



HOKKAIDO UNIVERSITY

Title	Design of reduced-order and pinning controllers for probabilistic Boolean networks using reinforcement learning
Author(s)	Takizawa, Michiaki; Kobayashi, Koichi; Yamashita, Yuh
Citation	Applied mathematics and computation, 457, 128211 https://doi.org/10.1016/j.amc.2023.128211
Issue Date	2023-11-15
Doc URL	https://hdl.handle.net/2115/95649
Rights	© <2023>. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/
Rights(URL)	https://creativecommons.org/licenses/by-nc-nd/4.0/
Type	journal article
File Information	amc_takizawa_husca.pdf



Design of Reduced-order and Pinning Controllers for Probabilistic Boolean Networks Using Reinforcement Learning^{*}

Michiaki Takizawa^a, Koichi Kobayashi^{a,*}, Yuh Yamashita^a

^a*Graduate School of Information Science and Technology, Hokkaido University, Kita 14,
Nishi 9, Kita-ku, Sapporo, 060-0814, Japan*

Abstract

In this paper, we study a stabilization method for probabilistic Boolean networks (PBNs) using Q-learning, which is one of the typical methods in reinforcement learning. A PBN is a class of discrete-time stochastic logical systems in which update functions are randomly chosen from the set of the candidates of Boolean functions. In the existing methods using reinforcement learning, a design method of structured controllers has not been studied. In this paper, we propose reward design methods to derive reduced-order controllers and pinning controllers. The key idea is to adjust the structure of controllers by the reward for the structure of the Q-table. The advantage of the proposed method is that implementation is easy, because the proposed method can be embedded in the existing Q-learning-based stabilization algorithm. In design of pinning controllers, we can calculate not only controllers but also pinning nodes in which the control input is assigned. We demonstrate through numerical examples that the controller obtained from our proposed method is simpler than that from the existing method.

Keywords: probabilistic Boolean networks, reinforcement learning, reduced-order controllers, pinning controllers, gene regulatory networks, systems biology

^{*}This work was partly supported by JSPS KAKENHI Grant Numbers JP21H04558, JP22K04163, JP23H01430.

^{*}Corresponding author

Email address: `k-kobaya@ssi.ist.hokudai.ac.jp` (Koichi Kobayashi)

1. Introduction

In the research field of systems biology, one of the central problems is to develop control theory of gene regulatory networks. Therapeutic interventions, which are realized by radiation, chemotherapy, and so on, may be regarded as control of gene regulatory networks [1, 2, 3]. Hence, development of control methods is important to realize gene therapy technologies [4] in the future.

As the first step in developing control theory of gene regulatory networks, it is significant to consider an appropriate mathematical model to represent behaviors of gene expression. Several mathematical models have been developed such as Bayesian networks, piecewise affine systems (hybrid systems), and Petri nets [5, 6]. In the last decade, analysis/control methods using a Boolean network (BN) model have been widely studied. A BN is a class of discrete-time logical systems [7, 8]. Each state takes on a binary value (0 (OFF), 1 (ON)), and its dynamics are modeled by Boolean functions. Moreover, a probabilistic BN (PBN) is an extended model of BNs [9]. In a PBN, the dynamics for each state are represented by a Boolean function which is randomly selected from their candidates. When we consider the noise included in the behavior of gene regulatory networks, it is suitable to use a PBN as a mathematical model.

Control theory that BNs and PBNs are used as a mathematical model has been extensively developed. To solve analysis/control problems, several techniques such as BDD (binary decision diagram) [10, 11], SAT (satisfiability) solvers [12, 13], and the semi-tensor product (STP) method [14, 15] are used. In particular, the STP method is widely used. The fundamental problems have been studied such as controllability of BNs (the STP-based methods [14, 16, 17], the polynomial-time algorithm [18], the use of Perron-Frobenius theory [19], computational complexity analysis [20]), controllability of PBNs [21, 22, 23], observability of BNs (the STP-based methods [14, 24, 25], the graph theoretic method [26]), stabilization of BNs [27, 28, 29], stabilization of PBNs [30, 31, 32], optimal control of BNs (the STP-based method [33, 34], the optimization-based method [35]), and optimal control of PBNs (the STP-based method [36, 37, 38], the optimization-based method [39]).

In these existing methods, suppose that either a BN or a PBN is given in advance. Control/analysis problems are solved using Boolean functions and selection probabilities. However, there is a case where full knowledge of the dynamics of gene regulatory networks is unavailable. Hence, in control

of gene regulatory networks, it is important to develop model-free methods.

In this paper, using reinforcement learning, we study a model-free design method of two types of simple controllers such that the closed-loop system is stable. In control of complex networks such as gene regulatory networks, a simpler structure of controllers is desirable. It will be expected that implementation of control systems becomes easier in the future by using controllers obtained by the proposed method. For the stabilization algorithm proposed in [40], design methods for reward functions are proposed. First, a reward design method for deriving a reduced-order controller is proposed. Here, we use the tentative controller derived on the way of the Q-learning based algorithm. The reward is decided based on the dimension of the tentative controller. Next, a reward design method for deriving a pinning controller is proposed. In pinning control, the external control input is added to some states (pinning nodes) (see [41] for general pinning control, and see [42, 43, 44] for pinning control of BNs/PBNs). In pinning control, it is desirable that the number of pinning nodes is as small as possible. Such objective can be achieved by the reward design. By the proposed design method of reward functions, both pinning nodes and controllers can be derived simultaneously.

The main contributions of this paper are listed as follows.

- We propose the reward design method for deriving a reduced-order controller.
- We propose the reward design method for deriving both a pinning-order controller and pinning nodes.
- We demonstrate through numerical examples that the controller obtained from our proposed method is simpler than that from the existing method.

This paper is organized as follows. In Section 2, applications of reinforcement learning to control of PBNs are briefly reviewed. In Section 3, the outline of PBNs, the stabilization problem, and the Q-learning based stabilization algorithm are summarized. In Section 4, a design procedure of the reward function for deriving a reduced-order controller is proposed. In Section 5, we consider the case of pinning controllers. In Section 6, two numerical examples are presented to demonstrate two methods. In Section 7, we give a conclusion of this paper.

Notation: Let $\{0, 1\}^n$ denote the set of n -dimensional vectors whose elements take either 0 or 1.

2. Related Work

In this section, we briefly review control methods of PBNs using reinforcement learning.

Reinforcement learning techniques such as Q-learning [45, 46], deep Q-network (DQN) [47], and Double DQN (DDQN) [48] are powerful tools as model-free methods. In the typical problem formulation, the overall system is composed of an environment and an agent. The environment is stated in the form of a Markov decision process (MDP). The agent takes an action to the environment. The environment returns the reward to the agent. The agent updates an action rule using the reward, and takes an action again. By repeating this procedure, an action rule is learned. We remark that the transition probability distribution in a given MDP is not used in model-free methods.

A PBN can be equivalently rewritten as a discrete-time MDP. Hence, we can use reinforcement learning techniques to solve the control problem of PBNs. Some control methods of PBNs using reinforcement learning have been studied. In [40, 49], the Q-learning based stabilization algorithm has been proposed. In [50], the DDQN based stabilization algorithm has been proposed. In [51, 52, 53], algorithms for self-triggered control that both the control input and the sampling interval are calculated have been proposed. In [51, 52], Q-learning has been used. In [53], DDQN has been used. In [54], the output tracking problem has been studied using DDQN.

In control of gene regulatory networks, it is also important to consider the structure of controllers. For example, since it is difficult to measure all states, design of reduced-order controllers is important. To the best of our knowledge, design methods of structured controller using reinforcement learning has not been studied.

3. Preliminaries

3.1. Probabilistic Boolean Networks

In this section, we give an overview of PBNs (see, e.g., [9] for further details).

A PBN with n states and m control inputs is described by

$$x_{t+1}^{(i)} = f^{(i)}(x_t, u_t), \quad i \in \mathcal{N} := \{1, 2, \dots, n\}, \quad (1)$$

where $x(t) = [x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(n)}]^\top \in \{0, 1\}^n$ is the state at time t , $u_t = [u_t^{(1)}, u_t^{(2)}, \dots, u_t^{(m)}]^\top \in \{0, 1\}^m$ is the control input at time t , and $t = \{0, 1, 2, \dots\}$ is the discrete time. The Boolean function $f^{(i)}$ is randomly selected from the set of candidates $\{f_1^{(i)}, f_2^{(i)}, \dots, f_{l(i)}^{(i)}\}$ ($f_j^{(i)} : \{0, 1\}^{n+m} \rightarrow \{0, 1\}$). The probability that $f_j^{(i)}$ is selected is given by $p_j^{(i)}$, where $\sum_{j=1}^{l(i)} p_j^{(i)} = 1$.

For given initial state $x_0 \in \{0, 1\}^n$ and control sequence $\mathbf{u}_t := \{u_0, u_1, \dots, u_{t-1}\}$, let $x(t; x_0, \mathbf{u}_t)$ denote the solution to PBN (1). The state $x_0 \in \{0, 1\}^n$ is called an equilibrium point if there exists a control input $u_0 \in \{0, 1\}^m$ such that $\Pr\{x(1; x_0, u_0) = x_0\} = 1$.

We present an example to explain the behavior of PBNs.

Example 3.1. Consider the following PBN with $n = 2$ and $m = 1$:

$$\begin{aligned} x_{t+1}^{(1)} &= \begin{cases} x_t^{(1)} \vee (x_t^{(2)} \wedge u_t), & p_1^{(1)} = 0.7, \\ x_t^{(1)} \wedge u_t, & p_2^{(1)} = 0.3, \end{cases} \\ x_{t+1}^{(2)} &= x_t^{(2)} \vee u_t, \quad p_1^{(2)} = 1.0. \end{aligned}$$

Fig. 1 shows the state transition diagram of this PBN, where each node represents the values of $x^{(1)}$ and $x^{(2)}$. For example, consider the case of $x_0 = [1, 0]^\top$, $\mathbf{u}_2 = \{1, 0\}$. Then, from Fig. 1, we can obtain

$$x(2; x_0, \mathbf{u}_2) = \begin{cases} [0, 1]^\top, & \text{Prob.} = 0.51, \\ [1, 1]^\top, & \text{Prob.} = 0.49. \end{cases}$$

In addition, from Fig. 1, we see that there are three equilibrium points (i.e., $[0, 0]^\top$, $[0, 1]^\top$, and $[1, 1]^\top$).

3.2. Stabilization Problem

First, we define the stabilizability of the PBN (1).

Definition 3.1. The PBN (1) is said to be asymptotically stabilizable at $\bar{x}_d \in \{0, 1\}^n$ in distribution, if for every $x_0 \in \{0, 1\}^n$ there exists $\mathbf{u}_t$ such that

$$\lim_{t \rightarrow \infty} \Pr\{x(t; x_0, \mathbf{u}_t) = \bar{x}_d\} = 1,$$

where $\bar{x}_d$ is a given equilibrium point and is called a target equilibrium point.

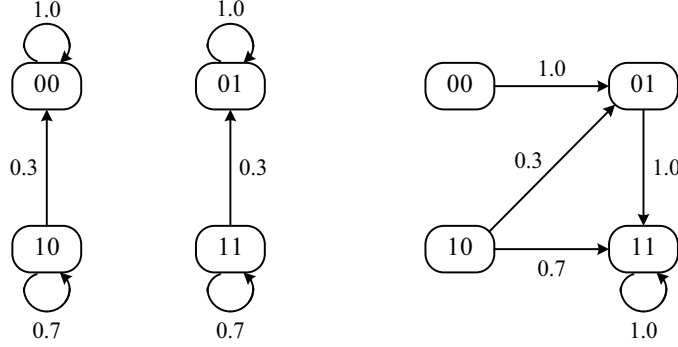


Figure 1: State transition diagram of the PBN in Example 3.1. (Left) Case of $u(t) = 0$, (Right) Case of $u(t) = 1$.

The l -step controllable set is defined by

$$\Gamma_l(\bar{x}_d) := \{x_0 \in \{0, 1\}^n : \exists \mathbf{u}(l) \text{ such that } \Pr\{x(t; x_0, \mathbf{u}_l) = \bar{x}_d\} > 0\}.$$

In [40], the following theorem has been derived.

Theorem 3.1. *A PBN (1) is asymptotically stabilizable at $\bar{x}_d$ in distribution if and only if there exists a non-negative integer τ such that $\Gamma_\tau(\bar{x}_d) = \{0, 1\}^n$ holds.*

3.3. Q-Learning Based Stabilization Algorithm

In this section, we briefly summarize the stabilization algorithm using Q-learning, which has been proposed in [40]. Q-learning is one of the reinforcement learning methods in the field of machine learning. In Q-learning, the environment is stated in the form of an MDP. The agent receives the reward from the environment, and determines the action to the environment.

Since the PBN (1) can be rewritten as an MDP, the stabilization problem can be solved by using Q-learning. Consider the following problem.

Problem 1. *For any initial state $x_0 \in \{0, 1\}^n$, find an optimal state feedback law $u_t = \mu^*(x_t, \bar{x}_d)$ maximizing the following objective function:*

$$J = E \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}(x_t, u_t, x_{t+1}, \bar{x}_d) \right]$$

subject to the PBN (1) and the condition that there exists a non-negative integer τ satisfying $\Gamma_\tau(\bar{x}_d) = \{0, 1\}^n$, where $r_{t+1}(x_t, u_t, x_{t+1}, \bar{x}_d)$ is a given reward function.

For example, the reward function $r_{t+1}(x_t, u_t, x_{t+1}, \bar{x}_d)$ is defined as follows: $r_{t+1} = 1$ if $x_{t+1} = \bar{x}_d$, otherwise $r_{t+1} = 0$ [40]. An algorithm for solving Problem 1 using Q-learning has been proposed in [40]. Its summary is given as follows.

Summary of Procedure for deriving a stabilizing controller using Q-learning:

Step 1: Give $\bar{x}_d$, N , T , τ , and ω .

Step 2: Set the initial value 0 for each element of the initial Q-table $Q_0(x_t, u_t, \bar{x}_d)$. Set $\Gamma_\tau(\bar{x}_d)$ as the empty set.

Step 3: Repeat from Step 3-1 to Step 3-3 for $ep = 0, 1, \dots, N - 1$.

Step 3-1: Set x_{ep} randomly. Set α_{ep} , t , and x_t as $1/(ep + 1)^\omega$, 0, and x_{ep} , respectively.

Step 3-2: Repeat from Step 3-2-1 to Step 3-2-5 until $t = T$ or $x_t = x_d$ hold.

Step 3-2-1: Choose u_t using the ϵ -greedy policy.

Step 3-2-2: Apply u_t to the environment. Receive x_{t+1} and r_{t+1} from the environment.

Step 3-2-3: Obtain the Q-table at $t + 1$ using x_t , x_{t+1} , and r_{t+1} .

Step 3-2-4: Update $\Gamma_\tau(\bar{x}_d)$ as $\Gamma_\tau(\bar{x}_d) \cup x_{ep}$, if $(x_{t+1} = \bar{x}_d) \wedge (t < \tau)$ holds.

Step 3-2-5: Update t as $t + 1$. Return to Step 3-2-1.

Step 3-3: Update Q_0 as Q_t .

Step 3-4: If $\Gamma_\tau(\bar{x}_d) = \{0, 1\}^n$ holds, then obtain the optimal Q-table Q^* as Q_t , and derive the optimal state feedback law as

$$u_t = \mu^*(x(t), \bar{x}_d) = \arg \max_u Q^*(x(t), u, \bar{x}_d),$$

otherwise the PBN (1) is not stabilizable. The procedure is terminated.

This algorithm is based on the conventional Q-learning algorithm, and can be implemented in conventional software tools such as Python and MATLAB. See [40] for further details. In a similar way to the conventional Q-learning algorithm, this algorithm is model-free, because the transition probability distribution associated with the MDP is not used.

Remark 3.1. *To efficiently work the above algorithm, parameters such as N and T should be appropriately set. When parameters are not given appropriately, there is a possibility that it is determined that a PBN is not stabilizable even if it is stabilizable.*

4. Design of Reward Functions for Reduced-order Controllers

In reinforcement learning, it is necessary to appropriately design the reward according to the objective that we want to achieve. Our purpose in this part is to obtain a stabilizing feedback controller in which the number of states is as small as possible. To achieve this purpose, the tentative controller obtained during learning is used. The reward is determined according to the number of states used in the tentative controller. Stability of the closed-loop system is also considered, and its priority is higher than the reduction of the controller.

Let n_T denote the number of states used in the tentative controller. The dimension n_T can be obtained based on Boolean functions calculated from $\mu^*(x_t, \bar{x}_d) = \arg \max_u Q^*(x_t, u, \bar{x}_d), \forall x_t$. Then, we present a design procedure of the reward function. In the derivation procedure of Section 3.3, Step 3-2-2 is replaced with the following procedure (Step 3-2-2').

Proposed Procedure of Step 3-2-2':

Step 3-2-2'-1: Apply u_t to the environment. Receive x_{t+1} from the environment.

Step 3-2-2'-2: If $x_{t+1} = \bar{x}_d$ then $r_{t+1} = \alpha$, otherwise $r_{t+1} = 0$, where $\alpha > 0$ is a given constant.

Step 3-2-2'-3: Derive a tentative control from $\mu^*(x_t, \bar{x}_d) = \arg \max_u Q^*(x_t, u, \bar{x}_d), \forall x_t$, where $u_t^{(i)}$ is set as $u_t^{(i)} = 0$, if $u_t^{(i)}$ is not uniquely determined from x_t and $\bar{x}_d$.

Step 3-2-2'-4: Derive n_T from Boolean functions of the tentative controller.

Step 3-2-2'-5: Based on n_T , update the reward r_{t+1} as $r_{t+1} \leftarrow r_t + \beta(n - n_T)$, where $\beta > 0$ is a given constant (weight).

In Step 3-2-2'-1, the agent applies the control input to the environment. After that, the agent receives the state from the environment. In Step 3-2-2'-2, the reward is tentatively determined based on the current state. In Step 3-2-2'-3, the agent derives a tentative controller based on the Q-table. In the beginning of learning, many elements of the Q-table are zero. As a result, a tentative controller is not uniquely determined. If $u_t^{(i)}$ is not uniquely determined from x_t and $\bar{x}_d$, then $u_t^{(i)}$ is set as $u_t^{(i)} = 0$. In Step 3-2-2'-4, the agent derives the dimension n_T of the obtained tentative controller. In Step 3-2-2'-5, the agent updates the reward based on n_T . If $n_T = n$, then the reward is not given in this step. This case corresponds that the controller utilizes all states, and is not desirable. For a smaller n_T , a larger reward is given. As a result, the Q-table such that n_T becomes smaller can be constructed. Thus, by this procedure, we can realize both stabilization of a given PBN and simplification of a controller. The Q-learning-based stabilization algorithm including the proposed procedure can also be implemented in conventional software tools such as Python and MATLAB. In addition, even if Step 3-2-2 is replaced with Step 3-2-2', the Q-learning based stabilization algorithm is also model-free.

5. Design of Reward Functions for Pinning Controllers

In this subsection, we propose a design method of reward functions for pinning control of PBNs. In pinning control, unlike with (1), external control inputs are added for a PBN with no control inputs. A state that is assigned the input is called a pinning node. It is desirable that the number of pinning nodes is as small as possible. In the conventional pinning control of PBNs, without loss of generality, the following PBN with pinning nodes is considered:

$$\begin{cases} x_{t+1}^{(i)} = f^{(i)}(x_t, u_t^{(j(i))}), & i \in \mathcal{P}, \\ x_{t+1}^{(i)} = f^{(i)}(x_t), & i \in \mathcal{N} \setminus \mathcal{P}, \end{cases} \quad (2)$$

where $i \in \mathcal{P} \subseteq \mathcal{N}$ is a pinning node, and $j(i) \in \{1, 2, \dots, m\}$ is the index set of external control inputs, which depends on i . We consider the problem of

deriving both the set $\mathcal{P}$ with a smaller number of elements and the controller for each external input such that the closed-loop system is asymptotically stable. In this paper, using reinforcement learning, we consider solving this problem. For simplicity of discussion, instead of (2), we consider the following PBN:

$$x_{t+1}^{(i)} = f^{(i)}(x_t) \oplus u_t^{(i)}, \quad i \in \mathcal{N}, \quad (3)$$

where $\oplus$ is the exclusive OR operation, and $n = m$ holds in the initial step. By designing reward functions appropriately, we consider deriving a controller that more elements of u_t are always zero. If $u_t^{(i)} = 0$ always holds, then i is not a pinning node. By excluding such i , we can obtain a smaller m . Hence, we can derive both a pinning controller and pinning nodes simultaneously.

In the derivation procedure of Section 3.3, Step 3-2-2 is replaced with the following procedure (Step 3-2-2").

Proposed Procedure of Step 3-2-2":

Step 3-2-2"-1: Apply u_t to the environment. Receive x_{t+1} from the environment.

Step 3-2-2"-2: If $x_{t+1} = \bar{x}_d$ then $r_{t+1} = \alpha$, otherwise $r_{t+1} = 0$, where $\alpha > 0$ is a given constant.

Step 3-2-2"-3: Calculate Boolean functions of the tentative controller from $\mu^*(x_t, \bar{x}_d) = \arg \max_u Q^*(x_t, u, \bar{x}_d), \forall x(t)$, where $u_t^{(i)}$ is set as $u_t^{(i)} = 0$, if $u_t^{(i)}$ is not uniquely determined from x_t and $\bar{x}_d$. Let m_0 denote the number of elements of u_t in which the Boolean function is identically given by zero.

Step 3-2-2"-4: Based on m_0 , update the reward r_{t+1} as $r_{t+1} \leftarrow r_{t+1} + \beta m_0$, where $\beta > 0$ is a given constant (weight).

In a similar way to Section 4, Step 3-2-2"-2 is related to stabilization to the target equilibrium point. In Step 3-2-2"-3, $n - m_0$ implies the tentative number of pinning nodes. In Step 3-2-2"-4, the additional reward is determined based on m_0 . In the case where m_0 is larger, the reward becomes larger. As a result, the Q-table is updated such that m_0 is larger. Hence, a pinning controller that the number of pinning nodes is smaller can be obtained. The proposed procedure can also be implemented in conventional

software tools such as Python and MATLAB. In a similar way to Section 4, the Q-learning based stabilization algorithm including the above procedure is also model-free.

Remark 5.1. *For large-scale PBNs, there is a possibility that an appropriate controller cannot be derived by only design of reward functions. This technical issue may be overcome by giving the candidates of pinning nodes in advance, where the number of candidates is much less than n . The candidates of pinning nodes are given based on biological knowledge. If there is no biological knowledge, we may give the candidates randomly.*

6. Numerical Examples

In this section, we present two numerical examples.

6.1. Numerical Example 1: Stabilization by Reduced-order Controllers

First, we present a numerical example on design of reduced-order controllers.

Consider the following PBN model of the apoptosis network given in [31]:

$$\begin{aligned} x_{t+1}^{(1)} &= \begin{cases} \neg x_t^{(2)} \wedge u_t, & p_1^{(1)} = 0.6, \\ u_t, & p_2^{(1)} = 0.4, \end{cases} \\ x_{t+1}^{(2)} &= \begin{cases} \neg x_t^{(1)} \wedge x_t^{(3)}, & p_1^{(2)} = 0.7, \\ x_t^{(2)}, & p_2^{(2)} = 0.3, \end{cases} \\ x_{t+1}^{(3)} &= \begin{cases} x_t^{(2)} \vee u_t, & p_1^{(3)} = 0.8, \\ x_t^{(3)}, & p_2^{(3)} = 0.2, \end{cases} \end{aligned}$$

where x_1 denotes the concentration levels of apoptosis protein inhibitors (IAP), x_2 active caspase 3 (C3a), and x_3 active caspase 8 (C8a), and u denotes tumor necrosis factor α (TNF).

For this PBN, we consider solving the stabilization problem to the target equilibrium point $\bar{x}_d = [0, 1, 1]^\top$. Here we use the Q-learning based algorithm in Section 3.3 and the reward design proposed in Section 4. The parameters in the Q-learning based algorithm are given as follows: $N = 8 \times 10^4$, $T = 120$, $\tau = 2^3 - 1$, $\gamma = 0.9$, $\omega = 0.85$. The parameters in the reward design are given as follows: $\alpha = 4$ and $\beta = 1$.

Table 1: Obtained Q-table.

	$u = 0$	$u = 1$
$x = [0, 0, 0]^\top$	9.28	10.28
$x = [0, 0, 1]^\top$	23.95	22.15
$x = [0, 1, 0]^\top$	26.76	22.14
$x = [0, 1, 1]^\top$	50.00	40.94
$x = [1, 0, 0]^\top$	9.28	10.29
$x = [1, 0, 1]^\top$	11.69	10.57
$x = [1, 1, 0]^\top$	24.30	18.15
$x = [1, 1, 1]^\top$	30.15	24.26

We present the computation result. Table 1 shows the obtain Q-table. From this Q-table, we can obtain the truth table shown in Table 2. From Table 2, the controller can be derived as

$$u_t = \neg x_t^{(2)} \wedge \neg x_t^{(3)}. \quad (4)$$

Fig 2 shows the average state trajectory of 1000 samples, where each element of the initial state is randomly chosen (0 (1) is chosen with the probability 0.5 (0.5)). At each time, the Boolean function is randomly chosen according to prescribed probabilities. From this figure, we see that the average state is oscillatory at first some time steps, but converges to the target equilibrium point $\bar{x}_d = [0, 1, 1]^\top$.

The controller obtained in [40], where a similar simulation is performed without the additional reward about the reduced dimensionality, is given by

$$\begin{aligned} u_t = & (x_t^{(1)} \wedge x_t^{(2)} \wedge \neg x_t^{(3)}) \\ & \vee (\neg x_t^{(1)} \wedge \neg x_t^{(2)} \wedge \neg x_t^{(3)}) \\ & \vee (\neg x_t^{(1)} \wedge x_t^{(2)} \wedge \neg x_t^{(3)}). \end{aligned} \quad (5)$$

From two controllers (4) and (5), we see that the proposed reward design works efficiently.

6.2. Numerical Example 2: Stabilization by Pinning Controllers

Next, we present a numerical example on design of pinning controllers.

Table 2: Truth table obtained from Q-table.

	u
$x = [0, 0, 0]^\top$	1
$x = [0, 0, 1]^\top$	0
$x = [0, 1, 0]^\top$	0
$x = [0, 1, 1]^\top$	0
$x = [1, 0, 0]^\top$	1
$x = [1, 0, 1]^\top$	0
$x = [1, 1, 0]^\top$	0
$x = [1, 1, 1]^\top$	0

Consider the following PBN:

$$\begin{aligned}
x_{t+1}^{(1)} &= (x_t^{(1)} \vee x_t^{(2)}) \oplus u_t^{(1)}, \\
x_{t+1}^{(2)} &= \begin{cases} \neg x_t^{(1)} \oplus u_t^{(2)}, & p_1^{(2)} = 0.1, \\ \neg x_t^{(3)} \oplus u_t^{(2)}, & p_2^{(2)} = 0.9, \end{cases} \\
x_{t+1}^{(3)} &= (x_t^{(2)} \vee x_t^{(3)}) \oplus u_t^{(3)}.
\end{aligned}$$

This PBN is obtained by adding the control inputs to the PBN given in [42], according to (3).

For this PBN, we consider solving the stabilization problem to the target equilibrium point $\bar{x}_d = [1, 1, 1]^\top$. Here we use the Q-learning based algorithm in Section 3.3 and the reward design proposed in Section 5. The parameters in the Q-learning based algorithm are the same as in previous subsection. The parameters in the reward design are also the same as in previous subsection. Then, the controller can be derived as

$$\begin{aligned}
u_t^{(1)} &= 0, \\
u_t^{(2)} &= x_t^{(1)} \vee (\neg x_t^{(1)} \wedge \neg x_t^{(2)} \wedge x_t^{(3)}) \vee (\neg x_t^{(1)} \wedge x_t^{(2)}), \\
u_t^{(3)} &= 0.
\end{aligned}$$

From the obtained controller, we see that the number of pinning nodes is only one, and the pinning node is given by 2.

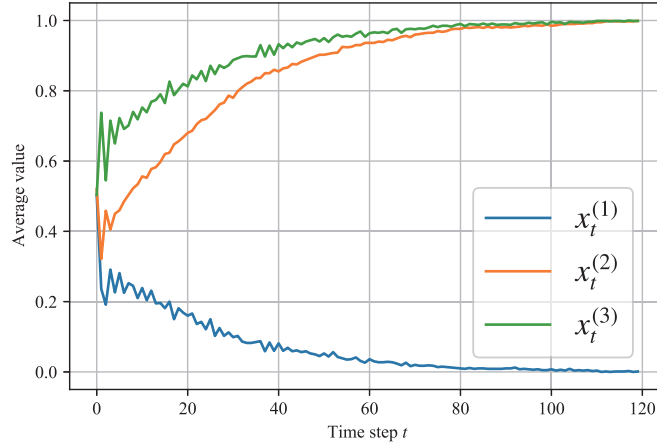


Figure 2: Average trajectory of the state (1000 samples).

Fig 3 shows the state transition diagram of the closed-loop systems in which the obtained controller is applied to a given PBN. Each node represents the values of $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$. From this figure, we see that the state converges to the target equilibrium point $\bar{x}_d = [1, 1, 1]^\top$. Thus, using the proposed reward design, pinning nodes can be specified, and their controller can also be derived.

Finally, we consider finding a controller by the existing method [40]. Then, we can obtain the following controller:

$$\begin{aligned} u_t^{(1)} &= \neg x_t^{(1)} \wedge \neg x_t^{(2)} \wedge x_t^{(3)}, \\ u_t^{(2)} &= x_t^{(1)} \wedge x_t^{(3)}, \\ u_t^{(3)} &= 0. \end{aligned}$$

From this controller, we see that the number of pinning nodes is two, which is larger than in the proposed method. Thus, we conclude that the proposed method efficiently works.

7. Conclusion

In this paper, we studied a design method of reward functions for the Q-learning based stabilization algorithm of PBNs. First, the reward design

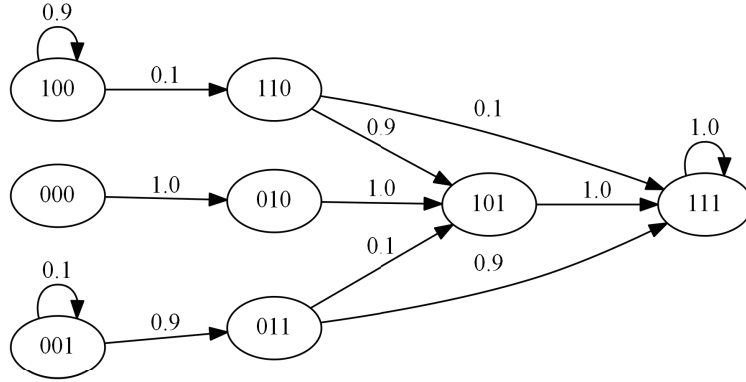


Figure 3: State transition diagram of the closed-loop system.

for reduced-order controllers was proposed. Next, the reward design for pinning controllers was proposed. Finally, numerical examples were presented to demonstrate the proposed method. We clarified that not only control specifications such as stabilization but also the structure of controllers can be considered by control methods using reinforcement learning.

In control of complex network systems such as gene regulatory networks, it is desirable that the number of times that the control input is updated is smaller. Self-triggered control is well known as one of the control methods achieving this control purpose [51, 55]. Based on the proposed approach, it is important to develop a reinforcement learning-based control method considering both the structure of controllers and self-triggered control. Moreover, to apply the proposed reward design method to large-scale PBNs, improving it is also future work.

The authors would like to thank Mr. Fuma Motoyama, Hokkaido University for fruitful discussions and valuable comments.

References

- [1] J. J. O’Shea, D. M. Schwartz, A. V. Villarino, M. Gadina, I. B. McInnes, A. Laurence, The JAK-STAT pathway: impact on human disease and therapeutic intervention, *Annual Review of Medicine* 66 (2015) 311–328.
- [2] I. Shmulevich, E. R. Dougherty, W. Zhang, From Boolean to probabilistic Boolean networks as models of genetic regulatory networks, *Proceedings of the IEEE* 90 (11) (2002) 1778–1792.

- [3] I. Shmulevich, E. R. Dougherty, W. Zhang, Gene perturbation and intervention in probabilistic Boolean networks, *Bioinformatics* 18 (10) (2002) 1319–1331.
- [4] H. Santos-Rosa, C. Caldas, Chromatin modifier enzymes, the histone code and cancer, *European Journal of Cancer* 41 (16) (2005) 2381–2402.
- [5] H. De Jong, Modeling and simulation of genetic regulatory systems: A literature review, *Journal of Computational Biology* 9 (1) (2002) 67–103.
- [6] S.-i. Azuma, E. Yanagisawa, J.-i. Imura, Controllability analysis of biosystems based on piecewise-affine systems approach, *IEEE Transactions on Automatic Control* 53 (Special Issue) (2008) 139–152.
- [7] S. A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, *Journal of Theoretical Biology* 22 (3) (1969) 437–467.
- [8] S. A. Kauffman, et al., *The origins of order: Self-organization and selection in evolution*, Oxford University Press, USA, 1993.
- [9] I. Shmulevich, E. R. Dougherty, S. Kim, W. Zhang, Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks, *Bioinformatics* 18 (2) (2002) 261–274.
- [10] Q. Yuan, H. Qu, J. Pang, A. Mizera, Improving BDD-based attractor detection for synchronous Boolean networks, *Science China Information Sciences* 59 (8) (2016) 1–16.
- [11] T. Van Giang, T. Akutsu, K. Hiraishi, An FVS-based approach to attractor detection in asynchronous random Boolean networks, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 19 (2) (2020) 806–818.
- [12] E. Dubrova, M. Teslenko, A SAT-based algorithm for finding attractors in synchronous Boolean networks, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8 (5) (2011) 1393–1399.
- [13] T. Tamura, T. Akutsu, Detecting a singleton attractor in a Boolean network utilizing SAT algorithms, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 92 (2) (2009) 493–501.

- [14] D. Cheng, H. Qi, Controllability and observability of Boolean control networks, *Automatica* 45 (7) (2009) 1659–1667.
- [15] D. Cheng, H. Qi, Z. Li, Analysis and control of Boolean networks: a semi-tensor product approach, London, U.K.: Springer, 2011.
- [16] D. Cheng, C. Li, X. Zhang, F. He, Controllability of Boolean networks via mixed controls, *IEEE Control Systems Letters* 2 (2) (2018) 254–259.
- [17] Q. Zhu, Y. Liu, J. Lu, J. Cao, Further results on the controllability of Boolean control networks, *IEEE Transactions on Automatic Control* 64 (1) (2018) 440–442.
- [18] K. Kobayashi, J.-I. Imura, K. Hiraishi, Polynomial-time algorithm for controllability test of a class of Boolean biological networks, *EURASIP Journal on Bioinformatics and Systems Biology* 2010 (1) (2010) 210685.
- [19] D. Laschov, M. Margaliot, Controllability of Boolean control networks via the Perron–Frobenius theory, *Automatica* 48 (6) (2012) 1218–1223.
- [20] E. Weiss, M. Margaliot, G. Even, Minimal controllability of conjunctive Boolean networks is NP-complete, *Automatica* 92 (2018) 56–62.
- [21] F. Li, J. Sun, Controllability of probabilistic Boolean control networks, *Automatica* 47 (12) (2011) 2765–2771.
- [22] Y. Liu, H. Chen, J. Lu, B. Wu, Controllability of probabilistic Boolean control networks based on transition probability matrices, *Automatica* 52 (2015) 340–345.
- [23] M. Toyoda, Y. Wu, On optimal time-varying feedback controllability for probabilistic Boolean control networks, *IEEE Transactions on Neural Networks and Learning Systems* 31 (6) (2019) 2202–2208.
- [24] E. Fornasini, M. E. Valcher, Observability, reconstructibility and state observers of Boolean control networks, *IEEE Transactions on Automatic Control* 58 (6) (2013) 1390–1401.
- [25] Y. Yu, M. Meng, J.-e. Feng, G. Chen, Observability criteria for Boolean networks, *IEEE Transactions on Automatic Control* 67 (11) (2021) 6248–6254.

- [26] D. Laschov, M. Margaliot, G. Even, Observability of Boolean networks: A graph-theoretic approach, *Automatica* 49 (8) (2013) 2351–2362.
- [27] Y. Guo, P. Wang, W. Gui, C. Yang, Set stability and set stabilization of Boolean control networks based on invariant subsets, *Automatica* 61 (2015) 106–112.
- [28] R. Liu, J. Lu, Y. Liu, J. Cao, Z.-G. Wu, Delayed feedback control for stabilization of Boolean control networks with state delay, *IEEE Transactions on Neural Networks and Learning Systems* 29 (7) (2017) 3283–3288.
- [29] A. Yerudkar, C. Del Vecchio, L. Glielmo, Feedback stabilization control design for switched Boolean control networks, *Automatica* 116 (2020) 108934.
- [30] M. Meng, J. Lam, J.-E. Feng, K. C. Cheung, Stability and stabilization of Boolean networks with stochastic delays, *IEEE Transactions on Automatic Control* 64 (2) (2018) 790–796.
- [31] R. Zhou, Y. Guo, Y. Wu, W. Gui, Asymptotical feedback set stabilization of probabilistic Boolean control networks, *IEEE Transactions on Neural Networks and Learning Systems* 31 (11) (2019) 4524–4537.
- [32] H. Li, X. Yang, S. Wang, Robustness for stability and stabilization of Boolean networks with stochastic function perturbations, *IEEE Transactions on Automatic Control* 66 (3) (2020) 1231–1237.
- [33] E. Fornasini, M. E. Valcher, Optimal control of Boolean control networks, *IEEE Transactions on Automatic Control* 59 (5) (2014) 1258–1270.
- [34] Y. Wu, X.-M. Sun, X. Zhao, T. Shen, Optimal control of Boolean control networks with average cost: A policy iteration approach, *Automatica* 100 (2019) 378–387.
- [35] K. Kobayashi, K. Hiraishi, Optimal control of gene regulatory networks with effectiveness of multiple drugs: A Boolean network approach, *BioMed research international* 2013 (2013).

- [36] Y. Wu, T. Shen, An algebraic expression of finite horizon optimal control algorithm for stochastic logical dynamical systems, *Systems & Control Letters* 82 (2015) 108–114.
- [37] Y. Wu, T. Shen, Policy iteration algorithm for optimal control of stochastic logical dynamical systems, *IEEE Transactions on Neural Networks and Learning Systems* 29 (5) (2017) 2031–2036.
- [38] Y. Wu, Y. Guo, M. Toyoda, Policy iteration approach to the infinite horizon average optimal control of probabilistic Boolean networks, *IEEE Transactions on Neural Networks and Learning Systems* (2020).
- [39] K. Kobayashi, K. Hiraishi, An integer programming approach to optimal control problems in context-sensitive probabilistic Boolean networks, *Automatica* 47 (6) (2011) 1260–1264.
- [40] A. Acernese, A. Yerudkar, L. Glielmo, C. Del Vecchio, Reinforcement learning approach to feedback stabilization problem of probabilistic Boolean control networks, *IEEE Control Systems Letters* 5 (1) (2020) 337–342.
- [41] X. Wang, H. Su, Pinning control of complex networked systems: A decade after and beyond, *Annual Reviews in Control* 38 (1) (2014) 103–111.
- [42] F. Li, L. Xie, Set stabilization of probabilistic Boolean networks using pinning control, *IEEE Transactions on Neural Networks and Learning Systems* 30 (8) (2019) 2555–2561.
- [43] L. Lin, J. Cao, J. Lu, J. Zhong, S. Zhu, Stabilizing large-scale probabilistic Boolean networks by pinning control, *IEEE Transactions on Cybernetics* (2021).
- [44] J. Zhong, Q. Pan, B. Li, J. Lu, Minimal pinning control for oscillatory of Boolean networks, *IEEE Transactions on Neural Networks and Learning Systems* (2021).
- [45] C. J. Watkins, P. Dayan, Q-learning, *Machine learning* 8 (1992) 279–292.
- [46] T. Jaakkola, M. Jordan, S. Singh, Convergence of stochastic iterative dynamic programming algorithms, *Advances in Neural Information Processing Systems* 6 (1993).

- [47] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *nature* 518 (7540) (2015) 529–533.
- [48] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, in: *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30, 2016.
- [49] P. Bajaria, A. Yerudkar, C. Del Vecchio, Random forest Q-learning for feedback stabilization of probabilistic Boolean control networks, in: *2021 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 2021, pp. 1539–1544.
- [50] S. Moschogiannis, E. Chatzaroulas, V. Šliogeris, Y. Wu, Deep reinforcement learning for stabilization of large-scale probabilistic Boolean networks, *IEEE Transactions on Control of Network Systems* (2022).
- [51] A. Acernese, A. Yerudkar, L. Glielmo, C. Del Vecchio, Model-free self-triggered control co-design for probabilistic Boolean control networks, *IEEE Control Systems Letters* 5 (5) (2020) 1639–1644.
- [52] P. Bajaria, A. Yerudkar, L. Glielmo, C. Del Vecchio, Y. Wu, Self-triggered control of probabilistic Boolean control networks: A reinforcement learning approach, *Journal of the Franklin Institute* 359 (12) (2022) 6173–6195.
- [53] A. Yerudkar, E. Chatzaroulas, C. Del Vecchio, S. Moschogiannis, Sampled-data control of probabilistic Boolean control networks: A deep reinforcement learning approach, *Information Sciences* 619 (2023) 374–389.
- [54] A. Acernese, A. Yerudkar, L. Glielmo, C. Del Vecchio, Double deep-Q learning-based output tracking of probabilistic Boolean control networks, *IEEE Access* 8 (2020) 199254–199265.
- [55] W. Heemels, K. H. Johansson, P. Tabuada, An introduction to event-triggered and self-triggered control, in: *51st IEEE Conference on Decision and Control*, IEEE, 2012, pp. 3270–3285.