ON THE COMPLEXITY OF FINDING CONTROL STRATEGIES FOR BOOLEAN NETWORKS

TATSUYA AKUTSU * MORIHIRO HAYASHIDA
Bioinformatics Center, Institute for Chemical Research, Kyoto University
Uji-city, Kyoto 611-0011, Japan
E-mail: {takutsu, morihiro}@kuicr.kyoto-u.ac.jp

WAI-KI CHING †
Department of Mathematics, The University of Hong Kong
Pokfulam Road, Hong Kong, China
E-mail: wkc@maths.hku.hk

MICHAEL K. NG ‡
Department of Mathematics, Hong Kong Baptist University
Kowloon Tong, Hong Kong, China
E-mail: mng@math.hkbu.edu.hk

This paper considers a problem of finding control strategies for Boolean networks, where Boolean networks have been used as a model of genetic networks. This paper shows that finding a control strategy leading to the desired global state is NP-hard even if there is only one control node in the network. This result justifies existing exponential time algorithms for finding control strategies for probabilistic Boolean networks. On the other hand, this paper shows that the problem can be solved in polynomial time if the network has a tree structure.

1. Introduction

One of the important future directions of bioinformatics and systems biology is to develop a control theory for complex biological systems. For example, Kitano[1,2] mentions that identification of a set of perturbations that induces desired changes in cellular behaviors may be useful for systems-based drug discovery and cancer treatment. Though many attempts have been done based on control theory, existing theories and technologies are not satisfactory. Many important results in control theory are based on linear algebra, but it seems that biological systems contain many non-linear subsystems. Therefore, it is required to develop a control theory for complex biological systems.

*Work partially supported by Grant Nos. #17017019 and #16300092 from MEXT, Japan.
†Work partially supported by RGC Grant Nos. HKU 7126/02P, and HKU CRGC Grant Nos. 10203919,10204437
‡Work partially supported by RGC Grant Nos. HKU 7130/02P, 7046/03P, 7035/04P, 7035/05P.

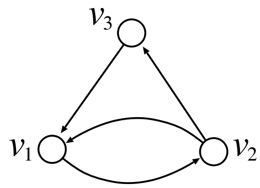
Various mathematical models have been proposed for modeling complex and nonlinear biological systems. Among them, the *Boolean network* (BN)[3] has been well-studied.[3,4,5,6,7,8] BN is a very simple model: each node (e.g., gene) takes either 0 (inactive) or 1 (active) and the states of nodes change synchronously. Though Boolean networks can not model detailed behaviors of biological systems, it may provide good approximations to the nonlinear functions appearing in many biological systems.[6] For example, Harris *et al.*[7] analyzed published data for over 150 regulated transcription systems and discussed relations between real transcription networks and Boolean networks. Therefore, it is reasonable to seek for a control theory for BNs. Even if a control theory for BNs is not practical, it may provide a new theoretical insight for systems biology.

Many studies have been done for understanding dynamical properties of BNs. For example, distribution of attractors,[3,5] relationship between network topology and chaotic behavior,[6] and inference of BNs from gene expression data[4,8] have been extensively studied. However, not much attention has been paid for finding control strategies on BNs. Recently, Datta *et al.*[9,10,11] proposed methods for finding a control strategy for probabilistic Boolean networks (PBNs), where a PBN[12] is an extension of a BN (therefore, a BN is a special case of a PBN). 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 the desirable state of a network. Since BNs are special cases of PBNs, their methods can also be applied to finding a control strategy for BNs. However, their methods require high computational costs: it is required to handle exponential size matrices. 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.

In this paper, we show that the control problem on BNs is NP-hard in general. This result justifies the use of exponential time algorithms for general BNs (and PBNs) as done by Datta *et al*. We further show that the control problem remains NP-hard even for some restricted cases of BNs. On the other hand, we show that the control problem can be solved in polynomial time if a BN has a tree topology. We finally discuss biological implications of the theoretical results.

2. Boolean Network and Its Control

First, we briefly review BN.[3] 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 treated as a model of a genetic network. Each node takes either 0 or 1 at each discrete time t, a regulation rule for each node is given by a Boolean function, and the states of nodes change synchronously. An example is given in Fig. 1. In this case, 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. Dynamics of a BN is well-described by a state transition table shown in Fig. 1. The first row means that if the state of BN is $[0,1,1]$ at time t then the state will be $[1,0,0]$ at time $t+1$. PBN[12] is an extension of BN, in which multiple Boolean functions are assigned to each node and one function is selected at each time t according to a given probability distribution. Therefore, BN is a special case of PBN in which the same function is always selected for each node.



$v_1(t+1) = v_2(t)$ AND $v_3(t)$
$v_2(t+1) = v_1(t)$
$v_3(t+1) =$ NOT $v_2(t)$

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

Figure 1. Example of a Boolean network (BN). Dynamics of BN (left) is well-described by a state transition table (right). For example, if the state of BN is $[0, 1, 1]$ at time t, the state will be $[1, 0, 0]$ at time $t + 1$.

In order to consider the control problem, we add *external control nodes* to a BN (original nodes are called *internal nodes*). The states of external nodes are not determined by Boolean functions. Instead, these are given externally.

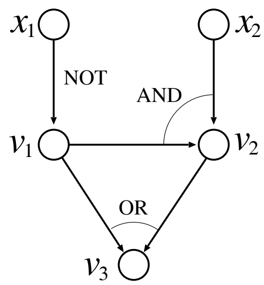
Now, we formally define the control problem. A BN with external control is represented by a set V of $n + m$ nodes $V = \{v_1, \ldots, v_n, v_{n+1}, \ldots, v_{n+m}\}$, where $v_1, \ldots, v_n$ are internal nodes (corresponding to genes) and $v_{n+1}, \ldots, v_{n+m}$ are external control nodes. We also use x_i to denote an external node v_{n+i} when it is convenient to distinguish external nodes from internal nodes. 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 $v_i(t)$. The value of each v_i ($i = 1, \ldots, n$) is directly controlled by k_i other nodes. Let $IN(v_i) = \{v_{i_1}, \ldots, v_{i_{k_i}}\}$ be the set of controlling elements of v_i, where $1 \leq i_j \leq n + m$. We assign to each v_i a Boolean function $f_i(v_{i_1}, \ldots, v_{i_{k_i}})$. Then the dynamics of the system is given by

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

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 network topology of a BN. We let $\mathbf{v}(t) = [v_1(t), \ldots, v_n(t)]$ and $\mathbf{x}(t) = [x_1(t), \ldots, x_m(t)]$. Note that a node without incoming edges is either an external node or a *constant node*, where a constant node is a node with a constant state.

Definition 2.1. (BN-CONTROL)
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 (BN-CONTROL) 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, "No" should be the output.

In this paper, a *control strategy* denotes a sequence of states of control nodes $\langle \mathbf{x}(0), \mathbf{x}(1), \ldots, \mathbf{x}(M) \rangle$. Fig. 2 illustrates BN-CONTROL. The left part is a BN, where v_1, v_2, v_3 are internal nodes, and x_1, x_2 are external nodes. We are also given initial and desired states as in the right top part of Fig. 2. If the control sequence is given as in the shaded region of Fig. 2, the state of BN will change as in the right bottom part and we will have the desired state at time $t = 3$.



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

Figure 2. Example of BN-CONTROL. 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) that leads to the desired state.

The desired states of all nodes are specified in the above. However, it may not be required to specify states of all the nodes because we may be interested only in controlling several important nodes (a set of these nodes is denoted by V' in this paper). We call this case *partial BN-CONTROL*.

In this paper, we assume that the number of input variables for each Boolean function is bounded by a constant. Otherwise, it is computationally difficult to find a control strategy even for one Boolean function (for example, one can consider a function representing a SAT formula). Due to this assumption, we can assume that enumeration of satisfying assignments can be done in constant time per Boolean function.

3. Hardness of Finding Control Strategies

As mentioned before, Datta *et al.*[9,10,11] proposed algorithms for finding control strategies for PBN based on Markov chains and dynamic programming. However, their algorithms are not efficient because it is required to consider all possible states of PBN (or BN) at all time steps between the initial and final time steps. For example, we need to consider state transition matrices of size $O(2^n \times 2^n)$ because there are $O(2^n)$ possible states and transitions among them must be also considered. We show here that the control problem is NP-hard in general, which implies that the approach by Datta *et al.* is reasonable.

Theorem 3.1. *BN-CONTROL is NP-hard.*

Proof. We present a simple polynomial time reduction from 3SAT[13] to BN-CONTROL (see Fig. 3), where a similar reduction was used in a study on Bayesian networks.[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 an instance of BN-CONTROL as follows. 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

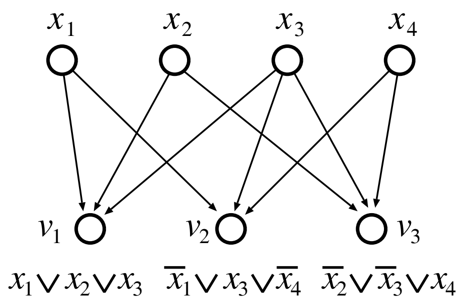


Figure 3. Reduction from 3SAT to BN-CONTROL. An instance of 3SAT $\{y_1 \vee y_2 \vee y_3, \overline{y_1} \vee y_3 \vee \overline{y_4}, \overline{y_2} \vee \overline{y_3} \vee y_4\}$ is transformed into an instance of BN-CONTROL in a simple way that external nodes correspond to variables in 3SAT, internal nodes correspond to clauses, and all the nodes must have value 1 at the desired state.

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 BN-CONTROL. Finally, we let $M = 1$, $\mathbf{v}^0 = [0, 0, \ldots, 0]$ and $\mathbf{v}^M = [1, 1, \ldots, 1]$.

Then, there exists a sequence $\langle \mathbf{x}(0), \mathbf{x}(1) \rangle$ which makes $\mathbf{v}(1) = [1, 1, \ldots, 1]$ if and only if there exists an assignment which satisfies all the clauses (see Fig. 3). Actually, a satisfying assignment for 3SAT corresponds to $\mathbf{x}(0)$. Since the above reduction can be done in linear time, BN-CONTROL is NP-hard. □

Since BN-CONTROL is a special case of partial BN-CONTROL, NP-hardness of partial BN-CONTROL directly follows from the above result. We can still prove that partial BN-CONTROL is NP-hard even if the desired state of only one node is specified. For that purpose, we simply add an internal node v_{L+1} to the BN in the above proof. Then, we let f_{L+1} be the conjunction of $v_1, \ldots, v_L$, and let $M = 2$, $v_{L+1}^0 = 0$ and $v_{L+1}^M = 1$.

Corollary 3.1. *Partial BN-CONTROL is NP-hard.*

Datta *et al.*[9] considered general cost functions C_k and C_M. We can consider a special case where $C_k = 0$ and C_M is the Hamming distance between the specified desired state and the final state given by a control strategy. Then, BN-CONTROL corresponds to the problem of asking whether or not the minimum cost is 0. Since BNs are special cases of PBNs, it follows that finding an optimal control strategy for PBN is NP-hard.

Corollary 3.2. *Finding an optimal control strategy for PBN is NP-hard.*

It is also possible to show that approximation of the Hamming distance is quite hard. For that purpose, we modify the network in the proof of Corollary 3.1. We add h nodes v_{L+1+i} $(i = 1, \ldots, h)$ with regulation rules $v_{L+1+i}(t+1) = v_{L+1}(t)$. Then, we let $V' = \{v_{L+2}, \ldots, v_{L+1+h}\}$, $M = 3$, $v_i^0 = 0$ and $v_i^M = 1$ for all $v_i \in V'$. Then, the cost is either 0 or h, which implies that obtaining approximate solutions (within a factor of $O(n)$ if we let $h = O(n)$) is still NP-hard.

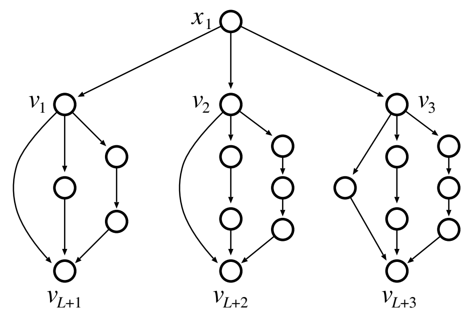


Figure 4. The network constructed (in the proof of Thm. 3.2) from the same 3SAT instance as in Fig. 3.

In the above, we used many control nodes. However, it is not plausible that we can control many genes. Thus, it is worthy to consider the following special case.

Theorem 3.2. *BN-CONTROL and partial BN-CONTROL are NP-hard even if there exists only one control node and the network structure is an almost tree of bounded degree.*

Proof. We give a proof for the partial control problem. Modification of the proof for BN-CONTROL is omitted in this version. As in Thm. 3.1, we use a reduction from 3SAT (see also Fig. 4). We construct an instance of the partial control problem so that the sequence of values of the single control node x_1 constitutes the satisfying assignment. For each clause c_i, we construct two special nodes v_i and v_{L+i}. Suppose that variables y_{i_1}, y_{i_2}, y_{i_3} appear in clause c_i in 3SAT. Then, we create 3 paths from v_i to v_{L+i}, where the lengths of paths are i_1, i_2 and i_3, respectively. The identify function is assigned to each gene (except v_{L+i}) in the paths, and a function corresponding to c_i is assigned for v_{L+i}. Then, we let $V' = \{v_{L+1}, \ldots, v_{2L}\}$, $M = N+1$, $v_i^0 = 0$ and $v_i^M = 1$ for $v_i \in V'$.

Then, the state $x_1(N-i)$ corresponds to an assignment of 0-1 value to y_i. From this, there exists a sequence $\langle \mathbf{x}(0), \mathbf{x}(1), \ldots, \mathbf{x}(N+1) \rangle$ which makes $v_i(N+1) = 1$ for all $v_i \in V'$ if and only if there exists an assignment which satisfies all the clauses. Therefore, partial BN-CONTROL is NP-hard even if there is only one control input.

Note that the above network structure belongs to the class of *almost trees*, where an undirected graph is called an almost tree if the number of edges in each bi-connected component is at most the number of nodes in the component plus some constant. Though the degree of x_1 can be high, it can be reduced to 3 by using a substructure like binary tree. □

4. Algorithms for Trees

In this section, we present polynomial time algorithms for special cases of the control problem. First, we consider the case where the network has a rooted tree structure (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 exponential time algorithms[9,10] for PBNs, it is used here in a significantly different way.

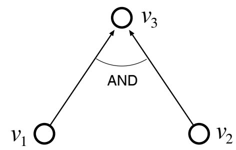


Figure 5. Computation of $S[v_3, t, 1]$. In this case, $S[v_3, t+1, 1] = 1$ if and only if $S[v_1, t, 1] = 1$ and $S[v_2, t, 1] = 1$. $S[v_3, t+1, 0] = 1$ if and only if $S[v_1, t, 0] = 1$ or $S[v_2, t, 0] = 1$.

In order to apply dynamic programming, we define $S[v_i, t, b]$ as below, where v_i is a node, t is a time step and b is a Boolean value (i.e., 0 or 1). Here $S[v_i, t, b]$ is 1 if there exists a control sequence (up to time t) that makes $v_i(t) = b$ (see also Fig. 5).

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

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

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

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

$S[v_i, t, 0]$ 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.

In the control problems, states of some (or all) internal nodes at the M-th step (more generally, at the t-th step) may be specified. Let $C[v_i, t, b] = 1$ denotes the constraint that the state of v_i at the t-th step can be b ($b \in \{0, 1\}$), otherwise $C[v_i, t, b] = 0$. For example, if $v_i(M) = 1$ must hold, we let $C[v_i, M, 1] = 1$ and $C[v_i, M, 0] = 0$. Then, we can modify the recurrence in dynamic programming as:

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

Then, we can decide whether or not there exists a control sequence by checking whether $S[v, M, 1] = 1$ or $S[v, M, 0] = 1$ holds for each node v. The required control sequence can be obtained by using the well-known traceback technique.[15]

Based on the above algorithm, we have the following theorem where the proof is omitted in this version.

Theorem 4.1. *If a BN has a rooted tree structure, both BN-CONTROL and partial BN-CONTROL can be solved in $O((n+m)M)$ time.*

We can generalize Thm. 4.1 for the case of unrooted trees. We call v_i a *branching node* if v_i has at least two outgoing edges. We call v_i an *outmost branching node* if either v_i is the only one branching node, or all paths from v_i to other branching nodes must pass the same branching node v_j. We denote such v_j by $nb(v_i)$.

Then, we can determine $S_0[v_i, t, b]$'s by repeatedly removing outmost branching nodes (see also Fig. 6 and Fig. 7), where we use $S_0[v_i, t, b]$ to denote the required table. For an outmost branching node v, we let

$$\Gamma^+(v) = \{w|(v,w) \in E\} - \{u\} \quad \text{and} \quad \Gamma^-(v) = \{w|(w,v) \in E\} - \{u\},$$

where u is the node adjacent to v and lying between v and $nb(v)$. If there is only one branching node, u can be empty. For each adjacent node w (except u) of v, we let $T_{v,w}$ be the subtree induced by $\{v, w\} \cup \{z|dist(v, z) < dist(nb(v), z)\}$, where $dist(v, z)$ denotes the number of edges of the path connecting v and z (without considering directions of edges). If $(u, v) \in E$, T_v is the subtree induced by v, u and the nodes in $\cup_{w \in \Gamma^-} T_{v,w}$. Otherwise (i.e., $(v, u) \in E$ or u is empty), T_v is the subtree induced by v and the nodes in $\cup_{w \in \Gamma^-} T_{v,w}$. It is worthy to note that $T_{v,w}$ is always a rooted tree and thus the algorithm for rooted trees can be used as a subroutine. Using the following procedure, we can determine $S_0[v, t, b]$.

Procedure BN-CONTROL-TREE
  for all v, t and $b \in \{0, 1\}$ **do** $S_0[v, t, b] \leftarrow 1$; $C[v, t, b] \leftarrow 1$
  while there exists a branching node **do**
    Select an arbitrary outmost and non-processed branching node v
    for all $w \in \Gamma^+(v)$ **do**
      for all t_0 and b_0 **do**
        if there does not exist a control strategy for $T_{v,w}$ such that $S[v, t_0, b_0] = 1$
        then $S_0[v, t_0, b_0] \leftarrow 0$
      Delete nodes in $T_{v,w}$ (except v)
    for all t and b **do** $C[v, t, b] \leftarrow S_0[v, t, b] \wedge C[v, t, b]$
    if $(u, v) \in E$ **then**
      for all t_0 and b_0 **do**
        if there does not exist a control strategy for T_v such that $S[u, t_0, b_0] = 1$
        then $S_0[u, t_0, b_0] \leftarrow 0$
      for all t and b **do** $C[u, t, b] \leftarrow S_0[u, t, b] \wedge C[u, t, b]$
      Delete nodes in T_v (including v)
    else
      for all t_0 and b_0 **do**
        if there does not exist a control strategy for T_v such that $S[v, t_0, b_0] = 1$
        then $S_0[v, t_0, b_0] \leftarrow 0$
      for all t and b **do** $C[v, t, b] \leftarrow S_0[v, t, b] \wedge C[v, t, b]$
      Delete nodes in T_v (except v)

Based on the above procedure, we have the following where the proof is omitted here.

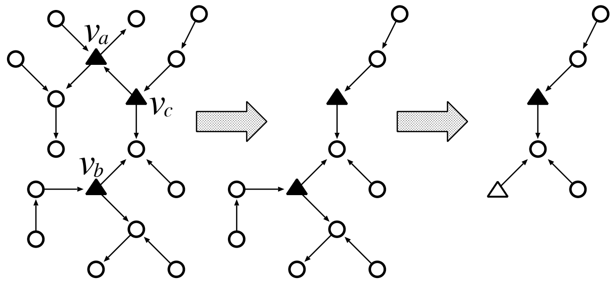


Figure 6. Illustration of the procedure for unrooted trees, where v_a, v_b and v_c are branching nodes. At the beginning, v_a and v_b are outmost branching nodes and $nb(v_a) = nb(v_b) = v_c$.

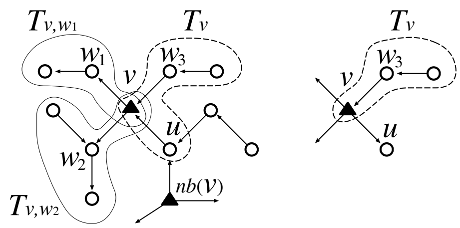


Figure 7. Example of $T_{v,w}$ and T_v. It should be noted that T_v includes u if $(u, v) \in E$ (left), whereas T_v does not include u if $(v, u) \in E$ (right). In both cases, $\Gamma^+(v) = \{w_1, w_2\}$ and $\Gamma^- = \{w_3\}$.

Theorem 4.2. *If a BN has a tree structure, both BN-CONTROL and partial BN-CONTROL can be solved in $O((n+m)M^2)$ time.*

The above algorithm may also be useful even if the network has a few loops. Suppose that the network becomes a forest if H nodes are removed. Though it is difficult to find the minimum H, a greedy approach may work well to find an appropriate H. Then, we examine all possible time series for these H nodes and apply the algorithm in Thm. 4.2. This tree-based method takes $O(2^{HM}(m+n)M^2)$ time. On the other hand, we can use the algorithm by Datta *et al.*[9] to solve BN-CONTROL and partial BN-CONTROL. Then, it will take $O(2^{2n+m}M)$ time. However, it is very time consuming even for small n (e.g., $n = 10$). Therefore, the tree-based method may be much more useful for BN-CONTROL and partial BN-CONTROL than the algorithm by Datta *et al.* when HM is small enough.

It should also be noted that the algorithm for trees can be extended for other discrete and finite domains. For that purpose, we modify $S[v_i, t, b]$ so that b takes values in the target domain and we replace Boolean functions with discrete functions for the domain.

5. Concluding Remarks

We have shown that finding a control strategy for Boolean networks is computationally very hard. Hardness results still hold for other models of biological systems if those can represent Boolean formula for 3SAT using control variables. Since close relationships between biological systems and Boolean circuits are suggested,[7,16,17] it seems difficult to find control strategies efficiently for all types of biological networks.

However, many biological sub-networks have special features. For example, Kitano[1,2] suggested that negative feedback loops play an important role in biological systems: these contribute to keeping robustness of biological systems. Such sub-networks are considered to be significantly different from the networks constructed in this paper because it seems impossible to describe negative and robust feedback loops using Boolean functions. Therefore, one of important future studies is to develop an efficient algorithm for finding control strategies for such robust sub-networks.

References

1. H. Kitano. Computational systems biology. *Nature*, 420:206–210, 2002.
2. H. Kitano. Cancer as a robust system: implications for anticancer therapy. *Nature Reviews Cancer*, 4:227–235, 2004.
3. S. A. Kauffman. *The Origins of Order: Self-organization and Selection in Evolution*. Oxford Univ. Press, 1993.
4. T. Akutsu, S. Miyano and S. Kuhara. Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics*, 16:727–734, 2000.
5. R. Albert and A-L. Barabási. Dynamics of complex systems: scaling laws for the period of Boolean networks. *Physical Review Letters*, 84:5660-5663, 2000.
6. L. A. Amaral, A. Diaz-Guilera, A. A. Moreira, A. K. Goldberger and L. A. Lipsitz. Emergence of complex dynamics in a simple model of signaling networks. *Proc. National Academy of Sciences USA*, 101:15551-15555, 2004.
7. S. E. Harris, B. K. Sawhill, A. Wuensche and S. Kauffman. A model of transcriptional regulatory networks based on biases in the observed regulation rules. *Complexity*, 7:23-40, 2002.
8. S. Liang, S. Fuhrman and R. Somogyi. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. *Proc. Pacific Symposium on Biocomputing*, 3:18–29, 1998.
9. A. Datta, A. Choudhary, M. L. Bittner and E. R. Dougherty. External control in Markovian genetic regulatory networks. *Machine Learning*, 52:169-191, 2003.
10. A. Datta, A. Choudhary, M. L. Bittner and E. R. Dougherty. External control in Markovian genetic regulatory networks: the imperfect information case. *Bioinformatics*, 20:924–930, 2004.
11. A. Datta, A. Choudhary, M. L. Bittner and E. R. Dougherty. Intervention in context-sensitive probabilistic Boolean networks. *Bioinformatics*, 21:1211-1218, 2005.
12. I. Shmulevich, E. R. Dougherty, S. Kim and W. Zhang. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18:261–274, 2002.
13. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., 1979.
14. G. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393-405, 1990.
15. P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction*. John Wiley and Sons Ltd., 2000.
16. H. H. McAdams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269:650–656, 1995.
17. C-H. Yuh, H. Bolouri and E. H. Davidson. Genomic Cis-regulatory logic: experimental and computational analysis of a sea urchin gene. *Science*, 279:1896–1902, 1998.