<table>
<thead>
<tr>
<th>Title</th>
<th>Design of Probabilistic Boolean Networks Based on Network Structure and Steady-State Probabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Kobayashi, Koichi; Hiraishi, Kunihiko</td>
</tr>
<tr>
<td>Citation</td>
<td>IEEE transactions on neural networks and learning systems, 28(8), 1966-1971</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2017-08</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2115/67196">http://hdl.handle.net/2115/67196</a></td>
</tr>
<tr>
<td>Rights</td>
<td>© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.</td>
</tr>
<tr>
<td>Type</td>
<td>article (author version)</td>
</tr>
<tr>
<td>File Information</td>
<td>PBN_IEEE_TNNLS_final.pdf</td>
</tr>
</tbody>
</table>
Design of Probabilistic Boolean Networks Based on Network Structure and Steady-State Probabilities

Koichi Kobayashi, Member, IEEE, and Kunihiko Hiraishi, Member, IEEE

Abstract—In this paper, we consider the problem of finding a probabilistic Boolean network (PBN) based on network structure and desired steady-state properties. In systems biology and synthetic biology, such problems are important as an inverse problem. Using a matrix-based representation of PBNs, a solution method for this problem is proposed. The problem of finding a Boolean network (BN) has been studied so far. In the problem of finding a PBN, we must calculate not only Boolean functions, but also the probabilities of selecting a Boolean function and the number of candidates of Boolean functions. Hence, the problem of finding a PBN is more difficult than that of finding a BN. The effectiveness of the proposed method is presented by numerical examples.

Index Terms—gene regulatory network, network structure, probabilistic Boolean network, systems biology.

I. INTRODUCTION

One of the aims in systems biology is to develop a method for modeling, analysis, and control of gene regulatory networks. Especially, control of gene regulatory networks corresponds to therapeutic interventions, and is important toward developing gene therapy technologies (see, e.g., [27]) in the future. In recent years, feedback control of synthetic biological circuits has been implemented in [24], and the experimental result in which cellular behavior is regulated by control has been obtained. Furthermore, in synthetic biology, a method to implement logic circuits using biological networks has been extensively studied [9], [10]. Then, a gene regulatory network satisfying some properties is designed. Motivated by the above backgrounds, we consider the problem of finding a mathematical model of a gene regulatory network under constraints on desired properties.

A gene regulatory network is in general modeled by ordinary/partial differential equations with high nonlinearity and high dimensionality. In order to deal with such a system, it is important to consider a simple model, and various models such as Bayesian networks, Boolean networks (BNs), hybrid systems (piecewise affine models), and Petri nets have been developed so far (see, e.g., [13] for further details). In control problems, a BN is frequently used [2], [16], [17], [21], [22]. In a BN, dynamics such as interactions between genes are expressed by Boolean functions [14], that is, gene expression is expressed by a binary value (0 or 1). There is a criticism that a BN is too simple as a model of gene regulatory networks (see, e.g., [25]), but a BN can handle more large-scale systems than other models such as piecewise affine models. In addition, since the behavior of gene regulatory networks is stochastic by effect of noise, it is appropriate that a Boolean function is randomly chosen at each time among the candidates of Boolean functions. Thus, a probabilistic BN (PBN) has been proposed in [28].

In this paper, we consider the problem of finding a PBN based on network structure and desired steady-state properties. Network structure implies the directed graph expressing interactions between genes. Its inference is relatively easier than that of Boolean functions, and has been extensively studied (see, e.g., [1], [8], [11]). Furthermore, in BNs, attractors such as fixed points and limit cycles represent cell types or states of cells, and are important for understanding the biological property [15]. In [12], [18], [26], the problem of finding a BN with the desired fixed points has been studied as an inverse problem. Instead of attractors, we focus on a steady-state probability distribution in PBNs. In the problem of finding a PBN, we must calculate not only Boolean functions, but also the probabilities of selecting a Boolean function and the number of candidates of Boolean functions. In [7], generation of a PBN from a given transition probability matrix has been studied, but a steady-state probability distribution has been not focused on. In [30], the problem of finding a PBN has been studied, but it is assumed that Boolean functions and the number of their candidates are given in advance. Hence, a solution method for this problem is not a straightforward extension of the existing methods, and a new solution method must be considered.

In the proposed solution method, we use a matrix-based representation for PBNs [19], [20]. This representation is an extension of that for BNs [18], and can be obtained from truth tables and selection probabilities. A matrix-based representation obtained by the semi-tensor product (STP) method [4], [5], [6], [23] can be also used. However, since manipulation of matrices using the STP method is not needed in this paper, we use a simple matrix-based representation based on truth tables. Then, the expected value of the state in the PBN is given by a probability-weighted sum of matrices. First, we consider the special case where the number of candidates of Boolean functions is given. Then, the problem of finding a PBN can be rewritten as a mixed integer linear programming problem. Next, we consider the general case. In this case, the solution method consists of two steps. In the first step, a probability-weighted sum of matrices is derived by solving a linear programming problem. In the next step, selection probabilities,
matrices expressing Boolean functions, and the number of candidates are derived by a certain algorithm proposed in this paper. The effectiveness of the proposed method is presented by two numerical examples.

**Notation:** For the $n$-dimensional vector $x = [x_1 \ x_2 \ \cdots \ x_n]^\top$ and the index set $\mathcal{I} = \{i_1, i_2, \ldots, i_m\} \subseteq \{1, 2, \ldots, n\}$, define $[x_i]_{i \in \mathcal{I}} := [x_{i_1} \ x_{i_2} \ \cdots \ x_{i_m}]^\top$. For two matrices $A$ and $B$, let $A \otimes B$ denote the Kronecker product of $A$ and $B$. In addition, for $q$ vectors $y_1, y_2, \ldots, y_q$ and the index set $\mathcal{J} = \{j_1, j_2, \ldots, j_p\} \subseteq \{1, 2, \ldots, q\}$, define $\bigotimes_{j \in \mathcal{J}} y_j := y_{j_1} \otimes y_{j_2} \otimes \cdots \otimes y_{j_p}$. For example, for two two-dimensional vectors $z_1, z_2, \ldots, z_q$ and $\mathcal{J} = \{1, 5\}$, we can obtain $\bigotimes_{j \in \mathcal{J}} z_j = z_1 \otimes z_5 = [z_1^{(1)} z_5^{(1)} \ z_1^{(2)} z_5^{(2)} z_1^{(2)} z_5^{(2)}]^\top$, where $z_j^{(i)}$ is the $i$-th element of $z_j$. Finally, let $1_{m \times n}$ denote the $m \times n$ matrix whose elements are all one.

II. Preliminaries

As preliminaries, we explain a probabilistic Boolean network (PBN) [28] and a matrix-based representation for a PBN [19], [20].

**A. Probabilistic Boolean Networks**

First, we explain a (deterministic) Boolean network (BN). A BN is defined by

\[
\begin{align*}
    x_1(k+1) &= f_1^{(1)}(x_j(k))_{j \in \mathcal{N}_1^{(1)}}, \\
    x_2(k+1) &= f_1^{(2)}(x_j(k))_{j \in \mathcal{N}_1^{(2)}}, \\
    &\vdots \\
    x_n(k+1) &= f_1^{(n)}(x_j(k))_{j \in \mathcal{N}_1^{(n)}},
\end{align*}
\]

(1)

where $x := [x_1 \ x_2 \ \cdots \ x_n]^\top \in \{0, 1\}^n$ is the state, and $k = 0, 1, 2, \ldots$ is the discrete time. The set $\mathcal{N}_1^{(i)} \subseteq \{1, 2, \ldots, n\}$ is a given index set, and the function $f_i : \{0, 1\}^{\mathcal{N}_1^{(i)}} \rightarrow \{0, 1\}$ is a given Boolean function consisting of logical operators such as AND ($\land$), OR ($\lor$), and NOT ($\neg$). If $\mathcal{N}_1^{(i)} = \emptyset$ holds, then $x_i(k+1)$ is uniquely determined as 0 or 1.

Next, we explain a probabilistic Boolean network (PBN) (see [28] for further details). In a PBN, the candidates of $f_i^{(i)}$ are given, and for each $x_i$, selecting one Boolean function is probabilistically independent at each time. Let

\[
    f_i^{(i)}(x_j(k))_{j \in \mathcal{N}_1^{(i)}}, \quad i = 1, 2, \ldots, q(i)
\]

denote the candidates of $f_i^{(i)}$. The probability that $f_i^{(i)}$ is selected is defined by

\[
    c_i^{(i)} := \text{Prob} \left( f_i^{(i)} = f_i^{(i)} \right).
\]

Then, the following relation

\[
    \sum_{i=1}^{q(i)} c_i^{(i)} = 1
\]

(2)

must be satisfied. Probabilistic distributions are derived from experimental results. Finally, $\mathcal{N}_i, i = 1, 2, \ldots, n$ are defined by

\[
    \mathcal{N}_i := \bigcup_{i=1}^{q(i)} \mathcal{N}_i^{(i)}.
\]

**B. Matrix-based Representation for PBNs**

In this subsection, we explain the outline of the matrix-based representation for PBNs [19], [20]. Instead of this

---

**Fig. 1.** State transition diagram.

We present a simple example. **Example 1:** Consider the PBN in which Boolean functions and probabilities are given by

\[
\begin{align*}
    f_1^{(1)} &= x_3(k), \quad c_1^{(1)} = 0.8, \\
    f_2^{(1)} &= \neg x_3(k), \quad c_2^{(1)} = 0.2, \\
    f_1^{(2)} &= x_1(k) \land \neg x_3(k), \quad c_1^{(2)} = 1.0, \\
    f_2^{(2)} &= x_2(k) \lor x_3(k), \quad c_2^{(2)} = 0.7, \\
    f_3^{(1)} &= x_1(k) \land \neg x_2(k), \quad c_1^{(3)} = 0.3, \\
    f_3^{(2)} &= x_2(k), \quad c_2^{(3)} = 0.3,
\end{align*}
\]

where $q(1) = 2, q(2) = 1$ and $q(3) = 2$ hold, $\mathcal{N}_1 = \{3\}$, $\mathcal{N}_2 = \{1, 3\}$, and $\mathcal{N}_3 = \{1, 2\}$ hold, and we see that the relation (2) is satisfied. Next, consider the state trajectory. Then, for $x(0) = [0 \ 0 \ 0]^\top$, we obtain

\[
\begin{align*}
    \text{Prob} \,(x(1) = [0 \ 0 \ 0]^\top | x(0) = [0 \ 0 \ 0]^\top) &= 0.8, \\
    \text{Prob} \,(x(1) = [1 \ 0 \ 0]^\top | x(0) = [0 \ 0 \ 0]^\top) &= 0.2.
\end{align*}
\]

In this example, the cardinality of the finite state set $\{0, 1\}^3$ is given by $2^3 = 8$, and we obtain the state transition diagram of Fig. 1 by computing the transition from each state. In Fig. 1, the number assigned to each node denotes $x_1, x_2, x_3$ (elements of the state), and the number assigned to each arc denotes the transition probability from some state to other state. Note here that for simplicity, the state transition from only $x(k) = [0 \ 0 \ 0]^\top$, $[0 \ 0 \ 1]^\top$, $[0 \ 1 \ 0]^\top$, $[1 \ 1 \ 0]^\top$ is illustrated in Fig. 1.

Finally, let $E[x_i(k)]$ denote the expected value of $x_i(k)$. The condition $x(0) = x_0$ should be described, but for simplicity of notation, it is omitted. Then, we define steady-state probabilities for PBNs.

**Definition 1:** For a given PBN, $s_i$ is called a steady-state probability of $x_1$ if $s_i = E[x_i(k+1)] = E[x_i(k)]$ holds.

In the PBN, $E[x_i(k)]$ is equivalent to the probability that $x_i(k)$ is equal to 1. Hence, using $E[x_i(k)]$, steady-state probabilities are defined.
representation, we may use the matrix-based representation proposed in [4], [5], [6], [23], where the semi-tensor product (STP) of matrices is used. Since in this paper, manipulation of matrices using the STP is not needed, we use a simple matrix-based representation based on truth tables.

First, we define the notation. Binary variables \( x_i^0(k) \) and \( x_i^1(k) \) are introduced. If \( x_i(k) = 0 \) holds, then \( x_i^0(k) = 1 \) holds, otherwise \( x_i^0(k) = 0 \) holds. If \( x_i(k) = 1 \) holds, then \( x_i^1(k) = 1 \) holds, otherwise \( x_i^1(k) = 0 \) holds. Then, the equality \( x_i^0(k) + x_i^1(k) = 1 \) is satisfied.

Using \( x_i^0(k) \) and \( x_i^1(k) \), Consider transforming the BN (1) into a matrix-based representation. Define

\[
\overline{x}_i(k) := \begin{bmatrix} x_i^0(k) \\ x_i^1(k) \end{bmatrix} = \begin{bmatrix} 1 - x_i(k) \\ x_i(k) \end{bmatrix}.
\]

Then, the matrix-based representation for \( x_i(k+1) \) is given by

\[
\overline{x}_i(k+1) = A(i) \otimes \overline{x}_j(k),
\]

(3)

where \( A(i) \in \{0,1\}^{2 \times 2^{|N_i|}} \) and \( \otimes_{j \in N(i)} \overline{x}_j(k) \in \{0,1\}^{2^{|N(i)|}} \). The matrix \( A(i) \) can be derived from the following procedure [18].

**Procedure for deriving \( A(i) \) in (3):**

**Step 1:** Derive a truth table for \( x_i(k+1) \).

**Step 2:** Based on the obtained truth table, assign \( x_i(k+1) = 0 \) or \( x_i(k+1) = 1 \) for each element of \( \otimes_{j \in N(i)} \overline{x}_j(k) \).

**Step 3:** Express the assignment obtained in Step 2 by a row vector. Denote the obtained row vector by \( A(i) \in \{0,1\}^{1 \times 2^{|N(i)|}} \).

**Step 4:** Derive \( A(i) \) as

\[
A(i) = \begin{bmatrix} 1_{1 \times 2^{|N_i|}} - A(i) \\ A(i) \end{bmatrix}.
\]

Next, consider extending the matrix-based representation of BNs to that of PBNs. Noting that the probability distribution of each \( \overline{x}_i(k) \) is independent, \( E[\overline{x}_i(k+1)] \) can be obtained as

\[
E[\overline{x}_i(k+1)] = \sum_{l=1}^{q(i)} c_i^{(l)} A(i)^{l} \otimes E[\overline{x}_j(k)],
\]

(4)

where \( A(i)^{l} \in \{0,1\}^{2 \times 2^{|N_i|}} \) and \( \otimes_{j \in N_i} \overline{x}_j(k) \in \{0,1\}^{2^{|N_i|}} \). The matrix \( A(i)^{l} \) can be derived from the above procedure.

We present a simple example.

**Example 2:** Consider the PBN in Example 1. Using the matrix-based representation, the expected value of \( \overline{x}_i(k+1) \) can be obtained as

\[
E[\overline{x}_i(k+1)] = \begin{bmatrix} 0.8 \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{bmatrix} A^{(2)} A^{(3)} \times \begin{bmatrix} E[x_2^0(k)] \\ E[x_2^1(k)] \\ E[x_3^0(k)] \\ E[x_3^1(k)] \end{bmatrix},
\]

where the matrices \( A^{(1)} \) and \( A^{(2)} \) correspond to the Boolean functions \( f^{(1)}_1 \) and \( f^{(2)}_1 \), respectively. In a similar way, \( A^{(2)} \), \( A^{(3)} \), and \( A^{(3)} \) correspond to the Boolean functions \( f^{(2)}_1 \), \( f^{(3)}_1 \), and \( f^{(3)}_2 \), respectively.

**III. PROBLEM FORMULATION**

Consider the following problem that the PBN is found based on the steady-state probabilities.

**Problem 1:** For the PBN, suppose that the candidates of Boolean functions \( f^{(l)}_i \), \( l = 1, 2, \ldots, q(i) \), are given (if there is no Boolean function given, \( r(i) \leq q(i) \)) and the index sets \( N_i \), \( i = 1, 2, \ldots, n \) are given. In addition, suppose also that the steady-state probabilities are given by \( s_i \), \( i = 1, 2, \ldots, n \). Then, find probabilities \( c_i^{(l)} \), \( l = 1, 2, \ldots, q(i) \), Boolean functions \( f^{(l)}_i \), and the number of candidates \( q(i) \) such that \( E[\overline{x}_i] = s_i \) is satisfied in the steady state.

The index set \( N_i \), \( i = 1, 2, \ldots, n \) corresponds to the network structure, i.e., the directed graph expressing interactions between genes in the PBN. Inference of the network structure is relatively easier than that of Boolean functions (see, e.g., [1], [8], [11]). Furthermore, attractors in BNs represent cell types or states of cells, and are important for understanding the biological property [15]. Instead of attractors in BNs, we focus on a steady-state probability distribution in PBNs. We remark here that in Problem 1, a part of Boolean functions may be given based on previous knowledge.

Using a simple example, we explain how to give \( s_i \).

**Example 3:** Consider the PBN in Example 1. Suppose that setting \( x_1 \) to a small value in the steady state is desirable. Then, \( s_1 = 0 \), \( s_2 = 0.5 \), and \( s_3 = 0.5 \) can be set as one of the examples of suitable \( s_i \). In other words, the desired steady state is given by \( [0 0 0] \) with the probability 0.25, \( [0 0 1] \) with the probability 0.25, \( [0 1 0] \) with the probability 0.25, and \( [0 1 1] \) with the probability 0.25.

**IV. SOLUTION METHOD**

Based on the matrix-based representation, consider solving Problem 1. Using (4), Problem 1 can be rewritten as the
following problem.

**Problem 2:** For the PBN, suppose that matrices expressing Boolean functions \( f_l^{(i)} \), \( l = 1, 2, \ldots, r(i) \) (\( r(i) \leq q(i) \)) are given by \( A_l^{(i)} \). In addition, suppose also that the steady-state probabilities are given by \( s_i, i = 1, 2, \ldots, n \). Then, find probabilities \( c_l^{(i)}, l = 1, 2, \ldots, q(i) \), matrices \( A_l^{(i)} \), \( l = r(i) + 1, 2, \ldots, q(i) \), and the number of candidates \( q(i) \) satisfying the following condition:

\[
\pi_i = \left( \sum_{l=1}^{r(i)} c_l^{(i)} A_l^{(i)} + \sum_{l=r(i)+1}^{q(i)} c_l^{(i)} A_l^{(i)} \right) \otimes \pi_j
\]  

(5)

where \( \pi_i := [1 - s_i, s_i]^T \).

**A. Special Case**

First, consider the special case where \( q(i) \) is given in advance. Then, Problem 2 can be rewritten as a mixed integer linear programming (MILP) problem, where \( c_l^{(i)} \) is a continuous decision variable, and each element of \( A_l^{(i)} \) is a binary decision variable. We may add a dummy objective function.

We present a simple example.

**Example 4:** For the matrix-based representation the PBN in Example 2, consider the case of \( r(i) = 1 \) and \( q(i) = 3 \). Matrices \( A_1^{(1)}, A_2^{(2)}, \) and \( A_3^{(3)} \) are given by matrices obtained in Example 2, respectively. Then, (5) can be obtained as

\[
\pi_1 = \left( c_1^{(1)} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} + c_2^{(1)} \begin{bmatrix} 1 - \delta_{21}^{(1)} & 1 - \delta_{22}^{(1)} \\ \delta_{21}^{(1)} & \delta_{22}^{(1)} \end{bmatrix} A_1^{(1)} + c_3^{(1)} \begin{bmatrix} 1 - \delta_{31}^{(1)} & 1 - \delta_{32}^{(1)} \\ \delta_{31}^{(1)} & \delta_{32}^{(1)} \end{bmatrix} A_2^{(1)} \right) \otimes \pi_2,
\]

\[
\pi_2 = \left( c_1^{(2)} \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} A_1^{(2)} + c_2^{(2)} \begin{bmatrix} 1 - \delta_{21}^{(2)} & 1 - \delta_{22}^{(2)} \\ \delta_{21}^{(2)} & \delta_{22}^{(2)} \end{bmatrix} A_2^{(2)} + c_3^{(2)} \begin{bmatrix} 1 - \delta_{31}^{(2)} & 1 - \delta_{32}^{(2)} \\ \delta_{31}^{(2)} & \delta_{32}^{(2)} \end{bmatrix} A_3^{(2)} \right) \otimes \pi_3,
\]

\[
\pi_3 = \left( c_1^{(3)} \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} A_1^{(3)} + c_2^{(3)} \begin{bmatrix} 1 - \delta_{21}^{(3)} & 1 - \delta_{22}^{(3)} \\ \delta_{21}^{(3)} & \delta_{22}^{(3)} \end{bmatrix} A_2^{(3)} + c_3^{(3)} \begin{bmatrix} 1 - \delta_{31}^{(3)} & 1 - \delta_{32}^{(3)} \\ \delta_{31}^{(3)} & \delta_{32}^{(3)} \end{bmatrix} A_3^{(3)} \right) \otimes \pi_4,
\]

where \( c_l^{(i)} \) is a continuous decision variable with the constraints \( 0 \leq c_l^{(i)} \leq 1 \) and \( \sum_{l=1}^{q(i)} c_l^{(i)} = 1 \), and \( \delta_{ij}^{(l)} \) is a binary decision variable. In the above expressions, the product of a continuous decision variable and a binary decision variable (e.g., \( c_2^{(1)} \delta_{21}^{(1)} \)) is included. By the method in [3] (see Appendix A for further details), this product can be transformed into a linear inequality. Hence, in the case where \( q(i) \) is given in advance, Problem 2 can be equivalently rewritten as an MILP problem.

**B. General Case**

Consider the general case where \( q(i) \) is not given in advance. Then, it is difficult to directly rewrite Problem 2 as a certain optimization problem. Here, consider rewriting (5) as

\[
\pi_i = \left( \sum_{l=1}^{r(i)} c_l^{(i)} A_l^{(i)} + \tilde{B}^{(i)} \right) \otimes \pi_j
\]

where each element of \( B(i) \) is a continuous decision variable, and takes a value in the interval \([0, 1 - \sum_{j=1}^{r(i)} c_j^{(i)}]\). In addition, a sum of elements in each column of \( B(i) \) is equal to \( 1 - \sum_{j=1}^{r(i)} c_j^{(i)} \).

Then, consider the following problem.

**Problem 3:** For the PBN, suppose that matrices expressing Boolean functions \( f_l^{(i)} \), \( l = 1, 2, \ldots, r(i) \) (\( r(i) \leq q(i) \)) are given by \( A_l^{(i)} \). In addition, suppose also that the steady-state probabilities are given by \( s_i, i = 1, 2, \ldots, n \). Then, find probabilities \( c_l^{(i)}, l = 1, 2, \ldots, q(i) \), matrices \( B(i) \) satisfying the condition (7).

By a simple calculation, Problem 3 can be rewritten as a linear programming (LP) problem. Feasibility of Problem 3 and uniqueness of the solution are not guaranteed in general.

To guarantee uniqueness, introducing a suitable objective function is important. This topic is one of the future efforts.

We present a simple example.

**Example 5:** For (6) in Example 4, consider the case of \( r(1) = 1 \). Then, (7) for \( i = 1 \) can be obtained as

\[
\pi_1 = \left( c_1^{(1)} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} v_{211}^{(1)} \\ v_{221}^{(1)} \end{bmatrix}, v_{212}^{(1)} \right) \otimes \pi_2,
\]

where \( c_1^{(1)}, v_{211}^{(1)}, v_{221}^{(1)}, \) and \( v_{212}^{(1)} \) are a continuous decision variable with the constraints \( 0 \leq c_1^{(1)} \leq 1, 0 \leq v_{211}^{(1)} \leq 1 \leq c_1^{(1)}, v_{211}^{(1)} + v_{221}^{(1)} = 1 - c_1^{(1)}, \) and \( v_{212}^{(1)} + v_{222}^{(1)} = 1 - c_1^{(1)} \). Hence, (7) can be expressed as a linear equality constraint.

Using \( B(i) \) obtained by solving Problem 3, the parameters/matrices \( c_l^{(i)} \), \( A_l^{(i)} \), \( l = r(i) + 1, r(i) + 2, \ldots, q(i) \), and
We also suppose that \( q(i) \) can be derived by the following algorithm.

**Procedure for deriving** \( c_1(i), A_1(i), \) and \( q(i) \) from \( \hat{B}(i) \):

1. **Step 1:** Set \( q(i) := r(i) \).
2. **Step 2:** Set \( q(i) := q(i) + 1 \).
3. **Step 3:** Find an element that has the smallest value from \( \hat{B}(i) \). The smallest value obtained corresponds to \( c_q(i) \).
4. **Step 4:** Generate a matrix \( \tilde{A}_1(i) = \{0, 1\}^{2 \times 2 |N_i|} \) satisfying the following conditions: (i) a sum of elements in each column is equal to 1, (ii) all elements of \( \hat{B}(i) - c_q(i) \tilde{A}_1(i) \) are non-negative, (iii) at least one element of \( \hat{B}(i) - c_q(i) \tilde{A}_1(i) \) is equal to zero.
5. **Step 5:** Update \( \tilde{B}(i) := \tilde{B}(i) - c_q(i) \tilde{A}_1(i) \).
6. **Step 6:** If \( \tilde{B}(i) \) is a zero matrix, then the procedure terminates, otherwise go to Step 2.

The above algorithm is executed for each \( i \). Although in the above algorithm, we focus on the smallest value, we may focus on the biggest value.

We present a simple example.

**Example 6:** Suppose that \( \hat{B}(i) \) is given by

\[
\hat{B}(i) = \begin{bmatrix}
0.6 & 0.7 & 0.3 & 0.8 \\
0.4 & 0.3 & 0.7 & 0.2 
\end{bmatrix}.
\]

We also suppose that \( r(i) = 0 \).

First, in Step 3, we can obtain \( c_1(i) = 0.2 \). In Step 4, we can obtain

\[
\tilde{A}_1(i) = \begin{bmatrix}
0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 
\end{bmatrix}.
\]

In Step 5, \( \tilde{B}(i) \) is updated to

\[
\tilde{B}(i) := \begin{bmatrix}
0.6 & 0.7 & 0.1 & 0.8 \\
0.2 & 0.1 & 0.7 & 0 
\end{bmatrix}.
\]

Since \( \tilde{B}(i) \) is not a zero matrix, then go to Step 2.

Next, we can choose \( c_2(i) = 0.1 \), and we can obtain

\[
\tilde{A}_2(i) = \begin{bmatrix}
0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 
\end{bmatrix}.
\]

The matrix \( \tilde{B}(i) \) is updated to

\[
\tilde{B}(i) := \begin{bmatrix}
0.6 & 0.7 & 0 & 0.7 \\
0.1 & 0 & 0.7 & 0 
\end{bmatrix}.
\]

Third, we can choose \( c_3(i) = 0.1 \), and we can obtain

\[
\tilde{A}_3(i) = \begin{bmatrix}
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 
\end{bmatrix}.
\]

The matrix \( \tilde{B}(i) \) is updated to

\[
\tilde{B}(i) := \begin{bmatrix}
0.6 & 0.6 & 0 & 0.6 \\
0 & 0 & 0.6 & 0 
\end{bmatrix}.
\]

Finally, we can choose \( c_4(i) = 0.6 \), and we can obtain

\[
\tilde{A}_4(i) = \begin{bmatrix}
1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 
\end{bmatrix}.
\]

The matrix \( \tilde{B}(i) \) is updated to a zero matrix, and the proposed procedure terminates. Then, we can obtain \( q(i) = 4 \), and as seen from the above, we can obtain \( A_1(i) \).

Thus, we can design the PBN satisfying the constraint on steady-state probabilities by solving Problem 3 and applying the above procedure to \( \tilde{B}(i) \) obtained. Finally, we can easily show that the above procedure will terminates in at most \( m+1 \), where \( m \) is the number of columns in which all elements are non-zero.

**Remark 1:** In [7], [30], transition probability matrices with the size of \( 2^n \times 2^n \) are manipulated (\( n \) is the number of the state). In the proposed method, matrices with the size of \( 2 \times 2 |N_i| \) are manipulated, where \( |N_i| \) is determined depending on given network structure. When the network structure is not given, we set \( N_i = \{1, 2, \ldots, n\} \) (\( |N_i| = n \)). Hence, knowing the network structure enables us to use matrices with the smaller size.

**Remark 2:** It is important to minimize \( q(i) \). In the case where \( q(i) \) is given, Problem 2 is rewritten as an MILP problem (see Section IV-A). Hence, the minimum value of \( q(i) \) can be obtained by repeatedly solving an MILP problem. Developing an efficient algorithm is one of the future efforts.

**V. Numerical Examples**

**A. Simple Example**

As a simple example, consider designing a PBN with three states. We suppose that \( N_i, r(i), i = 1, 2, 3 \) are given by \( N_1 = \{3\}, N_2 = \{1, 3\}, N_3 = \{1, 2\} \), and \( r(i) = 0 \). We also suppose that \( s_i, i = 1, 2, 3 \) are given by \( s_1 = 0.8, s_2 = 0.4, \) and \( s_3 = 0.7 \), respectively.

Then, by solving Problem 3, we can obtain

\[
\hat{B}(1) = \begin{bmatrix}
0.34 & 0.14 \\
0.66 & 0.86 
\end{bmatrix},
\]

\[
\hat{B}(2) = \begin{bmatrix}
0.52 & 0.54 & 0.56 & 0.64 \\
0.48 & 0.46 & 0.44 & 0.36 
\end{bmatrix},
\]

\[
\hat{B}(3) = \begin{bmatrix}
0.43 & 0.45 & 0.23 & 0.32 \\
0.57 & 0.55 & 0.77 & 0.68 
\end{bmatrix}.
\]

From \( \hat{B}(1) \), we can obtain \( c_1(1) = 0.14, c_2(1) = 0.2, c_3(1) = 0.66, \) and

\[
\tilde{A}(1) = \begin{bmatrix}
1 & 1 \\
0 & 0 
\end{bmatrix},
\tilde{A}(2) = \begin{bmatrix}
1 & 0 \\
0 & 1 
\end{bmatrix},
\tilde{A}(3) = \begin{bmatrix}
0 & 0 \\
1 & 1 
\end{bmatrix}.
\]

From \( \hat{B}(2) \), we can obtain \( c_1(2) = 0.36, c_2(2) = 0.08, c_3(2) = 0.02, c_4(2) = 0.02, c_5(2) = 0.52, \) and

\[
\tilde{A}(1) = \begin{bmatrix}
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 
\end{bmatrix},
\tilde{A}(2) = \begin{bmatrix}
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 
\end{bmatrix},
\tilde{A}(3) = \begin{bmatrix}
0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 
\end{bmatrix},
\tilde{A}(4) = \begin{bmatrix}
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 
\end{bmatrix},
\tilde{A}(5) = \begin{bmatrix}
1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 
\end{bmatrix}.
\]

From \( \hat{B}(3) \), we can obtain \( c_1(3) = 0.23, c_2(3) = 0.09, c_3(3) = \).
0.11, \( c_4^{(3)} = 0.02 \), \( c_5^{(3)} = 0.55 \), and
\[
\begin{align*}
A_1^{(3)} &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & A_2^{(3)} &= \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \\
A_3^{(3)} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, & A_4^{(3)} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \\
A_5^{(3)} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.
\end{align*}
\]

From these matrices, we can obtain Boolean functions. For example, consider \( A_4^{(3)} \). From the second row of \( A_4 \), the disjunctive normal form of the Boolean function \( f_4^{(3)} \) is given by \((\neg x_1(k) \land \neg x_2(k)) \lor (x_1(k) \land \neg x_2(k)) \lor (x_1(k) \land x_2(k))\).

From this form, \( f_4^{(3)} = x_1(k) \lor \neg x_2(k) \) is obtained. Thus, we can obtain the PBN in which Boolean functions and probabilities are given by
\[
\begin{align*}
&f^{(1)} = \left\{ \begin{array}{ll}
f_1^{(1)} &= 0, & c_1^{(1)} = 0.14, \\
f_2^{(1)} &= x_3(k), & c_2^{(1)} = 0.2, \\
f_3^{(1)} &= 1, & c_3^{(1)} = 0.66, \\
f_4^{(1)} &= \neg x_1(k), & c_4^{(1)} = 0.08, \\
f_5^{(1)} &= \neg x_2(k), & c_5^{(1)} = 0.36,
\end{array} \right. \\
&f^{(2)} = \left\{ \begin{array}{ll}
f_1^{(2)} &= 1, & c_1^{(2)} = 0.36, \\
f_2^{(2)} &= \neg x_1(k) \land x_3(k), & c_2^{(2)} = 0.02, \\
f_3^{(2)} &= \neg (x_1(k) \land x_2(k)), & c_3^{(2)} = 0.52, \\
f_4^{(2)} &= 0, & c_4^{(2)} = 0.09, \\
f_5^{(2)} &= x_1(k) \land \neg x_2(k), & c_5^{(2)} = 0.03,
\end{array} \right. \\
&f^{(3)} = \left\{ \begin{array}{ll}
f_1^{(3)} &= 0, & c_1^{(3)} = 0.23, \\
f_2^{(3)} &= x_1(k) \land x_2(k), & c_2^{(3)} = 0.09, \\
f_3^{(3)} &= x_1(k), & c_3^{(3)} = 0.11, \\
f_4^{(3)} &= x_1(k) \land \neg x_2(k), & c_4^{(3)} = 0.02, \\
f_5^{(3)} &= 1, & c_5^{(3)} = 0.55.
\end{array} \right.
\end{align*}
\]

B. Biological Example

Next, as a biological example, consider the gene regulatory network with the gene WNT5A, which is related to melanoma.

A BN model is given by
\[
\begin{align*}
x_1(k+1) &= f^{(1)}(x_6(k)) = \neg x_6(k), \\
x_2(k+1) &= f^{(2)}(x_2(k), x_4(k), x_6(k)) \\
&= (\neg x_2(k) \land x_4(k) \land x_6(k)) \\
&\lor (x_2(k) \land (x_4(k) \lor x_6(k))), \\
x_3(k+1) &= f^{(3)}(x_7(k)) = \neg x_7(k), \\
x_4(k+1) &= f^{(4)}(x_4(k)) = x_4(k), \\
x_5(k+1) &= f^{(5)}(x_2(k), x_7(k)) = x_2(k) \land x_7(k), \\
x_6(k+1) &= f^{(6)}(x_3(k), x_4(k)) = x_3(k) \lor x_4(k), \\
x_7(k+1) &= f^{(7)}(x_2(k), x_7(k)) = \neg x_2(k) \land x_7(k),
\end{align*}
\]

where the concentration level (high or low) of the gene WNT5A is denoted by \( x_1 \), the concentration level of the gene pI305 by \( x_2 \), the concentration level of the gene S100P by \( x_3 \), the concentration level of the gene RET1 by \( x_4 \), the concentration level of the gene MRT1 by \( x_5 \), the concentration level of the gene HADH by \( x_6 \), and the concentration level of the gene STC2 by \( x_7 \). See [29] for further details.

Here, we consider controlling the steady-state by adding probability perturbation. Suppose that \( s_i \), \( i = 1, 2, \ldots, 7 \) are given by \( s_1 = s_2 = s_3 = 0.2 \) and \( s_4 = s_5 = s_6 = s_7 = 0.8 \), respectively. According to the above BN model, we have \( \mathcal{N}_1 = \{ 6 \}, \mathcal{N}_2 = \{ 2, 4, 6 \}, \mathcal{N}_3 = \{ 7 \}, \mathcal{N}_4 = \{ 4 \}, \mathcal{N}_5 = \{ 2, 7 \}, \mathcal{N}_6 = \{ 3, 4 \}, \mathcal{N}_7 = \{ 2, 7 \} \), and we suppose \( r(i) = 1 \). This setting is artificially given.

Next, we show the computation result. By solving Problem 3, we can obtain
\[
\begin{align*}
x_1(k+1) &= f^{(1)}(x_6(k)) = \neg x_6(k), \\
x_2(k+1) &= \left\{ \begin{array}{ll}
f_2^{(1)}(x_2(k), x_4(k), x_6(k)) & c_1^{(1)} = 0.28, \\
0 & c_2^{(1)} = 0.72, \\
x_3(k+1) &= f^{(3)}(x_7(k)) = \neg x_7(k), \\
x_4(k+1) &= f^{(4)}(x_4(k)) = x_4(k), \\
x_5(k+1) &= f^{(5)}(x_2(k), x_7(k)) & c_1^{(5)} = 0.31, \\
1 & c_2^{(5)} = 0.69, \\
x_6(k+1) &= f^{(6)}(x_3(k), x_4(k)) & c_1^{(6)} = 0.95, \\
0 & c_2^{(6)} = 0.05, \\
x_7(k+1) &= f^{(7)}(x_2(k), x_7(k)) & c_1^{(7)} = 0.83, \\
0 & c_2^{(7)} = 0.17.
\end{array} \right.
\end{align*}
\]

From this PBN, we see that \( x_2(k), x_6(k) \), and \( x_7(k) \) must be forcibly inactive with the probability 0.72, 0.05, and 0.17, respectively, and \( x_5(k) \) must be forcibly active with the probability 0.69.

VI. CONCLUSION

In this paper, we studied the problem of finding a probabilistic Boolean network (PBN) with given network structure and desired steady-state probabilities. Based on the matrix-based representation, a solution method for the problem of finding a PBN was proposed. Two numerical examples were also presented.

Applying the proposed method to several biological networks is one of the future efforts.

This research was partly supported by Grant-in-Aid for Scientific Research (C) 26420412.

APPENDIX A

LINEAR INEQUALITY FOR THE PRODUCT OF BINARY AND CONTINUOUS VARIABLES

In this appendix, we introduce the following lemma on a linear inequality expression for the product of binary and continuous variables. See [3] for further details.

Lemma 1: Consider \( x \in \mathcal{X} \subseteq \mathcal{R}^n \), \( \delta \in \{ 0, 1 \} \), and \( g : \mathcal{R}^n \to \mathcal{R}^m \). Then, \( z = \delta g(x) \) is equivalent to two linear inequalities \( g_0 \leq z \leq g_0 \delta \) and \( g(x) - g(1 - \delta) \leq z \leq g(x) - g(1 - \delta) \), where \( g = \min_{x \in \mathcal{X}} g(x) \) and \( \delta = \max_{x \in \mathcal{X}} g(x) \).

REFERENCES


