<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Finding optimal control policy in probabilistic Boolean networks with hard constraints by using integer programming and dynamic programming</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Chen, X; Akutsu, T; Tamura, T; Ching, WK</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>The 2010 IEEE International Conference on Bioinformatics and Biomedicine (BiBM 2010), Hong Kong, China, 18-21 December 2010, p. 240-246, abstract no B323</td>
</tr>
<tr>
<td><strong>Issued Date</strong></td>
<td>2010</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10722/158878">http://hdl.handle.net/10722/158878</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>IEEE International Conference on Bioinformatics and Biomedicine Proceedings. Copyright © IEEE; ©2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.; This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.</td>
</tr>
</tbody>
</table>
Abstract—In this paper, we study control problems of Boolean Networks (BNs) and Probabilistic Boolean Networks (PBNs). For BN CONTROL, by applying external control, we propose to derive the network to the desired state within a few time steps. For PBN CONTROL, we propose to find a control sequence such that the network will terminate in the desired state with a maximum probability. Also, we propose to minimize the maximum cost of the terminal state to which the network will enter. Integer linear programming and dynamic programming in conjunction with hard constraints are then employed to solve the above problems. Numerical experiments are given to demonstrate the effectiveness of our algorithms. We also present a hardness result suggesting that PBN CONTROL is harder than BN CONTROL.

Keywords—Boolean networks; probabilistic Boolean networks; optimal control; integer linear programming; dynamic programming

I. INTRODUCTION

Developing efficient algorithms for control of genetic regulatory networks is an important research issue in bioinformatics. A number of formalisms have been developed for modeling genetic regulation processes, such as Bayesian networks, multivariate Markov chain model [1], Boolean networks and probabilistic Boolean networks. A brief review can be found in [2]. Among all the models, Boolean networks (BNs) and their extension probabilistic Boolean networks (PBNs) have received much attention since they are able to capture the switching behavior of the genetic process.

Boolean network (BN) was first introduced by Kauffman [3] in 1969. It is a very simple model: each gene is quantized to only two levels – on and off (represented as 1 and 0). The target gene can be regulated by several genes called its input genes via its Boolean function (predictor function). A BN is said to be well defined if all the input genes and Boolean functions are given. However, a BN is a deterministic model. The randomness comes only from the initial state. Therefore, it is more realistic to extend a BN to a stochastic one, namely, Probabilistic Boolean Network (PBN). In a PBN, instead of having only one Boolean function, each gene can have multiple Boolean functions with selecting probabilities assigned to them. The dynamics of a PBN can be studied and analyzed by the theory of Markov chain. Furthermore, it is possible to control one or more genes in a network such that the whole network is derived into a desired state or a steady-state distribution. Then therapeutic gene intervention or gene control policy can be developed [4], [5], [6].

In this paper, we propose to solve the control problem of PBNs by using integer linear programming and dynamic programming in conjunction with hard constraints. In [7] an integer programming approach has been applied to solve the control problem of PBN. Here we consider adding hard constraints (i.e. adding an upper bound for the number of controls that can be applied to the network [6]) into the problem and propose an integer linear programming based method with hard constraints to solve the control problem of BN and PBN. Introduction of hard constraints is important for medical applications because the number of treatments such as radiation and chemo-therapy is usually limited [6]. Moreover, given the terminal cost for each state, we want to derive the network into the state with the minimized maximum cost by applying external control. This problem is important also for medical applications because we may want to minimize the damage or cost even in the worst case. In addition to development of algorithms, we study the time complexity of control problems for PBN. We prove that both minimizing the maximum cost and minimizing the average cost are \( \Sigma_2 \)-hard, where the latter problem corresponds to the original control problem for PBN [8]. It is known that control of BN is NP-complete\(^1\) and control of PBN is \( \Sigma_2 \)-hard [4]. Since it is believed that \( \Sigma_2 \)-hard problems are much harder than NP-complete problems [10], this result suggests that control of PBN is much harder than control of BN. Furthermore, this result suggests that such methods as integer linear programming cannot be effectively applied to solve the control problem of PBN because (a decision problem version of) integer linear programming is known

\(^1\)Control of BN is NP-complete if the number of time steps is polynomially bounded. Otherwise, it is PSPACE-complete, which directly follows from the result of [9]. However, it is not usual to consider an exponential number of time steps.
to be NP-complete [10]. Therefore, integer programming-based approach can only be applied to control of BN [5] and special restricted variants of control of PBN [7].

II. PROBLEMS

A. Boolean Networks and Probabilistic Boolean Networks

A Boolean network (BN) is represented by a set of nodes (genes) $V = \{v_1, v_2, \ldots, v_n\}$ and a list of Boolean functions $F = \{f_1, f_2, \ldots, f_n\}$ 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 $v_i$. We use $IN(v_i)$ to represent the set of input nodes $v_{i_1}, \ldots, v_{i_k}$ to $v_i$. The number of inputs to $v_i$ is called the indegree of $v_i$. We define $K$ as the maximum indegree of a BN.

We define $v_i(t)$ to be the state (0 or 1) of the gene $i$ at time $t$. The rules of the regulatory interactions among the genes can then be represented by Boolean functions:

$$v_i(t+1) = f_i(v_{i_1}(t), \ldots, v_{i_k}(t)), \quad i = 1, 2, \ldots, n. \quad (1)$$

Here we let $v(t) = (v_1(t), \ldots, v_n(t))^T$ which is called the Gene Activity Profile (GAP). The GAP can take any possible states from the set $S = \{v_1, v_2, \ldots, v_n\}^T : v_i \in \{0, 1\}$ and thus totally there are $2^n$ possible states in the network. We then define

$$z(t) = 1 + \sum_{i=1}^{n} 2^{n-i}v_i(t).$$

As $v_1(t)v_2(t) \ldots v_n(t)$ ranges from $00 \ldots 0$ to $11 \ldots 1$, $z(t)$ will take on all values from 1 to $2^n$. Clearly, there is a one-to-one map from $x(t)$ to $z(t)$. Hence instead of the binary representation for the global state, one can use equivalent decimal representation $z(t)$.

Since a BN is a deterministic model, it is more realistic to extend it to a probabilistic setting. To extend the concepts of a BN to a stochastic model, for each vertex $v_i$ in a PBN, instead of having only one Boolean function as in BN, there are a multiple of Boolean functions (predictor functions) $f^{(i)}_j(j = 1, 2, \ldots, l(i))$ to be chosen for determining the state of gene $v_i$ and usually $l(i)$ is not very large. The probability of choosing $f^{(i)}_j$ as the predictor function is $c^{(i)}_j$,

$$0 \leq c^{(i)}_j \leq 1 \quad \text{and} \quad \sum_{j=1}^{l(i)} c^{(i)}_j = 1 \quad \text{for} \quad i = 1, 2, \ldots, n. \quad (2)$$

To estimate the probability $c^{(i)}_j$, one of the efficient methods is called Coefficient of Determination (COD) [11]. It can be used to estimate the probability $c^{(i)}_j$ with real gene expression data sets.

We let $f_j$ be the $j$th possible realization, where $f_j = (f^{(1)}_{j_1}, f^{(2)}_{j_2}, \ldots, f^{(n)}_{j_n})$, $1 \leq j_i \leq l(i)$, $i = 1, 2, \ldots, n$. Suppose that the selection of the Boolean function $f_{j_i}$ for each gene $i$ is an independent process, then the probability of choosing the corresponding BN with Boolean functions $f_j = (f^{(1)}_{j_1}, f^{(2)}_{j_2}, \ldots, f^{(n)}_{j_n})$ is given by

$$q_{j_1j_2\ldots j_n} = \prod_{i=1}^{n} c^{(i)}_{j_i}. \quad (3)$$

There are at most $N = \prod_{i=1}^{n} l(i)$ different possible realizations of BNs. Fig. 1 gives an example of PBN. We note that the transition process among the states in the set $S$ forms a Markov chain process. Let $a$ and $b$ be any two column vectors in the set $S$. Then the transition probability

$$P \{v(t+1) = a \mid v(t) = b\} = \sum_{j=1}^{N} P \{v(t+1) = a \mid v(t) = b, \text{the } j \text{th BN is selected}\} \cdot q_j = \sum_{j \in I} q_j$$

where $I$ is the set of BNs of which the transition probability from state $b$ to state $a$ is 1.

Here we let

$$q_j = q_{j_1j_2\ldots j_n} \quad \text{and} \quad j = j_1 + \sum_{i=2}^{n} \left\lfloor \frac{j_i - 1}{l(i)} \right\rfloor \cdot \prod_{k=1}^{i-1} l(k).$$

We can then use both of them when there is no confusion. By letting $a$ and $b$ take all the possible states in $S$, one can get the transition probability matrix of the Markov chain (or the PBN). The transition probability matrix can be written as

$$A = \sum_{j=1}^{N} q_j A_j \quad (5)$$

where $A_j$ is the corresponding transition matrix of the $j$th BN and $q_j$ is the probability of choosing the $j$th BN. We remark that there are at most $N \cdot 2^n$ nonzero entries in the transition probability matrix $A$ and this means the matrix is sparse, i.e., having a lot of zero entries.
C. Finding the Optimal Path with Hard Constraints

Here we let \( t \)rol node. Fig. 2 gives an example of BN CONTROL.

\[ \mathbf{v}(t) = \{v_1(t), v_2(t), \ldots, v_n(t)\} \]

Let a set \( \mathbf{v} \) correspond to control nodes. Let a set \( \mathbf{u} \) correspond to usual nodes in a BN and external nodes \( \mathbf{y} \) correspond to usual nodes in a BN and external nodes. Then the states of internal nodes at time \( t+1 \) are represented by

\[ v_i(t+1) = f_i(v_{i_1}(t), \ldots, v_{i_k}(t)), \quad i = 1, 2, \ldots, n. \quad (6) \]

where each \( v_{i_j} \) is either an internal node or a control node. Fig. 2 gives an example of BN CONTROL. Here we let \( \mathbf{v}(t) = \{v_1(t), v_2(t), \ldots, v_n(t)\} \) and \( \mathbf{u}(t) = \{v_{n+1}(t), v_{n+2}(t), \ldots, v_{n+m}(t)\} \).

\[ v_{n+i}(t) - v_{n+i}(t+1) \neq 0, \text{ for some } i \in \{1, \ldots, m\} \]

then we say that the external control is applied once to the network. Thus the number of controls applied to network is equal to

\[ \sum_{t=0}^{M-1} \sum_{i=1}^{m} |v_{n+i}(t) - v_{n+i}(t+1)|. \]

Then the control problem of BN under hard constraints is as follows:

**Definition 1**: Suppose an initial state of the network is \( \mathbf{v}^0 \) and the desired state of the network is \( \mathbf{v}^M \), find a control sequence \( \{\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(M)\} \) such that \( \mathbf{v}(0) = \mathbf{v}^0 \) and \( \mathbf{v}(M) = \mathbf{v}^M \), and the maximum number of controls applied to the network during the finite time period \( M \) is \( H \).

B. Control of BN with Hard Constraints

Akutsu et al. introduced a control problem for BNs in [4]. In BN CONTROL, there are two types of nodes: *internal nodes* and *external nodes*, where internal nodes correspond to usual nodes in a BN and external nodes correspond to control nodes. Let a set \( \mathbf{V} \) of \( n \) nodes be \( \mathbf{V} = \{v_1, v_n, v_{n+1}, \ldots, v_{n+m}\} \), where \( v_1, \ldots, v_n \) are internal nodes and \( v_{n+1}, \ldots, v_{n+m} \) are control nodes. Then the states of internal nodes at time \( t+1 \) are represented by

\[ v_i(t+1) = f_i(v_{i_1}(t), \ldots, v_{i_k}(t)), \quad i = 1, 2, \ldots, n. \]

where \( f_i \) is either a state function or a control function.

\[ \mathbf{v}(t) = \{v_1(t), v_2(t), \ldots, v_n(t)\} \]

and \( \mathbf{u}(t) = \{v_{n+1}(t), v_{n+2}(t), \ldots, v_{n+m}(t)\} \).

**Definition 2**: Suppose an initial state of the network is \( \mathbf{v}^0 \) and the desired state of the network is \( \mathbf{v}^M \), find a control sequence \( \{\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(M)\} \) such that the probability of the path with the initial state \( \mathbf{v}^0 \) and the terminal (desired) state \( \mathbf{v}^M \) can be described as follows:

\[ z_t = 1 + \sum_{i=1}^{n} 2^{n-i} v_i \]

which is the state of network at time step \( t \), and

\[ u_t = 1 + \sum_{i=1}^{m} 2^{m-i} v_{n+i} \]

which is the control input of network at time step \( t \). In a PBN, even if the network starts with the given initial state \( \mathbf{v}(0) \), the subsequent states will be random since the PBN is a stochastic model. That is, the terminal state \( \mathbf{v}(M) \) could take any possible values from 1 to \( 2^n \). We assign a terminal cost \( C_M(\mathbf{z}_M) \) to each of states \( \mathbf{z}_M \) at time step \( M \). Note that, depending on the particular PBN and the control input used in each step, it is possible that the network can not enter some of the states at time step \( M \). We define \( C_t(z_t) \) as the maximum cost of which, beginning from \( z_t \) at time step \( t \), the network can reach at the terminal time step. The problem of minimizing the maximum cost can be described as follows:

**Definition 3**: Given the terminal cost \( C_M(\mathbf{z}_M) \) for each of states \( \mathbf{z}_M \in \{1, 2, \ldots, 2^n\} \) at time step \( M \), by applying external control, minimize the maximum cost \( C_0(z_0) \) beginning from the given initial state \( z_0 \), and the maximum number of controls applied to the network is \( H \).

III. ALGORITHMS

A. ILP with Hard Constraints for BN CONTROL

In order to give ILP formalization for BN CONTROL, we first introduce several definitions.

Let \( x_{i,t} \) represent the Boolean value \( v_i(t) \). Define

\[ \sigma_b(x) = \begin{cases} x, & \text{if } b = 1, \\ \bar{x}, & \text{otherwise}. \end{cases} \]

\[ \sigma_b(x) = \begin{cases} x, & \text{if } b = 1, \\ \bar{x}, & \text{otherwise}. \end{cases} \]

\[ \sigma_b(x) = \begin{cases} x, & \text{if } b = 1, \\ \bar{x}, & \text{otherwise}. \end{cases} \]
Then any Boolean function \( f_i(x_{i_1}, \ldots, x_{i_k}) \) is equivalent to
\[
f_i(x_{i_1}, \ldots, x_{i_k}) = \bigvee_{b_{i_1} \ldots b_{i_k} \in \{0,1\}^k} \left( f_i(b_{i_1}, \ldots, b_{i_k}) \land \sigma_{b_1}(x_{i_1}) \land \ldots \land \sigma_{b_k}(x_{i_k}) \right).
\] (8)

Then we define binary variable \( h_{i,t} \in \{0,1\} \) \((i = n + 1, \ldots, n + m)\) as the node control variable. If \( h_{i,t} = 1 \), we say the node \( i \) changes its value at time step \( t \). Since the maximum number of controls applied to the network during the finite time period is \( H \), we have
\[
\sum_{t=0}^{M-1} \sum_{i=n+1}^{n+m} h_{i,t} \leq H.
\]

Also, we define \( \tau_b(x) \) as
\[
\tau_b(x) = \begin{cases} 
  x, & \text{if } b = 1. \\
  1 - x, & \text{otherwise.}
\end{cases}
\] (9)

Then the ILP-Formulation for the BN CONTROL based on the method of [5] is as follows:

Maximize
\[
\sum_{i=1}^{N} x_{i,M}
\]
S.T.
\[
x_{i,t+1,b_{i_1} \ldots b_{i_k}} \geq \sum_{j \in \{1,2,\ldots,k\}} \tau_{b_j}(x_{ij,t}) - (k - 1) \tag{10}
\]
\[
x_{i,t+1,b_{i_1} \ldots b_{i_k}} \leq \frac{1}{k} \sum_{j \in \{1,2,\ldots,k\}} \tau_{b_j}(x_{ij,t}) \tag{11}
\]
for all \( i \in \{1,2,\ldots,n\}, \ t \in \{1,2,\ldots,M - 1\} \) and \( b_{i_1} \ldots b_{i_k} \in \{0,1\}^k \) such that \( f_i(b_{i_1}, \ldots, b_{i_k}) = 1 \).

\[
x_{i,t+1} = 0 \tag{12}
\]
for all \( i \in \{1,2,\ldots,n\}, \ t \in \{1,2,\ldots,M - 1\} \), and \( b_{i_1} \ldots b_{i_k} \in \{0,1\}^k \) such that \( f_i(b_{i_1}, \ldots, b_{i_k}) = 0 \).

\[
x_{i,t} \leq \sum_{b_{i_1} \ldots b_{i_k} \in \{0,1\}^k} x_{i,t,b_{i_1} \ldots b_{i_k}} \tag{13}
\]
\[
x_{i,t} \geq \frac{1}{2^k} \sum_{b_{i_1} \ldots b_{i_k} \in \{0,1\}^k} x_{i,t,b_{i_1} \ldots b_{i_k}} \tag{14}
\]
for all \( i \in \{1,2,\ldots,n\} \) and \( t \in \{1,2,\ldots,M\} \).

Note that \( x_{i,t} \) and \( x_{i,t,b_{i_1} \ldots b_{i_k}} \) and \( h_{i,t} \) are binary values. Here (10) to (14) are to make sure that \( x_{i,t+1} = f_i(x_{i_1,t+1}, \ldots, x_{i_k,t+1}) \) is satisfied for all \( i \in \{1,2,\ldots,n\} \) and \( t \in \{1,2,\ldots,M - 1\} \).

### B. ILP with Hard Constraints for PBN CONTROL

To extend the above ILP formulation for PBN CONTROL, we define \( y_{r,t} \) as the selection variable. If \( y_{r,t} = 1 \), we say the \( r \)-th BN is selected at time step \( t \). Otherwise, we say it is not selected at time step \( t \). Then we have
\[
\sum_{r=1}^{R} y_{r,t} = 1, \text{ for } t = 1,2,\ldots,M - 1
\]

Here \( R \) is the total number of possible realizations for the PBN. Define \( f_{i,r} \) as the Boolean function for node \( v_i \) when the \( r \)-th BN is selected. Let \( P = (p_1, p_2, \ldots, p_R) \) be the selecting probabilities for the \( R \) possible realizations.

Then the objective function for the PBN control with hard constraints is as follows:

Maximize
\[
\sum_{t=0}^{M-1} \sum_{r=1}^{R} - \log(p_r) \cdot y_{r,t}
\]

For the constraints, we revise (10) to (12) as follows:
\[
x_{i,t+1,b_{i_1} \ldots b_{i_k}} \geq \sum_{j \in \{1,2,\ldots,k\}} \tau_{b_j}(x_{ij,t}) - (k - 1) + y_{r,t} - 1 \tag{15}
\]
\[
x_{i,t+1,b_{i_1} \ldots b_{i_k}} \leq \frac{1}{k} \sum_{j \in \{1,2,\ldots,k\}} \tau_{b_j}(x_{ij,t}) - y_{r,t} + 1 \tag{16}
\]
for all \( i \in \{1,2,\ldots,n\}, \ t \in \{1,2,\ldots,M - 1\} \), and \( b_{i_1} \ldots b_{i_k} \in \{0,1\}^k \) and \( r \in \{1,2,\ldots,R\} \) such that \( f_{i,r}(b_{i_1}, \ldots, b_{i_k}) = 1 \).

\[
x_{i,t+1} \leq 1 - y_{r,t} \tag{17}
\]
for all \( i \in \{1,2,\ldots,n\}, \ t \in \{1,2,\ldots,M - 1\} \), and \( b_{i_1} \ldots b_{i_k} \in \{0,1\}^k \) and \( r \in \{1,2,\ldots,R\} \) such that \( f_{i,r}(b_{i_1}, \ldots, b_{i_k}) = 0 \).

Then we also add the following constraints:
\[
\sum_{r=1}^{R} y_{r,t} = 1, \text{ for } t = 1,2,\ldots,M - 1, \text{ and } y_{r,t} \in \{0,1\}
\]
for all \( t \in \{1,2,\ldots,M\} \) and \( r \in \{1,2,\ldots,R\} \).

The other constraints are the same as the ones in ILP-Formulation for the BN CONTROL.

Here (13) to (17) are to make sure that \( x_{i,t} \) is regulated by the chosen BN and its corresponding Boolean functions for all \( i \in \{1,2,\ldots,n\}, \ t \in \{1,2,\ldots,M - 1\} \) and \( r \in \{1,2,\ldots,R\} \).
C. Minimizing the Maximum Cost

Define \( J(z_t, h_t) \) as the minimized maximum terminal cost \( C_M(z_M) \) when the state is \( z_t \), and the remaining number of external controls is \( h_t \), at time step \( t \). Define \( u(z_t, h_t) \) as the control function when the state is \( z_t \) and the remaining number of external controls is \( h_t \) at time step \( t \). Let \( F(z_t, u_t) \) be the set of states at time step \( t + 1 \) that can be reached from \( z_t \) with control \( u_t \). Then dynamic programming for the PBN control with hard constraints is as follows:

**Step 0:** Set \( t = M; J(z_M, h_M) = C_M(z_M) \) for all \( h_M = \{0, \ldots, H\} \).

**Step 1:** \( t := t - 1 \).

**Step 2:** For any \( z_t \in \{1, \ldots, 2^n\} \) and \( h_t \in \{0, \ldots, H\} \), compute

\[
J(z_t, h_t) = \min_{u_t \in \{1, \ldots, 2^m\}} \left\{ \begin{array}{ll}
\max_{z_{t+1} \in F(z_t, u_t)} J(z_{t+1}, h_t), & \text{if } u_t = u_{t+1}, h_t, \\
\max_{z_{t+1} \in F(z_t, u_t)} J(z_{t+1}, h_t - 1), & \text{otherwise}.
\end{array} \right.
\]

and

\[
u(z_t, h_t) = \text{argmin}_{u_t \in \{1, \ldots, 2^m\}} \left\{ \begin{array}{ll}
\max_{z_{t+1} \in F(z_t, u_t)} J(z_{t+1}, h_t), & \text{if } u_t = u_{t+1}, h_t, \\
\max_{z_{t+1} \in F(z_t, u_t)} J(z_{t+1}, h_t - 1), & \text{otherwise}.
\end{array} \right.
\]

**Step 3:** If \( t > 0 \), go back to step 1; Otherwise, stop.

In the above, \( u_t \neq u_{t+1} \) is counted as one control where we need to modify the algorithm for the case that the number of controls is defined as before. Finally, we take \( \min_{h_0 \in \{0, \ldots, H\}} J(z_0, h_0) \) for computing the minimized maximum cost. Though we do not consider costs of control operations in this paper, it is possible to extend the algorithm so that costs of control operations are taken into account.

IV. COMPLEXITY ANALYSIS

In this section, we give some analysis on the complexity of minimizing the maximum cost and minimizing the average cost. In order to analyze the time complexity, we assume that a PBN is not given in the matrix form \( A \) of (5) but in the form of \( f_j \) and \( e_j \) because it is of exponential size and thus it is almost meaningless to discuss the time complexity if we use \( A \). Furthermore, we assume that it is only required to output \( u(0) \) for given \( z_0 \) and PBN (otherwise, we should output \( u(t) \) for an exponential number of GAPs). Then, we can keep both the sizes of input and output polynomial of \( n \) and thus can discuss the time complexity with respect to the network size. Furthermore, we assume that the number of time steps (i.e., \( M \)) is polynomially bounded. Otherwise, both BN CONTROL and PBN CONTROL would be PSPACE-hard [9], [12]. Since it is not realistic to consider an exponential number of time steps, this is a reasonable assumption.

For the complexity of minimizing the maximum cost, we have the following theorem by using a polynomial time reduction similar to that in [5].

**Theorem 1:** Minimizing the maximum cost in control of PBN is \( \Sigma_p^2 \)-hard.

**Proof:** As in [5], we use a polynomial time reduction from the quantified Boolean formula problem for 3-DNF (disjunction of conjunctions of which consisting 3 literals). Let \( \psi(x, y) \) be a 3-DNF over variables \( x = (x_1, \ldots, x_m) \) and \( y = (y_1, \ldots, y_n) \). Then it is known that deciding whether or not \( (\exists x)(\forall y)\psi(x, y) \) is true is \( \Sigma_p^2 \)-complete [10], [13]. We show a polynomial time reduction from this problem to minimization of the maximum cost.

From a given \( \psi(x, y) \), we construct a PBN as follows. Let \( n_2 \) be the number of terms in \( \psi(x, y) \). Then we let \( V = \{v_1, v_2, \ldots, v_{n_1+n_2+2m+1}\} \). For \( i = 1, \ldots, n_1 \), \( v_i \) corresponds to \( y_i \). For \( i = 1, \ldots, n_2 \), \( v_{n_1+i} \) corresponds to the \( i \)th term of \( \psi(x, y) \), where the \( i \)th term is represented as \( l_1 \land l_2 \land l_3 \). For \( i = 1, \ldots, m_1 \), both \( v_{n_1+n_2+1+i} \) and \( v_{n_1+n_2+2m_1+i} \) correspond to \( x_i \). Then, we assign the following Boolean functions to \( V \):

\[
v_i(t + 1) = v_i(t) \text{ with probability } 0.5, \quad \text{for } i = 1, \ldots, n_1, \\
v_i(t + 1) = v_i(t) \text{ with probability } 0.5, \quad \text{for } i = 1, \ldots, n_1, \\
v_{n_1+i}(t + 1) = l_1 \land l_2 \land l_3, \quad \text{for } i = 1, \ldots, n_2, \\
v_{n_1+n_2+1+i}(t + 1) = v_{n_1+n_2+1+i}(t), \\
\text{for } i = 1, \ldots, m_1, \\
v_{n_1+n_2+1+i}(t + 1) = v_{n_1+i}(t),
\]

where we identify \( v_i(t) \) with \( y_i \) and \( v_{n_1+n_2+1+i}(t) \) with \( x_i \) in \( l_1 \land l_2 \land l_3 \). Finally, we let \( n = n_1 + n_2 + m_1 + 1 \), \( m = m_1 \), \( v(0) = (0, 0, \ldots, 0) \), and \( M = 3 \). The cost is given by \( J = 0 \) if \( v_{n_1+n_2+1}(M) = 1 \). Otherwise \( J = 1 \).

Then we see that the minimum of the maximum cost is 0 iff \( (\exists x)(\forall y)\psi(x, y) \) is true. First, suppose \( (\exists x)(\forall y)\psi(x, y) \) is true for an assignment of \( x = (b_1, b_2, \ldots, b_m) \). Then, it is straight-forward to see that the maximum cost is 0 by the control input \( (v_{n_1+n_2+m_1+2}(0), \ldots, v_{n_1+n_2+2m_1+1}(0)) = (b_1, b_2, \ldots, b_m) \).

Next, suppose that the minimum of the maximum cost is 0. Since the minimum cost \( J \) is only determined by \( v_{n_1+n_2+1}(M) \) and \( J = 0 \) must hold, \( v_{n_1+n_2+1}(M) = 1 \) holds for any assignment on \( v_1(1), \ldots, v_{n_1}(1) \). Therefore, \( (\exists x)(\forall y)\psi(x, y) \) is true.

We modify the proof of Theorem 1 for proving \( \Sigma_p^2 \)-hardness of the original control problem of PBN (i.e., minimizing the average cost). For that purpose, we use the same reduction as in the proof of Theorem 1. Then, we note

\[2\text{We need to modify the algorithm if there exist multiple } u_t \text{s giving the minimum cost.}\]
Table I
THE RESULTS OF EXPERIMENT 1

<table>
<thead>
<tr>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILP method</td>
<td>0.38</td>
<td>4.22</td>
<td>3.66</td>
<td>0.69</td>
<td>6.54</td>
</tr>
<tr>
<td>DP method</td>
<td>20.81</td>
<td>5.93</td>
<td>5.75</td>
<td>7.77</td>
<td>9.53</td>
</tr>
</tbody>
</table>

Table II
THE RESULTS OF EXPERIMENT 2

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum cost</td>
<td>245</td>
<td>252</td>
<td>249</td>
<td>244</td>
<td>239</td>
</tr>
<tr>
<td>Minimized cost</td>
<td>193</td>
<td>194</td>
<td>177</td>
<td>173</td>
<td>163</td>
</tr>
</tbody>
</table>

that for any x, the average cost is 0 iff \( (\forall y)\psi(x, y) \) is true (i.e., the average cost is greater than 0 iff \( \psi(x, y) \) is false for some y). By utilizing this observation, we can prove the following:

**Theorem 2:** Minimizing the average cost in control of PBN is \( \sum_2^n \)-hard.

V. COMPUTATIONAL EXPERIMENTS

In this section, we give some computational experiments to demonstrate the effectiveness of our algorithms. Computational experiments were done on a PC with an Intel core 2 Duo CPU (p8600 2.4GHz) and 2GB RAM running under the Windows XP professional 2002 operating system. For solving integer linear programs, we used ILOG CPLEX (version 11.2, http://www.ilog.com/products/cplex/) with a LINUX (version 2.6.16) operating system. For the Windows XP professional 2002 operating system. For computational experiments were done on a PC with an Intel core 2 Duo CPU (p8600 2.4GHz) and 2GB RAM running under the Windows XP professional 2002 operating system. For solving integer linear programs, we used ILOG CPLEX (version 11.2, http://www.ilog.com/products/cplex/) with a LINUX (version 2.6.16) operating system.

A. Experiment 1: Comparison of ILP and DP for the Optimal Path Problem

For comparison of ILP and DP methods, we consider a 10-gene example with \( K = 3 \), where genes \( v_1, \ldots, v_8 \) are internal nodes and genes \( u_1, u_2 \) are control nodes. The PBN is consisting of 8 possible BNs with the selecting probabilities given by

\[
q = (0.1, 0.1, 0.3, 0.05, 0.05, 0.2, 0.15, 0.05).
\]

Let \( M = 5 \) and \( H = 3 \). Suppose the initial state is \( 00...0 \) and the desired state is \( 11...1 \). We conducted the experiments five times with different sets of Boolean functions which were randomly generated. By using the ILP method given in Section III-B and the revised DP method for finding the optimal path, we got the same results. Table I gives the CPU time (sec) for both methods. From the table, we see that the ILP method is faster than the DP method though PCs used for ILP and DP are not the same.

B. Experiment 2: Minimizing the Maximum Cost

For evaluating the DP method presented in Section III-C, we apply the DP method to minimize the maximum cost for a 10-gene example with 8 internal nodes and 2 control nodes. The PBN is consisting of 4 possible BNs and the Boolean functions are randomly generated. Suppose the initial state is \( (0, 0, \ldots, 0) \), and the terminal cost is given by \( C_M(z_M) = z_M \). Let \( M = 3 \) and \( H = 2 \). We conducted the experiments five times. The results are given in Table II.

C. Experiment 3: On Realistic Network

We also conducted an experiment based on a 10-gene WNT5A network (see Fig. 3, which is extracted from [14]). There are 10 nodes in this network, we assume WNT5A, pirin, MART-1, synnclein, PHO-C, S100P, STC2 and MMP-3 are internal nodes, and RET-1 and HADHB are control nodes. The initial state is \( (0, 0, \ldots, 0) \), and the desired state is \( (1, 1, \ldots, 1) \). Let \( M = 10 \) and \( H = 3 \). The Boolean functions are randomly generated and the PBN is consisting of eight possible BNs with the following selection probabilities:

\[
q = (0.0011491, 0.2165696, 0.2018161, 0.0740315, 0.0587344, 0.0304030, 0.1348942, 0.2824019).
\]

By using ILP with hard constraints, the probability of optimal path is \( 1.2306 \times 10^{-8} \). The external control is applied three times to the network.

Then we consider minimizing the maximum cost for the initial state \( (0, 0, \ldots, 0) \). Also, the terminal cost is given by \( C_M(z_M) = z_M \). Let \( M = 3 \) and \( H = 2 \). Without applying external control, the maximum cost was 256. By using DP method, the maximum cost was reduced to 172.

VI. CONCLUSION

In this paper, we have presented ILP-based methods for control of BN and for finding an optimal path for PBN both with hard constraints. We have also presented a DP-based method for finding a control policy that minimizes the maximum cost for PBN under hard constraints, where it uses exponential size tables. The results of preliminary computational experiments suggest that for finding an optimal path for PBN, the ILP-based method is faster than the DP-based method (specialized for this problem). However, the hardness results presented in this paper suggest that...
ILP cannot be effectively applied to minimization of the maximum or average cost for PBN.

ACKNOWLEDGMENT

The work of TA was partially supported by MEXT Grant-in-Aid No. 22240009. The work of WC was partially supported by HKGRF Grant No. 7017/07P and HKU CRCG Grants and HKU strategic theme grant on computational sciences.

REFERENCES


