueness of the solution should be verified in the identification problem.

3.2 Upper and Lower Bounds on Sample Complexity

To study the sample complexity, we consider the following quite simple algorithm for inference of BNs: for each node v_i, we generate all possible Boolean functions f_i and output a function that satisfies $O_i^j = f_i(I_{i_1}^j, \ldots, I_{i_k}^j)$ for all $j = 1, \ldots, m$. If there is no restriction, it is well known that the number of possible Boolean functions for each v_i is 2^{2^n}. If the maximum indegree is bounded by K, the number of possible Boolean functions (along with possible sets of input nodes) is at most $\binom{n}{K} 2^{2^K}$, which is a polynomial of n if K can be regarded as a constant.

Now we analyze the sample complexity: the number of samples that are required to uniquely identify a BN. It is known that BNs correspond to state-transition tables in the one-to-one manner. This means that all rows of a state-transition table are required in order to uniquely specify a BN.

Proposition 1. [1] *2^n samples are required to uniquely identify a BN if there is no restriction on a BN.*

This proposition suggests that an exponential number of gene expression profiles are required, which is non-realistic. However, if the maximum indegree

Table 1. Explanation of Proposition 3. For $K = 1$, EX_1 satisfies the condition of the proposition, whereas EX_2 does not satisfy since $[v_2 = 0, v_3 = 0]$ does not appear.

	EX_1				EX_2			
	v_1	v_2	v_3	v_4	v_1	v_2	v_3	v_4
I_1	0	1	1	1	1	1	0	0
I_2	1	0	1	1	0	0	1	1
I_3	1	1	0	1	1	0	1	0
I_4	1	1	1	0	0	1	0	1
I_5	0	0	0	0	1	1	1	1

is bounded, the situation drastically changes. First, we show a lower bound on the sample complexity.

Proposition 2. [1] *$\Omega(2^K + K \log n)$ samples are necessary in the worst case to identify a BN of maximum indegree K.*

Proof. We consider the number of mutually distinct Boolean networks. Since there are $\Omega(n^K)$ possible combinations of input nodes and 2^{2^K} possible Boolean functions per node, there are $\Omega((2^{2^K} \cdot n^K)^n)$ BNs whose maximum indegree is K.[1] Therefore, $\Omega(2^K n + nK \log n)$ bits are required to represent a BN. On the other hand, each sample gives information quantity of n bits. Therefore, $\Omega(2^K + K \log n)$ samples are required in the worst case. $\square$

Next, we show an upper bound of the sample complexity. For that purpose, we need the following proposition (see also Table 1).

Proposition 3. [4] *If all assignments (i.e., 2^{2K} assignments) of Boolean values to all subsets of V with $2K$ nodes (i.e., $\binom{n}{2K}$ subsets) appear in I^js, the Boolean function together with input nodes for each node is determined uniquely, if it exists.*

Theorem 1. [1] *If $O(2^{2K} \cdot (2K + \alpha) \cdot \log n)$ samples (i.e., I^js) are given uniformly randomly, the following holds with probability at least $1 - \frac{1}{n^\alpha}$: there exists at most one BN of n nodes with maximum indegree $\leq K$ which is consistent with given samples.*

Proof. We derive the number of I^js satisfying the condition of Proposition 3. For that purpose, we consider the probability that the condition is not satisfied when m random I^js are given.

For any fixed set of nodes $\{v_{i_1}, \ldots, v_{i_{2K}}\}$, the probability that a sub-assignment $v_{i_1} = v_{i_2} = \cdots = v_{i_{2K}} = 1$ does not appear in one random I^j is $1 - \dfrac{1}{2^{2K}}$. Thus, the probability that $v_{i_1} = \cdots = v_{i_{2K}} = 1$ does not appear in any of m random I^js

[1] The same Boolean functions may be counted multiple times. But, it does not cause a problem since we use Ω notation.

is $(1-\frac{1}{2^{2K}})^m$. Since the number of combinations of $2K$ nodes is less than n^{2K}, the probability that there exists a combination of $2K$ nodes for which an assignment $v_{i_1} = \cdots = v_{i_{2K}} = 1$ does not appear in any of m random I^js is at most $n^{2K} \cdot (1-\frac{1}{2^{2K}})^m$. Since there are 2^{2K} possible assignments to $2K$ variables, the probability that the condition of Proposition 3 is not satisfied is at most $2^{2K} \cdot n^{2K} \cdot (1-\frac{1}{2^{2K}})^m$. It is not difficult to see that $2^{2K} \cdot n^{2K} \cdot (1-\frac{1}{2^{2K}})^m < p$ holds for $m > \ln 2 \cdot 2^{2K} \cdot (2K + 2K \log n + \log \frac{1}{p})$. Letting $p = \frac{1}{n^\alpha}$, we obtain the theorem. □

3.3 Computational Complexity Issue

The simple algorithm shown in Section 3.2 works in $O(mn^{K+1})$ time for constant K. Though it is polynomial, the degree of the polynomial becomes very high as K increases. Several efforts have been done to reduce the worst case time complexity and the practical computation time. However, it still takes long time for large K. Indeed, both the inference and identification problems are shown to be NP-hard if there is no restriction on K [2].

Though it is quite difficult to reduce the worst case time complexity, some greedy type approximation algorithms have been proposed. It is proven under the uniform distribution of samples that greedy type algorithms can identify BNs with high probability for wide-class of Boolean functions [6, 12]. Furthermore, some sophisticated algorithms are proposed which work for all types of Boolean functions under the uniform distribution [23].

4 Identification of Attractors

One of extensively studied topics for BNs is analysis of the number and length of attractors in randomly generated BNs with average indegree K, where attractors correspond to steady-states. Starting from [16], a fast increase of number of attractors has been seen [7, 10, 27]. Although there is no conclusive result on the mean length of attractors, many researches have also been done [10, 16]. Recently, several methods have been developed for efficient identification of attractors [9, 13, 15, 30], whereas it is known that finding a singleton attractor (i.e., a fixed point) is NP-hard [21, 30]. Devloo et al. developed a method using transformation to a constraint satisfaction problem [9]. Garg et al. developed a method based on Binary Decision Diagrams (BDDs) [13]. Irons developed a method that makes use of small subnetworks [15]. However, theoretical analysis of the average case time 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 [30]. In this section, we overview our algorithms and their analyses.

4.1 Attractors in Boolean Networks

As mentioned in Section 2, $\mathbf{v}(t+1)$ is determined from $\mathbf{v}(t)$ in a BN. Starting from an initial GAP $\mathbf{v}(0)$, 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 (i.e., $\mathbf{v} = \mathbf{f}(\mathbf{v})$) is called a *singleton attractor*, which corresponds to a fixed point. Otherwise, it is called a *cyclic attractor* with period p if it consists of p global states (i.e., $\mathbf{v}^1 = \mathbf{f}(\mathbf{v}^p) = \mathbf{f}(\mathbf{f}(\mathbf{v}^{p-1})) = \cdots = \mathbf{f}(\mathbf{f}(\cdots \mathbf{f}(\mathbf{v}^1)\cdots)))$. The set of all GAPs that eventually evolve into the same attractor is called the *basin of attraction*. Different basins of attraction correspond to different connected components in the state transition diagram, and each connected component contains exactly one directed cycle. For example, in Fig. 1, 001 is a singleton attractor, $\{011, 100\}$ is a cyclic attractor with period 2, and $\{111, 110, 010, 000, 001\}$ are the basin of the singleton attractor 001.

In this paper, the attractor identification problem is defined as a problem of enumerating all attractors for a given BN. However, it is very difficult to find attractors with long periods. Thus, we focus on identification of singleton attractors and identification of attractors with period at most some given threshold p_{max}. These problems are defined as below, where the *singleton attractor identification problem* corresponds to the case of $p_{max} = 1$.

Definition 3. [Identification of Attractors in BN]
Instance: *a BN and the maximum length of period* p_{max}*,*
Problem: *enumerate all attractors with period at most* p_{max}*.*

4.2 Simple Recursive Algorithm and Its Average Case Analysis

We developed several algorithms for identifying singleton attractors and cyclic attractors with short periods [30]. For that purpose, we proposed a very simple algorithm, which is referred to as the *basic recursive algorithm* in this paper.

The number of singleton attractors in a BN depends on the regulatory rules of the network. If the 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 all the singleton attractors are to be identified. 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 [22]. The basic recursive algorithm was designed based on these facts. It examines much smaller number of states than 2^n in the average case.

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 (i.e., a random 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 this algorithm is given below, where this procedure is invoked with $m = 1$.

```
**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**;
```

This algorithm extends a partial GAP by one gene at a time in a recursive manner. At the m-th recursive step, the states of the first $m - 1$ genes (i.e., a partial GAP) are already determined. Then, the algorithm extends the partial GAP by letting $v_m(t) = 0$. If $v_j(t + 1) = v_j(t)$ holds or the value of $v_j(t + 1)$ is not determined for each $j = 1, \ldots, m$, 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 letting $v_m(t) = 1$ 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 [30].

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

$$P(v_i(t) \neq v_i(t+1)) \ = \ 0.5 \cdot \frac{\binom{m}{k_i}}{\binom{n}{k_i}} \ \approx \ 0.5 \cdot (\frac{m}{n})^{k_i} \ \geq \ 0.5 \cdot (\frac{m}{n})^{K},$$

where we assume that Boolean functions of k_i inputs are selected at uniformly random. If $v_i(t) \neq v_i(t + 1)$ holds for some $i \leq m$, the algorithm cannot go to the next recursive level. Therefore, the probability that the algorithm examines the $(m + 1)$-th gene is no more than

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

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

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

Let $s = \frac{m}{n}$, 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 computing 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 we want to analyze the time complexity as a function of n, we only need to compute the maximum value of the function $g(s) = (2 - s^K)^s$, which can

Table 2. Theoretically estimated average case time complexities of basic, outdegree-based, and BFS-based algorithms for the singleton attractor detection problem [30].

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

be obtained by a simple numerical calculations for fixed K. Then, the average case time complexity of the algorithm can be estimated as $O((\max(g))^n)$. The average case time complexities for $K = 2, \ldots, 5$ are listed in the first row of Table 2. It should be noted that the naive exhaustive search-based algorithm takes at least $O(2^n)$ time. Thus, the basic recursive algorithm is much faster than the naive algorithm for small K.

We obtained variants of this basic recursive algorithm by sorting nodes before invoking the recursive procedure [30]. 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 (see Table 2). We also performed computational experiments to confirm these theoretical results (it is to be noted that some approximations were included in theoretical analyses). As a result, good agreement was observed. We also extended the basic recursive algorithm for identifying cyclic attractors with short period [30]. Though the extended algorithm is not efficient compared with those in Table 2, it still works in $o(2^n)$ time in the average case.

4.3 Issues on the Worst Case Time Complexity

We have considered the average case time complexity in the above. However, it is also very important to consider the worst case time complexity. We have shown that the singleton attractor detection problem (i.e., decide whether or not there exists a singleton attractor) can be solved in $o(2^n)$ time for constant K by a reduction to the satisfiability problem for CNF (conjunctive normal form). We have also shown that the singleton attractor detection problem can be solved in $o(2^n)$ time for general K if Boolean functions are restricted to AND/OR of literals [29]. However, no $o(2^n)$ time algorithm is known for more general cases of the singleton attractor detection problem and thus development of such an algorithm is left as an open problem.

5 Control of Boolean Networks

One of the major goals of systems biology is to develop a control theory for biological systems [17, 18]. Development of such a control theory is important both from a theoretical viewpoint and from a practical viewpoint. From a theoretical viewpoint, biological systems are complex and contain highly non-linear subsystems and thus existing methods in control theory cannot be directly applied to control of biological systems. Therefore, it is quite interesting to develop theory and methods for control of biological systems. From a practical viewpoint, control of cells may be useful for systems-based drug discovery and cancer treatment [17, 18]. Since BNs are highly non-linear systems, it is reasonable to try to develop methods for control of BNs.

Datta et al. proposed a method for finding a control strategy for PBN [8], from which many extensions followed [11, 25, 26]. In their approach, it is assumed that states of some nodes can be externally controlled and the objective is to find a sequence of control actions with the minimum cost that leads to a desirable state of a network. Their approach is based on the theory of Markov chains and makes use of the classical technique of dynamic programming. Since BNs are special cases of PBNs, their methods can also be applied to control of BNs. However, it is required in their methods to handle exponential size matrices and thus their methods can only be applied to small biological systems. Therefore, it is reasonable to ask how difficult it is to find control strategies for BNs. We showed that finding control strategies for BNs is NP-hard [5], which means that there is no polynomial time algorithm unless P=NP [14]. On the other hand, we showed that this problem can be solved in polynomial time if BN has a tree structure. In this section, we review these results along with the essential idea of [8].

5.1 Definition of the Control Problem

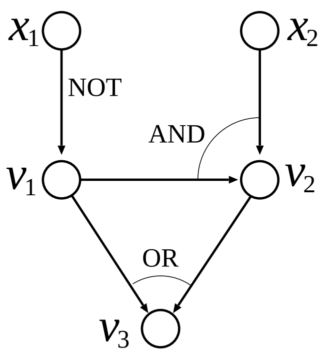
Here we give a formal definition of the problem of finding control strategies for BNs (Control of BN) [5].

In Control of BN, it is assumed that there exist 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, states of internal nodes ($v_i(t+1)$ 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, Control of BN is defined as follows (see also Fig. 3) [5].

Definition 4. (Control of BN)
Instance: *a BN, 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,*
Problem: *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.*



initial (t=0)	0	0	0
desired (t=3)	0	1	1

⇩

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

Fig. 3. Example of Control of BN. In this problem, given initial and desired states of internal nodes (v_1, v_2, v_3), it is required to compute a sequence of states of external nodes (x_1, x_2) leading to the desired state.

5.2 Dynamic Programming Algorithms for Control of BNs

As mentioned before, Datta et al. proposed a dynamic programming based method for finding a control strategy for PBN [8], which can also be applied to BN. Here, we briefly review their method in the context of BN.

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 control sequence $\langle \mathbf{x}(t), \ldots, \mathbf{x}(M) \rangle$ which leads to the target state $\mathbf{v}^m$ beginning from the state $[b_1, \ldots, b_n]$ at time t. This table is computed from $t = M$ to $t = 0$ by using the following procedure:

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

$$D[b_1, \ldots, b_n, t-1] = \begin{cases} 1, \text{ if there exists } (\mathbf{c}, \mathbf{x}) \text{ such that } D[c_1, \ldots, c_n, t] = 1 \\ \quad \text{and } \mathbf{c} = \mathbf{f}(\mathbf{b}, \mathbf{x}), \\ 0, \text{ otherwise,} \end{cases}$$

where $\mathbf{b} = [b_1, \ldots, b_n]$ and $\mathbf{c} = [c_1, \ldots, c_n]$. Then, there exists a desired control sequence if and only if $D[a_1, \ldots, a_n, 0] = 1$ holds for $\mathbf{v}^0 = [a_1, \ldots, a_n]$. Once the table is constructed, a desired control sequence can be obtained using the *traceback technique*, which is a standard technique in dynamic programming.

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 $O(M \cdot 2^{n+m})$ time excluding the time for calculation of Boolean functions. Therefore, the dynamic programming algorithm in [8] is an exponential time algorithm.

As shown in the next subsection, control of BN is NP-hard, which suggests that exponential time is inevitable in a general case. However, we may be able to develop polynomial time algorithms for special cases. We developed such an algorithm for the case where the network has a tree structure (i.e., the graph is connected and there is no cycle). Since the algorithm is a bit complicated, we show here a simple algorithm for the case where the network has a rooted tree structure (i.e., all paths are directed from leaves to the root). In order to compute a control strategy, we employ dynamic programming. Though dynamic programming is also employed in the above, it is used here in a significantly different way.

In order to apply dynamic programming, we define a table $S[v_i, t, b]$ as below, where v_i is a node in a BN, t is a time step and b is a Boolean value (i.e., 0 or 1). Here $S[v_i, t, b]$ takes 1 if there exists a control sequence (up to time t) that makes $v_i(t) = b$.

$$S[v_i, t, 1] = \begin{cases} 1, \text{ if there exists } \langle \mathbf{x}(0), \ldots, \mathbf{x}(t) \rangle \text{ such that } v_i(t) = 1, \\ 0, \text{ otherwise.} \end{cases}$$

$$S[v_i, t, 0] = \begin{cases} 1, \text{ if there exists } \langle \mathbf{x}(0), \ldots, \mathbf{x}(t) \rangle \text{ such that } v_i(t) = 0, \\ 0, \text{ otherwise.} \end{cases}$$

Then, $S[v_i, t, 1]$ can be computed by the following dynamic programming procedure.

$$S[v_i, t+1, 1] = \begin{cases} 1, \text{ if there exists } [b_{i_1}, \ldots, b_{i_k}] \text{ such that } f_i(b_{i_1}, \ldots, b_{i_k}) = 1 \\ \quad \text{holds and } S[v_{i_j}, t, b_{i_j}] = 1 \text{ holds for all } j = 1, \ldots, k, \\ 0, \text{ otherwise.} \end{cases}$$

$S[v_i, t, 0]$ can be computed in a similar way. It should be noted that each leaf is either a constant node or an external node. For a constant node, either $S[v_i, t, 1] = 1$ and $S[v_i, t, 0] = 0$ hold for all t, or $S[v_i, t, 1] = 0$ and $S[v_i, t, 0] = 1$ hold for all t. For an external node, $S[v_i, t, 1] = 1$ and $S[v_i, t, 0] = 1$ hold for all t. Since the size of table $S[v_i, t, b]$ is $O((n + m)M)$, the above dynamic programming algorithm works in polynomial time where we assume that the value of each Boolean function can be computed in polynomial time. A desired control sequence can also be obtained from the table in polynomial time using the traceback technique. This algorithm was extended for BNs with general tree structures [5].

Theorem 2. [5] *Control of BN can be solved in polynomial time if BN has a tree structure.*

5.3 NP-hardness Results on Control of BNs

As mentioned in the above, the dynamic programming algorithm for control of general BNs takes exponential time and thus is not efficient. However, the following theorem suggests that it is impossible (under the assumption of P≠NP) to develop a polynomial time algorithm for the general case.

Theorem 3. [5] *Control of BN is NP-hard.*

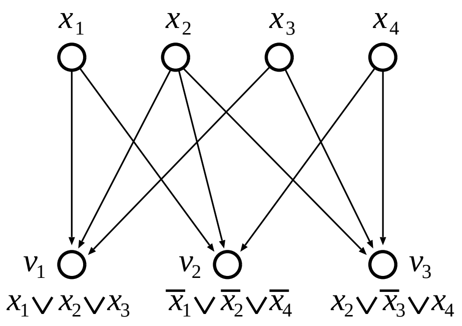

x_1
x_2
x_3
x_4
v_1
v_2
v_3
$x_1 \vee x_2 \vee x_3$
$\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}$
$x_2 \vee \overline{x_3} \vee x_4$

Fig. 4. Reduction from 3SAT to Control of BN. An instance of 3SAT $\{y_1 \vee y_2 \vee y_3, \overline{y_1} \vee \overline{y_2} \vee \overline{y_4}, y_2 \vee \overline{y_3} \vee y_4\}$ is transformed into this instance of Control of BN.

Proof. We present a simple polynomial time reduction from 3SAT [14]. 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 $y_1, \ldots, y_N$, 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 $y_1, \ldots, y_N$ which satisfies all the clauses (i.e., the values of all clauses are 1).

From an instance of 3SAT, we construct a BN as follows (see also Fig. 4). We let the set of nodes $V = \{v_1, \ldots, v_L, x_1, \ldots, x_N\}$ where each v_i corresponds to c_i and each x_j corresponds to y_j. Suppose that $f_i(y_{i_1}, \ldots, y_{i_3})$ is a Boolean function assigned to c_i in 3SAT. Then, we assign $f_i(x_{i_1}, \ldots, x_{i_3})$ to v_i in the BN. Finally, we let $M = 1$, $\mathbf{v}^0 = [0, 0, \ldots, 0]$ and $\mathbf{v}^M = [1, 1, \ldots, 1]$.

Then, it is straight-forward to see that there exists a control strategy $\langle \mathbf{x}(0), \mathbf{x}(1) \rangle$ which makes $\mathbf{v}(1) = [1, 1, \ldots, 1]$ if and only if there exists an assignment which satisfies all the clauses. Since the reduction can be done in linear time, Control of BN is NP-hard. $\square$

It is also shown in [5] that the control problem remains NP-hard even for BNs having very restricted network structures. Especially, it is shown that it remains NP-hard if the network contains only one control node and all the nodes are OR or AND nodes (i.e., there is no negative control). However, it is unclear whether the control problem is NP-hard or can be solved in polynomial time if a BN contains a fixed number of directed cycles or loops (it is unclear even for the case of two cycles). Deciding the complexity of such a special case is left as an open problem.

Though we have shown negative results, NP-hardness does not necessarily mean that we cannot develop practical algorithms which work efficiently in the average case. For that purpose, an approximate finite-horizon optimal control has been introduced [24] and a heuristic method based on Q-learning algorithm for approximating the optimal infinite-horizon control policy has been proposed

[11]. However, application of these approaches is still limited to small networks. Recently, Langmund and Jha proposed a method using techniques from the field of *model checking* [19]. They reported that the method could be applied to a BN model of embryogenesis in *D. melanogaster* with 15,360 Boolean variables.

6 Concluding Remarks

We have overviewed algorithmic aspects of inference, analysis and control of BNs with focusing on the authors' works. We understand that there is a criticism that BN is too simple as a model of genetic networks and/or other biological networks. However, studies on BNs may provide some insights into other models. At least, negative results should hold for more general models. Some ideas in positive results may also be useful for theoretical analyses and design of algorithms for more general models. For instance, Mochizuki performed theoretical analysis on the number of steady states in some continuous biological networks that are based on nonlinear differential equations [22]. The core part of the analysis is done in a combinatorial manner and is very close to that for BNs. Therefore, it is worthy to study extensions of the methods and results on BNs for more general models.

References

1. Akutsu, T., Miyano, S., Kuhara, S.: Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. Proc. Pacific Symposium on Biocomputing 1999, 17–28 (1999)
2. Akutsu, T., Miyano, S., Kuhara, S.: Algorithms for identifying Boolean networks and related biological networks based on matrix multiplication and fingerprint function. Journal of Computational Biology, 7, 331–343 (2000)
3. Akutsu, T., Miyano, S., Kuhara, S.: Inferring qualitative relations in genetic networks and metabolic pathways. Bioinformatics, 16, 727–734 (2000)
4. Akutsu, T., Kuhara, S., Maruyama, O., Miyano, S.: Identification of genetic networks by strategic gene disruptions and gene overexpressions under a boolean model. Theoretical Computer Science, 298, 235–251 (2003)
5. Akutsu, T., Hayashida, M., Ching, W-K., Ng, M. K.: Control of Boolean networks: Hardness results and algorithms for tree-structured networks. Journal of Theoretical Biology, 244, 670–679 (2007)
6. Arpe, J., Reischuk, R.: When does greedy learning of relevant attributes succeed? Lecture Notes in Computer Science, vol. 4598, pp. 296–306. Springer, Heidelberg (2007)
7. Bilke, S., Sjunnesson, F.: Number of attractors in random Boolean networks. Physical Review E, 72, 016110 (2005)
8. Datta, A., Choudhary, A., Bittner, M. L., Dougherty, E. R.: External control in Markovian genetic regulatory networks. Machine Learning, 52, 169–191 (2003)
9. Devloo, V., Hansen, P., Labbé, M.: Identification of all steady states in large networks by logical analysis. Bulletin of Mathematical Biology, 65, 1025–1051 (2003)
10. Drossel, B., Mihaljev, T., Greil, F.: Number and length of attractors in a critical Kauffman model with connectivity one. Physical Review Letters, 94, 088701 (2005)

11. Faryabi, B., Datta, A., Dougherty, E. R.: On approximate stochastic control in genetic regulatory networks. IET Systems Biology, 1, 361–368 (2007)
12. Fukagawa, D., Akutsu, T.: Performance analysis of a greedy algorithm for inferring Boolean functions. Information Processing Letters, 93, 7–12 (2005)
13. Garg, A., Xenarios, I., Mendoza, L., DeMicheli, G.: An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments. Lecture Notes in Computer Science, vol. 4453, pp. 62–76. Springer, Heidelberg (2007)
14. Garey, M. R., Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., New York (1979)
15. Irons, D. J.: Improving the efficiency of attractor cycle identification in Boolean networks. Physica D, 217, 7–21 (2006)
16. Kauffman, S. A.: The Origins of Order: Self-organization and Selection in Evolution. Oxford Univ. Press, New York (1993)
17. Kitano, H.: Computational systems biology. Nature, 420, 206–210 (2002)
18. Kitano, H.: Cancer as a robust system: implications for anticancer therapy. Nature Reviews Cancer, 4, 227–235 (2004)
19. Langmead, C. J., Jha, S. K.: Symbolic approaches for finding control strategies in Boolean networks. Proc. 6th Asia-Pacific Bioinformatics Conference, pp. 307–319. Imperial College Press, London (2008)
20. Liang, S., Fuhrman, S., Somogyi, R.: REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. Proc. Pacific Symposium on Biocomputing 1998, pp. 18–29 (1998)
21. Milano, M., Roli, A.: Solving the safistiability problem through Boolean networks. Lecture Notes in Artificial Intelligence, vol. 1792, pp. 72–93. Springer, Heidelberg (2000).
22. Mochizuki, A.: An analytical study of the number of steady states in gene regulatory networks. Journal of Theoretical Biology, 236, 291–310 (2005)
23. Mossel, E., O'Donnell, R., Servedio, R. A.: Learning functions of k relevant variables. Journal of Computer and System Sciences, 69, 421–434 (2004)
24. Ng, M. K., Zhang, S-Q., Ching, W-K., Akutsu, T.: A control model for Markovian genetic regulatory network. Transactions on Computational Systems Biology, V, 36–48 (2006)
25. Pal, R., Datta, A., Bittner, M. L., Dougherty, E. R.: Intervention in context-sensitive probabilistic Boolean networks. Bioinformatics, 21, 1211–1218 (2005)
26. Pal, R., Datta, A., Bittner, M. L., Dougherty, E. R.: Optimal infinite-horizon control for probabilistic Boolean networks. IEEE Transactions on Signal Processing, 54, 2375–2387 (2006)
27. Samuelsson, B., Troein, C.: Superpolynomial growth in the number of attractors in kauffman networks. Physical Review Letters, 90, 098701 (2003)
28. Shmulevich, I., Dougherty, E.R., Kim, S., Zhang, W.: Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. Bioinformatics, 18, 261–274 (2002)
29. Tamura, T., Akutsu, T.: An improved algorithm for detecting a singleton attractor in a Boolean network consisting of AND/OR nodes. Proceedings of the 3rd International Conference on Algebraic Biology, to appear.
30. Zhang, S-Q., Hayashida, M., Akutsu, T., Ching, W-K., Ng, M. K.: Algorithms for finding small attractors in Boolean networks. EURASIP Journal on Bioinformatics and Systems Biology, 2007, 20180 (2007)