Algorithms for Analysis and Control of Boolean Networks

Tatsuya Akutsu(✉)

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

Abstract. Boolean network is a discrete mathematical model of gene regulatory networks. In this short article, we briefly review algorithmic results on finding attractors in Boolean networks. Since it is known that the problem of finding a singleton attractor is NP-hard and the problem can be trivially solved in $O^*(2^n)$ time (under a reasonable assumption), we focus on special cases in which the problem can be solved in $O((2-\delta)^n)$ time for some constant $\delta > 0$. We also briefly review algorithmic results on control of Boolean networks.

Keywords: Boolean networks · Attractors · Controllability

1 Boolean Networks

Mathematical analysis of biological networks is an important topic in bioinformatics and computational biology. For that purpose, various kinds of mathematical models have been proposed. Among them, the *Boolean network* (BN) has been extensively studied since 1960's [3]. BN is a discrete mathematical model of gene regulatory networks, in which each node (e.g., gene) takes either 0 or 1 and the states of nodes change synchronously according to regulation rules given as Boolean functions, where 1 (resp., 0) means that the corresponding gene is expressed (resp., not expressed).

Formally, a BN $N(V, F)$ consists of a set $V = \{x_1, \ldots, x_n\}$ of nodes and a list $F = (f_1, \ldots, f_n)$ of *Boolean functions*, where a Boolean function $f_i(x_{i_1}, \ldots, x_{i_{k_i}})$ with inputs from specified nodes $x_{i_1}, \ldots, x_{i_{k_i}}$ is assigned to each node x_i. We use $IN(x_i)$ to denote the set of input nodes $x_{i_1}, \ldots, x_{i_k}$ to x_i. Each node takes either 0 or 1 at each discrete time t, and the state of node x_i at time t is denoted by $x_i(t)$. Then, the state of node x_i at time $t+1$ is determined by

$$x_i(t+1) = f_i(x_{i_1}(t), \ldots, x_{i_{k_i}}(t)).$$

The state of the whole network at time step t is represented by an n-dimensional 0–1 vector $\mathbf{x}(t) = [x_1(t), \ldots, x_n(t)]$. We also write $x_i(t+1) = f_i(\mathbf{x}(t))$ to denote

Partially supported by JSPS, Japan: Grant-in-Aid 26240034.

the regulation rule for x_i and $\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$ to denote the regulation rule for the whole BN. The network structure of a BN $N(V,F)$ is represented by a directed graph $G(V,E)$ such that $E = \{(x_{i_j}, x_i) | x_{i_j} \in IN(x_i)\}$. The dynamics of a BN can be well represented by a *state transition diagram*, in which a vertex and a directed edge correspond to a (global) state of the BN and a state transition, respectively. For example, consider a BN $N(V,F)$ defined by

$$\begin{aligned} x_1(t+1) &= x_3(t), \\ x_2(t+1) &= x_1(t) \wedge \overline{x_3(t)}, \\ x_3(t+1) &= x_1(t) \wedge \overline{x_2(t)}. \end{aligned}$$

Then, $G(V,E)$ and its state transition diagram are as in Fig. 1(A) and (B), respectively.

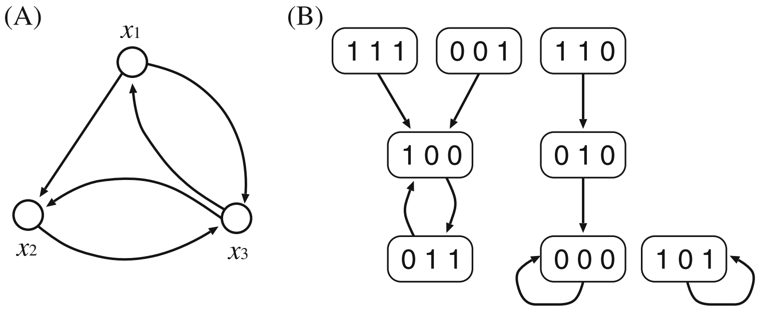


Fig. 1. Example of BN. (A) $G(V,E)$. (B) State transition diagram.

Starting from any initial state, a BN will eventually reach a cyclic sequence of states, called an *attractor*, which is often regarded as a type of a cell. An attractor consisting of only one global state (i.e., $\mathbf{x} = \mathbf{f}(\mathbf{x})$) is called a *singleton attractor*. Otherwise, it is called a *periodic attractor*. A periodic attractor consisting of p states is called a p-periodic attractor. For example, the BN given in Fig. 1 has two singleton attractors $\langle [0,0,0] \rangle$ and $\langle [1,0,1] \rangle$, and one 2-periodic attractor $\langle [0,1,1],[1,0,0] \rangle$.

2 Attractor Detection

After making a BN model of some organism or its part, it is important to find or enumerate attractors because they are considered to correspond to cell types. Since an attractor corresponds to a directed cycle in a state transition diagram, attractors can be enumerated by enumerating cycles in the diagram. Since a state transition diagram consists of 2^n vertices and 2^n edges, enumeration of attractors can be done in $O^*(2^n)$ time[1] once the diagram is constructed. Construction of

[1] $O^*(f(n))$ denotes $O(f(n)poly(n))$.

the diagram can also be done in $O^*(2^n)$ time if the value of each Boolean function appearing in the BN can be computed in polynomial time. On the other hand, it is known that deciding whether or not there exists a singleton attractor is NP-hard. Since it is quite difficult to break the $O^*(2^n)$ barrier for the general case, existing studies focused on developments of $O^*((2-\delta)^n)$ time algorithms for special but important subclasses of BNs. In particular, the author and his colleagues developed the following algorithms [1]:

- $O(1.587^n)$ time algorithm for detection of a singleton attractor for a BN consisting of AND/OR functions (i.e., each Boolean function is a conjunction or disjunction of literals),
- $O(1.871^n)$ time algorithm for detection of a singleton attractor for a BN consisting of nested canalyzing functions (see [1] for the definition of a nested canalyzing function),
- $O(1.985^n)$ time algorithm for detection of a 2-periodic attractor for a BN consisting of AND/OR functions,
- $O^*(n^{2p(w+1)})$ time algorithm for detection of a p-periodic attractor for a BN consisting of nested canalyzing functions and having bounded treewidth w.

Improvements of these algorithms and developments of $O^*((2-\delta)^n)$ time algorithms for other important subclasses are left as open problems.

3 Control of Boolean Networks

Recently, control of BNs has captured a lot of attentions because of its potential application to control of cells and diseases. In particular, algebraic approaches based on *semi-tensor product* have been extensively studied [2,4].

Here, we focus on computational complexity of control of BNs. Although there exist many variants, one of the fundamental control problems is defined as: 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. Formally, this problem is defined as follows. Let $V = \{x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m}\}$, where $x_1, \ldots, x_n$ are internal nodes and $x_{n+1}, \ldots, x_{n+m}$ are external nodes (i.e., control nodes). We use u_i to denote an external node x_{n+i}. Let $\mathbf{x}(t) = [x_1(t), \ldots, x_n(t)]$ and $\mathbf{u}(t) = [u_1(t), \ldots, u_m(t)]$. Then, the state of each internal node $x_i(t+1)$ $(i = 1, \ldots, n)$ at time step $t+1$ is determined by

$$x_i(t+1) = f_i(x_{i_1}(t), \ldots, x_{i_{k_i}}(t)),$$

where each x_{i_j} is either an internal node or an external node. We can describe the dynamics of a BN with external nodes by

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)).$$

Then, control of BN is defined as follows (see also Fig. 2): given a BN with external nodes, initial and target states $\mathbf{x}^0$ and $\mathbf{x}^M$ of the internal nodes, find a sequence of 0–1 vectors $\langle \mathbf{u}(0), \ldots, \mathbf{u}(M-1) \rangle$ such that $\mathbf{x}(0) = \mathbf{x}^0$ and $\mathbf{x}(M) = \mathbf{x}^M$

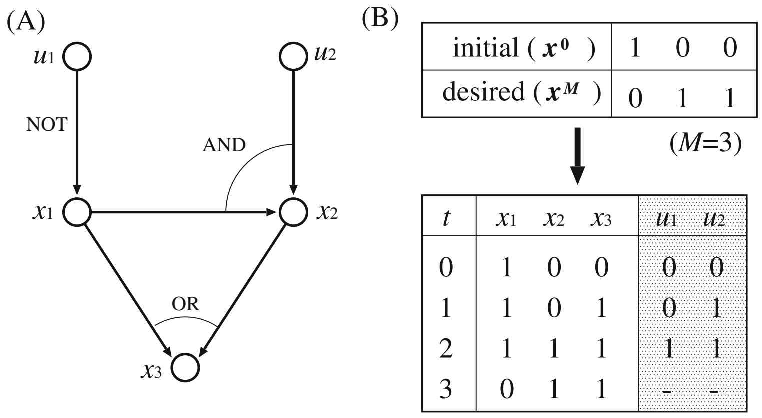
(if there does not exist such a sequence, "None" should be the output). For example, consider a BN defined by

$$x_1(t+1) = \overline{u_1(t)},$$
$$x_2(t+1) = x_1(t) \wedge u_2(t),$$
$$x_3(t+1) = x_1(t) \vee x_2(t).$$

Suppose that $\mathbf{x}^0 = [1,0,0]$, $\mathbf{x}^M = [0.1.1]$, and $M = 3$ (see also Fig. 2). Then, we have a control sequence $\langle \mathbf{u}(0), \mathbf{u}(1), \mathbf{u}(2) \rangle$ with

$$\mathbf{u}(0) = [0,0], \quad \mathbf{u}(1) = [0,1], \quad \mathbf{u}(2) = [1,1],$$

which drives the BN from $\mathbf{x}^0$ to $\mathbf{x}^M$.



initial ($\boldsymbol{x}^0$)	1	0	0
desired ($\boldsymbol{x}^M$)	0	1	1

t	x_1	x_2	x_3	u_1	u_2
0	1	0	0	0	0
1	1	0	1	0	1
2	1	1	1	1	1
3	0	1	1	-	-

Fig. 2. Example of control of BN. (A) BN with external nodes u_1 and u_2. (B) State transitions from the initial state to the target state.

This control problem can be solved by a dynamic programming algorithm as follows. We use a table $D[b_1, \ldots, b_n, t]$, where each entry (except t) takes either 0 or 1, and t corresponds to a time step. $D[b_1, \ldots, b_n, t]$ takes 1 if there exists a control sequence $\langle \mathbf{u}(t), \ldots, \mathbf{u}(M-1) \rangle$ which leads to the target state $\mathbf{x}^M$ 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:

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

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

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 $\mathbf{x}^0 = [a_1, \ldots, a_n]$. Once the table is constructed, a desired control sequence can be obtained using the standard traceback technique. It is easy to see that this algorithm requires $O(M \cdot 2^{n+m})$ time excluding the time for calculation of Boolean functions. This is an exponential-time algorithm. Actually, the problem is NP-hard even for $M = 1$ and BNs with very simple network structures. Furthermore, it is PSPACE-hard if M is not bounded [1]. A polynomial time algorithm is known only for BNs with tree structures [1]. Development of an algorithm that is faster than the $O(M \cdot 2^{n+m})$ time one is left as an open problem, even for considerably restricted BNs.

References

1. Akutsu, T.: Algorithms for Analysis, Inference, and Control of Boolean Networks. World Scientific, Singapore (2018)
2. Cheng, D., Qi, H., Li, Z.: Analysis and Control of Boolean Networks: A Semi-tensor Product Approach. Springer, London (2011). https://doi.org/10.1007/978-0-85729-097-7
3. Kauffman, S.A.: Homeostasis and differentiation in random genetic control networks. Nature **224**, 177–178 (1969)
4. Lu, J., Li, H., Liu, Y., Li, F.: Survey on semi-tensor product method with its applications in logical networks and other finite-valued systems. IET Control Theory Appl. **11**, 2040–2047 (2017)