frontiers | Frontiers in Computational Neuroscience

TYPE Original Research
PUBLISHED 02 May 2024
DOI 10.3389/fncom.2024.1385047

Check for updates

OPEN ACCESS

EDITED BY
Yang Cui,
University of Science and Technology Liaoning, China

REVIEWED BY
Geyang Xiao,
Zhejiang Lab, China
Junqi Yang,
Henan Polytechnic University, China

*CORRESPONDENCE
Hui Tian
✉ tianhui@cqupt.edu.cn

RECEIVED 11 February 2024
ACCEPTED 08 April 2024
PUBLISHED 02 May 2024

CITATION
Tian H, Su X and Hou Y (2024) Feedback stabilization of probabilistic finite state machines based on deep Q-network. *Front. Comput. Neurosci.* 18:1385047. doi: 10.3389/fncom.2024.1385047

COPYRIGHT
© 2024 Tian, Su and Hou. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

# Feedback stabilization of probabilistic finite state machines based on deep Q-network

Hui Tian[1]*, Xin Su[1] and Yanfang Hou[2]

[1]Key Laboratory of Industrial Internet of Things and Networked Control, Ministry of Education, Chongqing University of Posts and Telecommunications, Chongqing, China, [2]School of Electrical and Electronic Engineering, Chongqing University of Technology, Chongqing, China

**Background:** As an important mathematical model, the finite state machine (FSM) has been used in many fields, such as manufacturing system, health care, and so on. This paper analyzes the current development status of FSMs. It is pointed out that the traditional methods are often inconvenient for analysis and design, or encounter high computational complexity problems when studying FSMs.

**Method:** The deep Q-network (DQN) technique, which is a model-free optimization method, is introduced to solve the stabilization problem of probabilistic finite state machines (PFSMs). In order to better understand the technique, some preliminaries, including Markov decision process, $\epsilon$-greedy strategy, DQN, and so on, are recalled.

**Results:** First, a necessary and sufficient stabilizability condition for PFSMs is derived. Next, the feedback stabilization problem of PFSMs is transformed into an optimization problem. Finally, by using the stabilizability condition and deep Q-network, an algorithm for solving the optimization problem (equivalently, computing a state feedback stabilizer) is provided.

**Discussion:** Compared with the traditional Q learning, DQN avoids the limited capacity problem. So our method can deal with high-dimensional complex systems efficiently. The effectiveness of our method is further demonstrated through an illustrative example.

KEYWORDS

**probabilistic finite state machine (PFSM), deep Q-network (DQN), feedback stabilization, artificial neural network (ANN), controller**

## 1 Introduction

The finite state machine (FSM), also known as finite automata (Yan et al., 2015b), is an important mathematical model, which has been used in many different fields, such as manufacturing system (Wang et al., 2017; Piccinini et al., 2018), health care (Shah et al., 2017; Zhang, 2018; Fadhil et al., 2019), and so on. The deterministic finite state machine (DFSM) is known for its deterministic behaviors, in which each subsequent state is uniquely determined by its input event and preceding state (Vayadande et al., 2022). However, DFSMs may not be effective in dealing with random behaviors (Ratsaby, 2019), for example, the randomness caused by component failures in sequential circuits (El-Maleh and Al-Qahtani, 2014). To address the challenge, a probabilistic finite state machine (PFSM) was proposed in the study by Vidal et al. (2005), which provides a more flexible framework for those systems that exhibit random behaviors. Especially, it gives an effective solution to practical issues, such as the reliability assessment of sequential circuits (Li and Tan, 2019). Therefore, the PFSM offers a new perspective for the theoretical research of FSMs.

On the other hand, the stabilization of systems is an important and fundamental research topic, and there have been many excellent research results in various fields, for example, Boolean control network (Tian et al., 2017; Tian and Hou, 2019), time-delay systems (Tian and Wang, 2020), neural networks (Ding et al., 2019), and so on.

The stabilization research of FSMs is no exception and has also attracted the attention of many scholars. The concepts of stability and stabilization of discrete event systems described by FSMs were given in the study by Özveren et al. (1991). A polynomial solution of stability detection and a method for constructing stabilizers were presented. Passino et al. (1994) utilzed the Lyapunov method to study the stability and stabilization of FSMs. Tarraf et al. (2008) proposed some new concepts, including gain stability, incremental stability and external stability, and then established a research framework for robust stability of FSMs. Kobayashi et al. developed a linear state equation representation method for modeling DFSMs in the study by Kobayashi (2006) and Kobayashi and Imura (2007) and derived a necessary and sufficient condition for DFSM to be stabilizable at a target equilibrium node in the study by Kobayashi et al. (2011).

However, as we know, the FSM is most often non-linear. Moreover, none of the above methods are convenient when analyzing and designing various FSMs. In the last decade, scholars applied the semi-tensor product (STP) of matrices to FSMs and derived many excellant results. First, with the help of STP, an algebraic form of DFSMs was given in the study by Xu et al. (2013). This algebraic form is a discrete-time bilinear equation. Then, the classic control theory can be used to investigate FSMs. Especially, under the algebraic form, necessary and sufficient conditions for the stabilizability of DFSMs were derived in the study by Xu et al. (2013), and a state feedback controller was obtained by computing a corresponding matrix inequality. Moreover, Yan et al. (2015a) provided a necessary and sufficient condition to check whether a set of states can be stabilized. Han and Chen (2018) considered the set stabilization of DFSMs and provided an optimal design approach for stabilizing controllers. Later, Zhang et al. used the STP method to investigate PFSMs and non-deterministic FSMs. Specifically, a necessary and sufficient condition for stabilization with probability one and a design method for optimal state feedback controller were provided in the study by Zhang et al. (2020a). Moreover, a systematic procedure was designed to get a static output feedback stabilizer for non-deterministic FSMs in the study by Zhang et al. (2020b). Although the STP method is very useful in analyzing discrete event systems, including various FSMs, it suffers from high computational complexity and can only handle small-scale or even micro-scale discrete event systems. To solve the problem, this study refers to techniques developed by Acernese et al. (2020) to solve the stabilization problem of high-dimensional PFSMs, and then provides a reinforcement learning algorithm to compute a state feedback stabilizer for PFSMs. The algorithm is especially advantageous in dealing with high-dimensional systems.

The rest of this study is arranged as follows: Section 2 introduces some preliminary knowledge, including PFSM, Markov decision process (MDP), deep Q newtwork (DQN), and $\epsilon$-greedy strategy. In Section 3, a stabilizabillity condition is derived and an algorithm based on DQN is provided. An illustrative example is employed to show the effectiveness of our results, as shown in Section 4, which is followed by a brief conclusion in Section 5.

# 2 Methods

For the convenience of statement, some symbol explanations are provided first.

**Notation:** $\mathbb{R}$ expresses the set of all real numbers. $\mathbb{Z}^+$ stands for the set of all positive integers. $\mathbb{Z}^+_{a,b}$ denotes the set $\{a, a+1, \cdots, b\}$, where $a, b \in \mathbb{Z}^+$, $a \leq b$. $|A|$ is the cardinality of set $A$.

## 2.1 Probabilistic finite state machine

A PFSM is a five-tuple

$$\Lambda = (\mathcal{X}, \mathcal{U}, P, f, \mathcal{X}_0), \tag{1}$$

where the set $\mathcal{X} := \{\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_n\}$ represents a finite set of states, and $\mathcal{X}_0 \in \mathcal{X}$ is the initial state. $\mathcal{U} := \{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m\}$ denotes a finite set of events. $P : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, 1]$ is a transition probability function, and $P(\mathcal{X}_i, \mathcal{U}_k, \mathcal{X}_j) := P^{\mathcal{U}_k}_{\mathcal{X}_i, \mathcal{X}_j}$ expresses the probability of PFSM (1), transiting from state $\mathcal{X}_i \in \mathcal{X}$ to state $\mathcal{X}_j \in \mathcal{X}$ under the input event $\mathcal{U}_k \in \mathcal{U}$, satisfying

$$\sum_{\mathcal{X}_j \in \mathcal{X}} P^{\mathcal{U}_k}_{\mathcal{X}_i, \mathcal{X}_j} = 1$$

or

$$\sum_{\mathcal{X}_j \in \mathcal{X}} P^{\mathcal{U}_k}_{\mathcal{X}_i, \mathcal{X}_j} = 0.$$

The state transition function $f : \mathcal{X} \times \mathcal{U} \to 2^{\mathcal{X}}$ describes that PFSM (1) may reach different states from one state under the same input event, where $2^{\mathcal{X}}$ is the power set of $\mathcal{X}$.

## 2.2 Markov decision process and optimization methods

A Markov decision process (MDP) is characterized by a quintuple

$$\Omega = (S, A, P, R, \gamma), \tag{2}$$

where $S$ is a set of states, $A$ is a set of actions, $P$ is a state transition probability function, $R$ is a reward function, and $\gamma \in [0, 1]$ is a discount factor that determines the trade-off between short-term and long-term gains.

MDP (2) may reach state $s_{t+1}$ from state $s_t \in S$ under the chosen action $a_t \in A$, and its probability is determined by the function $P^{a_t}_{s_t, s_{t+1}} = P(s_{t+1} \mid s_t, a_t)$. The expected one-step reward from state $s_t$ to state $s_{t+1}$ via action $a_t$ is as follows:

$$R^{a_t}_{s_t, s_{t+1}} = \mathbb{E}\left[r_{t+1} \mid s_t, a_t\right]$$

where $r_{t+1} = r_{t+1}(s_t, a_t, s_{t+1})$ represents the immediate return after adopting action $a_t$ at time $t$, and $\mathbb{E}[\cdot]$ is the expected value of $[\cdot]$.

The objective of MDP (2) is to determine an optimal policy $\pi$. This policy can maximize the expected return $\mathbb{E}_\pi[G_t]$ under policy $\pi$ where

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

For a given policy $\pi$, the value function of a state $s_t$, denoted by $v_\pi(s_t)$, is the expected return of MDP (2) taking an action according to the policy $\pi$ at time step $t$:

$$v_\pi(s_t) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t\right], \forall s_t \in S. \tag{3}$$

The optimal policy is as follows:

$$\pi^*(s_t, a_t) = \arg\max_{\pi \in \Pi} v_\pi(s_t), \forall s_t \in S \tag{4}$$

where $\Pi$ is the set of all admissible policies.

From (4), it is easy to understand $v^*(s_t) = v_{\pi^*}(s_t)$. Since $v_\pi(\cdot)$ satisfies the Bellman equation, we have

$$v^*(s_t) = \max_{a \in A} \sum_{s \in S} P^a_{s_t,s} \left[ R^a_{s_t,s} + \gamma v^*(s) \right] \tag{5}$$

Similarly, the action-value function describes the cumulative return from state-action $(s_t, a_t)$ under policy $\pi$

$$q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s = s_t, a = a_t \right], \forall a_t \in A. \tag{6}$$

By substituting (3) into (6), we can obtain

$$q_\pi(s_t, a_t) = \mathbb{E}_\pi [r_{t+1} + \gamma v_\pi(s_{t+1})],$$

which represents the expected return of action $a_t$ adopted by MDP (2) at state $s_t$, following policy $\pi$. The action-value function under optimal strategy $\pi^*$ is called as the optimal action-value function,i.e., $q^*(s_t, u_t) := q_{\pi^*}(s_t, a_t), \forall s_t \in S, \forall a_t \in A$. Since $v^*(s_t) = \max_a q^*(s_t, a)$, from (5), we can get

$$q^*(s_t, a_t) = \sum_{s \in S} P^{a_t}_{s_t,s} \left[ R^{a_t}_{s_t,s} + \gamma \max_a q^*(s, a) \right].$$

Therefore, if MDP (2) exists an optimal deterministic policy, it can be expressed as follows:

$$\mu^*(s_t) = \arg\max_{a \in A} q^*(s_t, a), \forall s_t \in S.$$

DQN is such a technique that combines Q leaning with arificial neural networks (ANNs), providing an effective approach to decision-making problems in dynamic and uncertain environments. It uses ANNs to construct parametric models and estimate action value functions online. Compared with Q learning, the main advantages of DQN are as follows: (1) DQN uses ANNs to approximate Q functions, overcoming the issue of limited capacity in Q tables and enabling the algorithm to handle high-dimensional state spaces. (2) DQN makes full use of empirical knowledge.

Q learning updates the value function according to the following temporal difference (TD) formula:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \max_{a'} q\left(s_{t+1}, a'\right) - q(s_t, a_t) \right], \tag{7}$$

where $r_{t+1} + \gamma \max_{a'} q\left(s_{t+1}, a'\right)$ is the TD target, $r_{t+1} + \gamma \max_{a'} q\left(s_{t+1}, a'\right) - q(s_t, a_t)$ is the TD error $\delta$, and $0 < \alpha \leq 1$ is a constant that determines how quickly the past experiences are forgotten.

When dealing with high-dimensional complex systems, the action-value function $q(s, a)$, as described in Equation (7), is approximated by an ANN to reduce computational complexity. This can be achieved by minimizing the following loss function

$$\mathcal{L}(\theta_t) = (r_{t+1} + \gamma \max_{a'} q\left(s_{t+1}, a'; \theta_t^-\right) - q(s_t, a_t; \theta_t))^2, \tag{8}$$

where the parameter $\theta_t^-$ is a periodic copy of the current network parameter $\theta_t$.

By differentiating Equation (8), we have

$$\begin{aligned} \nabla_{\theta_t} \mathcal{L}(\theta_t) = 2(r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a'; \theta_t^-) \\ - q(s_t, a_t; \theta_t))(- \nabla_{\theta_t} q(s_t, a_t; \theta_t)), \end{aligned} \tag{9}$$

where $\nabla_{\theta_t} q(s_t, a_t; \theta_t)$ represents the gradient of $q(s_t, a_t; \theta_t)$ with respect to the parameter $\theta_t$.

We choose the gradient descent method as the optimization strategy

$$\theta_{t+1} = \theta_t - \frac{\alpha}{2} \nabla_{\theta_t} \mathcal{L}(\theta_t). \tag{10}$$

By substituting Equations (9) into (10), we obtain an update formula for parameter $\theta_t$

$$\begin{aligned} \theta_{t+1} = \theta_t + \alpha [r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a'; \theta_t^-) \\ - q(s_t, a_t; \theta_t)] \nabla_{\theta_t} q(s_t, a_t; \theta_t). \end{aligned}$$

Finally, the $\epsilon$-greedy strategy is used for action selection. Specifically, an action is chosen randomly with probability $\epsilon \in \mathbb{R}(0 < \epsilon \leq 1)$, and the best estimated action is chosen with probability $1 - \epsilon$. As learning progresses, $\epsilon$ gradually decreases, and the policy is shifted from exploring the action space to exploiting the learned Q values. The policy $\pi(a \mid s)$ is as follows:

$$\pi(a \mid s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{if } a = \arg\max_{a \in A} q(s, a) \\ \frac{\epsilon}{|A|} & \text{other actions}, \end{cases}$$

where $\pi(a \mid s)$ is the probability of MDP (2) selecting action $a$ at state $s$. $\arg\max_{a \in A} q(s, a)$ stands for the action with the highest estimated Q value for state $s$.

# 3 Results

We first give a definition.

**Definition 1**: Assume that $\mathcal{X}_e$ is an equilibrium state of PFSM (1). The PFSM is said to be feedback stabilizable to $\mathcal{X}_e$ with probability one, if for any initial state $\mathcal{X}_i \in \mathcal{X}$, there exists a control sequence $\mathbf{U} := \mathcal{U}_{l_1}, \mathcal{U}_{l_2}, \cdots, \mathcal{U}_{l_k} \in \mathcal{U}$, such that $P^{\mathbf{U}}_{\mathcal{X}_i, \mathcal{X}_e} = 1$.

We define an attraction domain $\Re_k(\mathcal{X}_e)$ for an equilibrium state $\mathcal{X}_e$, which is a set of states that can reach $\mathcal{X}_e$ in $k$ steps.

$$\begin{aligned} \Re_k(\mathcal{X}_e) = \{ \mathcal{X}_i \in \mathcal{X} \mid \text{there exists a control sequence } \mathbf{U}: \\ = \mathcal{U}_{l_1}, \mathcal{U}_{l_2}, \cdots, \mathcal{U}_{l_k} \in \mathcal{U}, \text{ such that } P^{\mathbf{U}}_{\mathcal{X}_i, \mathcal{X}_e} = 1 \}. \end{aligned} \tag{11}$$

Next, we give an important result.

**Theorem 1:** Assume that $\mathcal{X}_e$ is an equilibrium state of PFSM (1). The PFSM is feedback stabilizable to $\mathcal{X}_e$ with probability one, if and only if there exists an integer $\rho \leq n - 1$ such that

$$\Re_\rho(\mathcal{X}_e) = \mathcal{X}. \tag{12}$$

**Proof** (Necessity): Assume that PFSM (1) is feedback stabilizable to the equilibrium state $\mathcal{X}_e$ with probability one. Then, according to

Check whether $\Re_\rho(\mathcal{X}_e)=\mathcal{X}$ holds or not. If yes, perform the following calculations. Otherwise, PFSM (1) cannot be stabilized to $\mathcal{X}_e$.
**Input:** $B$, $\mathcal{X}_e$, $\gamma$, $\epsilon$, $\tau$, $T$, mini-batch size $M$, $\theta$
**Output:** $\mu^*(\mathcal{X}_i,\mathcal{X}_e)$
Initialize weights $\theta \leftarrow rand([0,1])$, $\theta^- \leftarrow \theta$, the replay memory $B \leftarrow \emptyset$
Define reward function to reach state $\mathcal{X}_e$
**for** $episode = 1,2,\cdots,N$ **do**
  $t \leftarrow 0$, $\mathcal{X}_t \leftarrow rand(\mathcal{X})$
  **while** $t < T$ and $\mathcal{X}_t \neq \mathcal{X}_e$ **do**
    Choose $\mathcal{U}_t$ by using $\epsilon$-greedy policy
    Applying $\mathcal{U}_t$, read $\mathcal{X}_{t+1}$ and $r_{t+1}$
    Store transition $(\mathcal{X}_t,\mathcal{U}_t,r_{t+1},\mathcal{X}_{t+1},done)$ in $B$
    **if** $|B| \geq M$ **then**
      sample mini-batch $M$ from $B$
      **for** every state $\mathcal{X}_i \in \mathcal{X}$ in $M$ **do**
        **if** $\mathcal{X}_{t+1} == done$ **then**
          target[i] $= r_i$
        **else**
          target[i] $= r_i + \gamma \max_{\mathcal{U}' \in \mathcal{U}} q(\mathcal{X}_{t+1},\mathcal{U}';\theta^-)$
        **end if**
      **end for**
      Perform gradient descent and calculate the loss
      Train and update network weight $\theta$
    **end if**
    $t \leftarrow t+1$
  **end while**
  **if** $episode\%\tau == 0$ **then**
    update target network weight: $\theta^- \leftarrow \theta$
  **end if**
**end for**
**return** $\mu^*(\mathcal{X}_i,\mathcal{X}_e) \leftarrow \arg\max_{\mathbf{U}}(\mathcal{X}_i,\mathbf{U},\mathcal{X}_e;\theta^-), \forall \mathcal{X}_i \in \mathcal{X}$

**Algorithm 1.** State feedback stabilization of PFSM (1) based on deep Q-network.

Definition 1, for any initial state $\mathcal{X}_i$, there exists a control sequence $\mathbf{U}:=\mathcal{U}_{l_1},\mathcal{U}_{l_2},\cdots,\mathcal{U}_{l_\rho}$, such that $P^{\mathbf{U}}_{\mathcal{X}_i,\mathcal{X}_e}=1$, namely $\mathcal{X}_i \in \Re_k(\mathcal{X}_e)$. Due to the fact that the state space is a finite set, there must be an integer $\rho$, such that $\Re_\rho(\mathcal{X}_e)=\mathcal{X}$ holds.

(Sufficiency): Assume that Equation (12) holds. For any initial state $\mathcal{X}_i \in \mathcal{X}$, we have $\mathcal{X}_i \in \Re_\rho(\mathcal{X}_e)$. From Equation (11), there exists a positive integer $\rho$ and a control sequence $\mathbf{U}:=\mathcal{U}_{l_1},\mathcal{U}_{l_2},\cdots,\mathcal{U}_{l_\rho}$ such that $\mathcal{X}_i$ can be driven to $\mathcal{X}_e$ by $\mathbf{U}$ in $\rho$ steps with probability one. According to Definition 1, PFSM (1) is feedback stabilizable to $\mathcal{X}_e$ with probability one. ■

We cast the feedback control problem of PFSM (1) into a model-free reinforcement learning framework. The main aim is to find a state feedback controller, which can guarantee the finite time stabilization of PFSM (1). This means that all states can be controlled and brought to an equilibrium state within finite steps. Therefore, PFSM (1) is rewritten as $(\mathcal{X},\mathcal{U},P,R,\gamma)$, where $P$ is

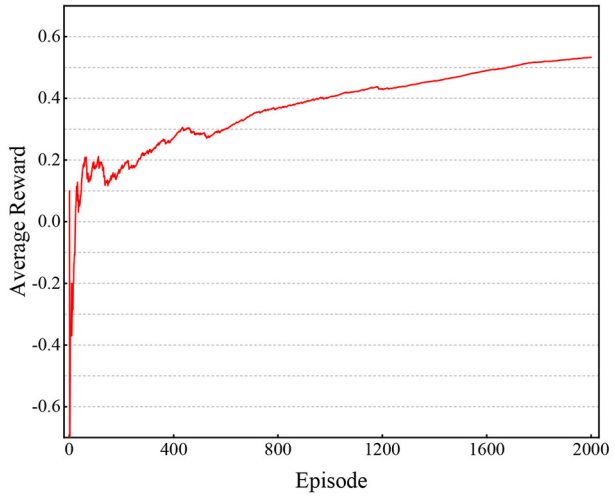


FIGURE 2
Performance of Algorithm 1 in Example 1.

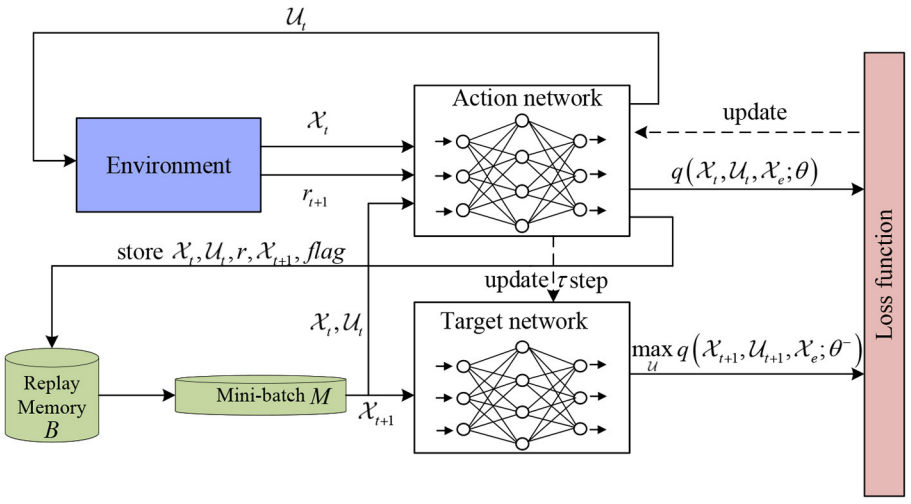


FIGURE 1
Structure diagram of DQN.

10.3389/fncom.2024.1385047

TABLE 1 A state feedback controller of PFSM (14).

| State | $\mathcal{X}_1$ | $\mathcal{X}_2$ | $\mathcal{X}_3$ | $\mathcal{X}_4$ | $\mathcal{X}_5$ | $\mathcal{X}_6$ | $\mathcal{X}_7$ | $\mathcal{X}_8$ | $\mathcal{X}_9$ | $\mathcal{X}_{10}$ | $\mathcal{X}_{11}$ | $\mathcal{X}_{12}$ | $\mathcal{X}_{13}$ | $\mathcal{X}_{14}$ | $\mathcal{X}_{15}$ | $\mathcal{X}_{16}$ | $\mathcal{X}_{17}$ | $\mathcal{X}_{18}$ | $\mathcal{X}_{19}$ | $\mathcal{X}_{20}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Action | $\mathcal{U}_1$ | $\mathcal{U}_2$ | $\mathcal{U}_1$ | $\mathcal{U}_1$ | $\mathcal{U}_2$ | $\mathcal{U}_1$ | $\mathcal{U}_2$ | $\mathcal{U}_2$ | $\mathcal{U}_1$ | $\mathcal{U}_3$ | $\mathcal{U}_2$ | $\mathcal{U}_3$ | $\mathcal{U}_3$ | $\mathcal{U}_3$ | $\mathcal{U}_2$ | $\mathcal{U}_2$ | $\mathcal{U}_2$ | $\mathcal{U}_3$ | $\mathcal{U}_2$ | $\mathcal{U}_2$ |

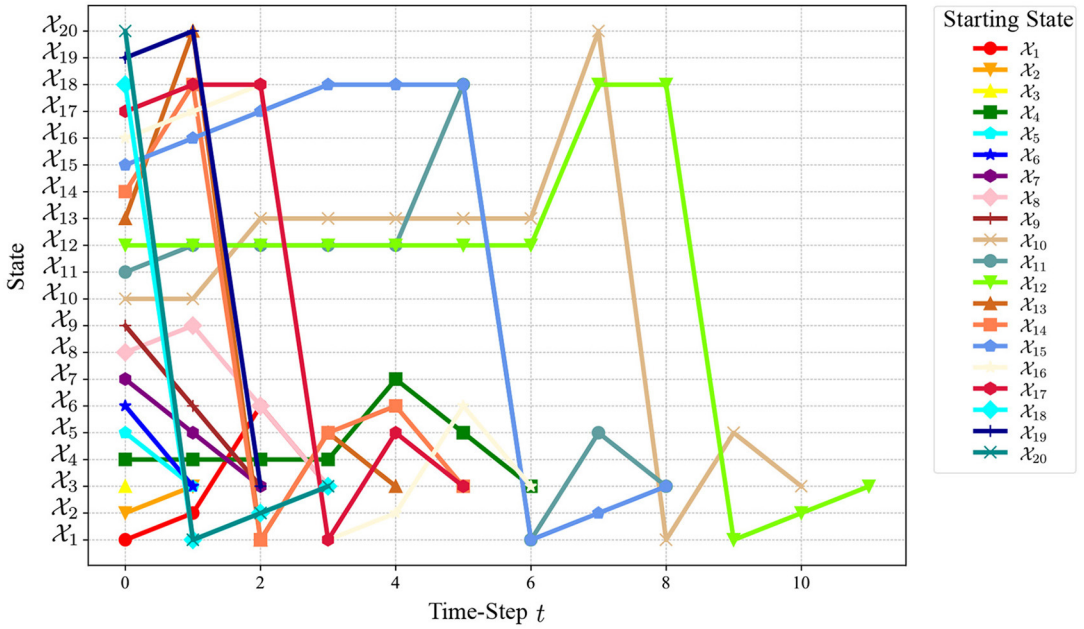


FIGURE 3
Evolution of the closed-loop system (16).

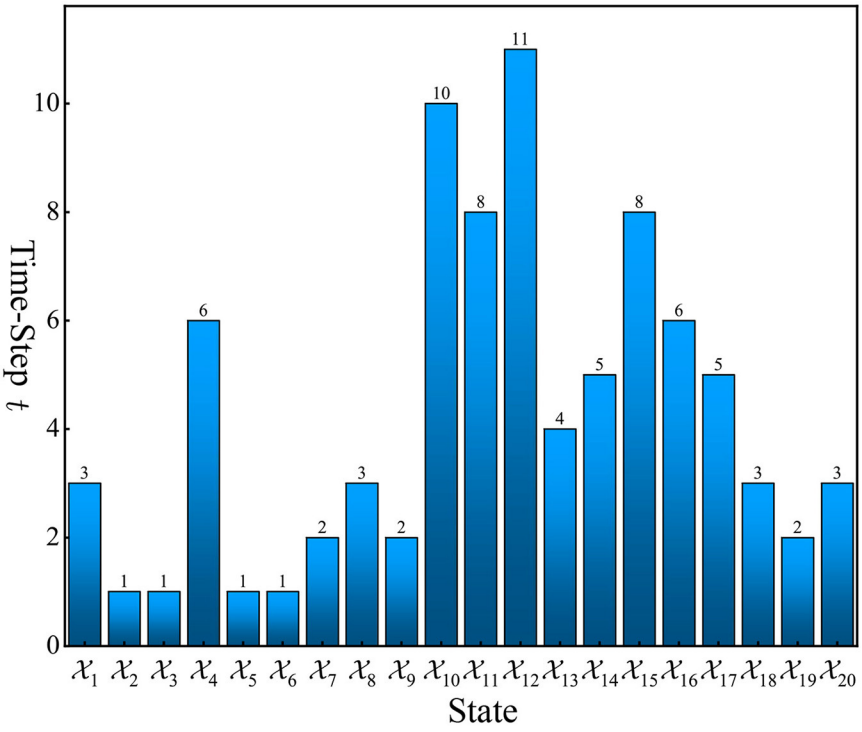


FIGURE 4
The number of steps required to stabilize PFSM (14) to $\mathcal{X}_3$.

unknown. The stabilization problem of PFSM (1) is formulated as follows:

$$\max_{\mu(\cdot)} \mathbb{E}_{\mu}\left[\sum_{t=0}^{\infty}\gamma^{t} r_{t+1}\left(\mathcal{X}_t,\mathcal{U}_t,\mathcal{X}_{t+1}\right)\right], \forall \mathcal{X}_0 \in \mathcal{X} \quad \text{subject to (1)}, \tag{13}$$

where

$$r_{t+1}=\begin{cases}1, \text{if } \mathcal{X}_{t+1}=\mathcal{X}_e\\ -0.1, \text{otherwise.}\end{cases}$$

The objective of Equation (13) is to find an action $\mathbf{U}$ that maximizes the action-value function $q^*$ among all possible actions in $\mathcal{U}$. Therefore, for any state $\mathcal{X}_t$ and external condition $\mathcal{X}_e$, the optimal state feedback control law of PFSM (1) is as follows:

$$\mu^*(\mathcal{X}_t,\mathcal{X}_e)=\arg\max_{\mathbf{U}\in\mathcal{U}} q^*(\mathcal{X}_t,\mathbf{U},\mathcal{X}_e;\theta^-), \forall \mathcal{X}_t\in\mathcal{X}.$$

Based on the above discussion, we are ready to introduce an algorithm to design an optimal feedback controller (see Algorithm 1). It should be noted that in this algorithm, DQN uses two ANNs. The structure diagram of DQN is shown in Figure 1.

**Remark 1:** This algorithm is mainly used to solve the stabilization problem of high dimensional PFSMs. For small or micro-scale PFSMs, it is slightly more complex. In this case, we can choose the STP method. Therefore, Algorithm 1 and the STP method complement each other.

According to the results calculated by Algorithm 1, a state feedback controller can be given. Specifically, from Algorithm 1, the result is an optimal policy. Assume that $\mu^*(\mathcal{X}_i,\mathcal{X}_e)$ is the calculation result. Then, we get a state feedback controller $\mu_i^* := \mu^*(\mathcal{X}_i,\mathcal{X}_e)$, $\forall i\in\mathbb{Z}_{1,n}^{+}$.

# 4 Discussion

**Example 1:** Consider a PFSM

$$\mathcal{X}_1(t+1)=\begin{cases}f(\mathcal{X}_1(t),\mathcal{U}_1)=\begin{cases}\mathcal{X}_2, P=0.5\\ \mathcal{X}_5, P=0.5\end{cases}\\ f(\mathcal{X}_1(t),\mathcal{U}_2)=\mathcal{X}_1, P=1.0\\ f(\mathcal{X}_1(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_1, P=0.6\\ \mathcal{X}_3, P=0.4,\end{cases}\end{cases}$$

$$\mathcal{X}_2(t+1)=\begin{cases}f(\mathcal{X}_2(t),\mathcal{U}_1)=\mathcal{X}_2, P=1.0\\ f(\mathcal{X}_2(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_3, P=0.7\\ \mathcal{X}_6, P=0.3\end{cases}\\ f(\mathcal{X}_2(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_2, P=0.5\\ \mathcal{X}_4, P=0.5,\end{cases}\end{cases}$$

$$\mathcal{X}_3(t+1)=\begin{cases}f(\mathcal{X}_3(t),\mathcal{U}_1)=\mathcal{X}_3, P=1.0\\ f(\mathcal{X}_3(t),\mathcal{U}_2)=\mathcal{X}_3, P=1.0\\ f(\mathcal{X}_3(t),\mathcal{U}_3)=\mathcal{X}_3, P=1.0,\end{cases}$$

$$\mathcal{X}_4(t+1)=\begin{cases}f(\mathcal{X}_4(t),\mathcal{U}_1)=\begin{cases}\mathcal{X}_4, P=0.4\\ \mathcal{X}_7, P=0.6\end{cases}\\ f(\mathcal{X}_4(t),\mathcal{U}_2)=\mathcal{X}_5, P=1.0\\ f(\mathcal{X}_4(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_4, P=0.3\\ \mathcal{X}_6, P=0.7,\end{cases}\end{cases}$$

$$\mathcal{X}_5(t+1)=\begin{cases}f(\mathcal{X}_5(t),\mathcal{U}_1)=\mathcal{X}_5, P=1.0\\ f(\mathcal{X}_5(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_3, P=0.9\\ \mathcal{X}_6, P=0.1\end{cases}\\ f(\mathcal{X}_5(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_5, P=0.8\\ \mathcal{X}_9, P=0.2,\end{cases}\end{cases}$$

$$\mathcal{X}_6(t+1)=\begin{cases}f(\mathcal{X}_6(t),\mathcal{U}_1)=\mathcal{X}_3, P=1.0\\ f(\mathcal{X}_6(t),\mathcal{U}_2)=\mathcal{X}_6, P=1.0\\ f(\mathcal{X}_6(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_6, P=0.7\\ \mathcal{X}_7, P=0.3,\end{cases}\end{cases}$$

$$\mathcal{X}_7(t+1)=\begin{cases}f(\mathcal{X}_7(t),\mathcal{U}_1)=\mathcal{X}_7, P=1.0\\ f(\mathcal{X}_7(t),\mathcal{U}_2)=\mathcal{X}_5, P=1.0\\ f(\mathcal{X}_7(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_7, P=0.6\\ \mathcal{X}_8, P=0.4,\end{cases}\end{cases}$$

$$\mathcal{X}_8(t+1)=\begin{cases}f(\mathcal{X}_8(t),\mathcal{U}_1)=\begin{cases}\mathcal{X}_7, P=0.7\\ \mathcal{X}_8, P=0.3\end{cases}\\ f(\mathcal{X}_8(t),\mathcal{U}_2)=\mathcal{X}_9, P=1.0\\ f(\mathcal{X}_8(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_8, P=0.9\\ \mathcal{X}_1, P=0.1,\end{cases}\end{cases}$$

$$\mathcal{X}_9(t+1)=\begin{cases}f(\mathcal{X}_9(t),\mathcal{U}_1)=\mathcal{X}_6, P=1.0\\ f(\mathcal{X}_9(t),\mathcal{U}_2)=\mathcal{X}_9, P=1.0\\ f(\mathcal{X}_9(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_9, P=0.5\\ \mathcal{X}_2, P=0.5,\end{cases}\end{cases}$$

$$\mathcal{X}_{10}(t+1)=\begin{cases}f(\mathcal{X}_{10}(t),\mathcal{U}_1)=\begin{cases}\mathcal{X}_{11}, P=0.3\\ \mathcal{X}_{12}, P=0.7\end{cases}\\ f(\mathcal{X}_{10}(t),\mathcal{U}_2)=\mathcal{X}_{10}, P=1.0\\ f(\mathcal{X}_{10}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{10}, P=0.4\\ \mathcal{X}_{13}, P=0.6,\end{cases}\end{cases}$$

$$\mathcal{X}_{11}(t+1)=\begin{cases} f(\mathcal{X}_{11}(t),\mathcal{U}_1)=\mathcal{X}_{11}, P=1.0 \\ f(\mathcal{X}_{11}(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_{12}, P=0.5\\ \mathcal{X}_{14}, P=0.5\end{cases} \\ f(\mathcal{X}_{11}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{11}, P=0.7\\ \mathcal{X}_{15}, P=0.3,\end{cases}\end{cases}$$

$$\mathcal{X}_{12}(t+1)=\begin{cases} f(\mathcal{X}_{12}(t),\mathcal{U}_1)=\begin{cases}\mathcal{X}_{16}, P=0.6\\ \mathcal{X}_{17}, P=0.4\end{cases} \\ f(\mathcal{X}_{12}(t),\mathcal{U}_2)=\mathcal{X}_{12}, P=1.0 \\ f(\mathcal{X}_{12}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{12}, P=0.8\\ \mathcal{X}_{18}, P=0.2,\end{cases}\end{cases}$$

$$\mathcal{X}_{13}(t+1)=\begin{cases} f(\mathcal{X}_{13}(t),\mathcal{U}_1)=\mathcal{X}_{13}, P=1.0 \\ f(\mathcal{X}_{13}(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_{14}, P=0.9\\ \mathcal{X}_{19}, P=0.1\end{cases} \\ f(\mathcal{X}_{13}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{13}, P=0.5\\ \mathcal{X}_{20}, P=0.5,\end{cases}\end{cases}$$

$$\mathcal{X}_{14}(t+1)=\begin{cases} f(\mathcal{X}_{14}(t),\mathcal{U}_1)=\begin{cases}\mathcal{X}_{15}, P=0.8\\ \mathcal{X}_{16}, P=0.2\end{cases} \\ f(\mathcal{X}_{14}(t),\mathcal{U}_2)=\mathcal{X}_{14}, P=1.0 \\ f(\mathcal{X}_{14}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{17}, P=0.6\\ \mathcal{X}_{18}, P=0.4,\end{cases}\end{cases}$$

$$\mathcal{X}_{15}(t+1)=\begin{cases} f(\mathcal{X}_{15}(t),\mathcal{U}_1)=\mathcal{X}_{15}, P=1.0 \\ f(\mathcal{X}_{15}(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_{16}, P=0.7\\ \mathcal{X}_{20}, P=0.3\end{cases} \\ f(\mathcal{X}_{15}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{19}, P=0.5\\ \mathcal{X}_{15}, P=0.5,\end{cases}\end{cases}$$

$$\mathcal{X}_{16}(t+1)=\begin{cases} f(\mathcal{X}_{16}(t),\mathcal{U}_1)=\mathcal{X}_{16}, P=1.0 \\ f(\mathcal{X}_{16}(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_{17}, P=0.8\\ \mathcal{X}_{18}, P=0.2\end{cases} \\ f(\mathcal{X}_{16}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{16}, P=0.6\\ \mathcal{X}_{19}, P=0.4,\end{cases}\end{cases} \tag{14}$$

$$\mathcal{X}_{17}(t+1)=\begin{cases} f(\mathcal{X}_{17}(t),\mathcal{U}_1)=\mathcal{X}_{17}, P=1.0 \\ f(\mathcal{X}_{17}(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_{18}, P=0.9\\ \mathcal{X}_{20}, P=0.1\end{cases} \\ f(\mathcal{X}_{17}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{17}, P=0.7\\ \mathcal{X}_{19}, P=0.3,\end{cases}\end{cases}$$

$$\mathcal{X}_{18}(t+1)=\begin{cases} f(\mathcal{X}_{18}(t),\mathcal{U}_1)=\mathcal{X}_{18}, P=1.0 \\ f(\mathcal{X}_{18}(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_{19}, P=0.8\\ \mathcal{X}_{20}, P=0.2\end{cases} \\ f(\mathcal{X}_{18}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{18}, P=0.5\\ \mathcal{X}_{1}, P=0.5,\end{cases}\end{cases}$$

$$\mathcal{X}_{19}(t+1)=\begin{cases} f(\mathcal{X}_{19}(t),\mathcal{U}_1)=\mathcal{X}_{19}, P=1.0 \\ f(\mathcal{X}_{19}(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_{20}, P=0.6\\ \mathcal{X}_{1}, P=0.4\end{cases} \\ f(\mathcal{X}_{19}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{19}, P=0.8\\ \mathcal{X}_{2}, P=0.2,\end{cases}\end{cases}$$

$$\mathcal{X}_{20}(t+1)=\begin{cases} f(\mathcal{X}_{20}(t),\mathcal{U}_1)=\mathcal{X}_{20}, P=1.0 \\ f(\mathcal{X}_{20}(t),\mathcal{U}_2)=\begin{cases}\mathcal{X}_{1}, P=0.7\\ \mathcal{X}_{3}, P=0.3\end{cases} \\ f(\mathcal{X}_{20}(t),\mathcal{U}_3)=\begin{cases}\mathcal{X}_{20}, P=0.9\\ \mathcal{X}_{4}, P=0.1,\end{cases}\end{cases}$$

where $\mathcal{X}_i(t)$ represents the $i$-th state of PFSM (14) at time step $t$. It is easy to observe that $\mathcal{X}_3$ is an equilibrium state.

We now use Algorithm 1 to compute a state feedback controller to stabilize PFSM (14) to $\mathcal{X}_3$. The computation is performed on a computer with Intel i5-11300H processor, 2.6 GHz frequency, 16 GB RAM, and Python 3.7 software. We adopt TensorFlow in Keras to train the DQN model, where the discount factor $\gamma$ is 0.99, the rang for $\epsilon$ in $\epsilon$-greedy policy is from 0.05 to 1.0, and the sizes of memory buffer $B$ and mini-batch $M$ are 10,000 and 128, respectively.

Through calculation, we obtain a state feedback controller

$$\mu_i^*=\mu^*(\mathcal{X}_i,\mathcal{X}_3), i\in[1,20], \tag{15}$$

which is shown in Table 1.

Model (14) is a PFSM with 20 states, which is not a simple system. Here, we utilize average rewards to track the performance during training (see Figure 2). It is easy to observe that as training time goes on, the performance inceases and tends to be stable. We put the state feedback controller (15), as shown in Table 1, into PFSM (14) and get a closed-loop system.

$$\mathcal{X}_1(t+1)=\begin{cases}\mathcal{X}_2, P=0.5\\ \mathcal{X}_5, P=0.5,\end{cases} \qquad \mathcal{X}_2(t+1)=\begin{cases}\mathcal{X}_3, P=0.7\\ \mathcal{X}_6, P=0.3,\end{cases}$$

$$\mathcal{X}_3(t+1)=\mathcal{X}_3, P=1.0, \qquad \mathcal{X}_4(t+1)=\begin{cases}\mathcal{X}_4, P=0.4\\ \mathcal{X}_7, P=0.6,\end{cases}$$

$$\mathcal{X}_5(t+1)=\begin{cases}\mathcal{X}_3, P=0.9\\ \mathcal{X}_6, P=0.1,\end{cases} \qquad \mathcal{X}_6(t+1)=\mathcal{X}_3, P=1.0,$$

$$\mathcal{X}_7(t+1)=\mathcal{X}_5, P=1.0, \qquad \mathcal{X}_8(t+1)=\mathcal{X}_9, P=1.0,$$

$$\mathcal{X}_9(t+1)=\mathcal{X}_6, P=1.0, \qquad \mathcal{X}_{10}(t+1)=\begin{cases}\mathcal{X}_{10}, P=0.4\\ \mathcal{X}_{13}, P=0.6,\end{cases}$$

$$\mathcal{X}_{11}(t+1)=\begin{cases}\mathcal{X}_{12}, P=0.5\\ \mathcal{X}_{14}, P=0.5,\end{cases} \qquad \mathcal{X}_{12}(t+1)=\begin{cases}\mathcal{X}_{12}, P=0.8\\ \mathcal{X}_{18}, P=0.2,\end{cases}$$

$$\mathcal{X}_{13}(t+1)=\begin{cases}\mathcal{X}_{13}, P=0.5\\ \mathcal{X}_{20}, P=0.5,\end{cases} \qquad \mathcal{X}_{14}(t+1)=\begin{cases}\mathcal{X}_{17}, P=0.6\\ \mathcal{X}_{18}, P=0.4,\end{cases}$$

$$\mathcal{X}_{15}(t+1)=\begin{cases}\mathcal{X}_{16}, P=0.7\\ \mathcal{X}_{20}, P=0.3,\end{cases} \qquad \mathcal{X}_{16}(t+1)=\begin{cases}\mathcal{X}_{17}, P=0.8\\ \mathcal{X}_{18}, P=0.2,\end{cases}$$

$$\mathcal{X}_{17}(t+1)=\begin{cases}\mathcal{X}_{18}, P=0.9\\ \mathcal{X}_{20}, P=0.1,\end{cases} \qquad \mathcal{X}_{18}(t+1)=\begin{cases}\mathcal{X}_{18}, P=0.5\\ \mathcal{X}_{1}, P=0.5,\end{cases}$$

$$\mathcal{X}_{19}(t+1)=\begin{cases}\mathcal{X}_{20}, P=0.6\\ \mathcal{X}_{1}, P=0.4,\end{cases} \qquad \mathcal{X}_{20}(t+1)=\begin{cases}\mathcal{X}_{1}, P=0.7\\ \mathcal{X}_{3}, P=0.3.\end{cases} \tag{16}$$

The state transition trajectory of the closed-loop system (16) starting from any initial state is shown in Figure 3. It can be

observed from Figure 3 that all states reach $\mathcal{X}_3$ after a finite number of steps and then stay at $\mathcal{X}_3$ forever with probability one. This demonstrates the effectiveness of our controller. The number of steps required to reach $\mathcal{X}_3$ for each state is shown in Figure 4. From these results, we can observe that based on DQN, Algorithm 1 can solve the stabilization problem of non-small -scale PFSMs.

## 5 Conclusion

This article studied the state feedback stabilization of PFSMs using the DQN method. The feedback stabilization problem of PFSMs was first transformed into an optimization problem. A DQN was built, whose two key parts: TD target and Q function, are approximated through neural networks. Then, based on the DQN and a stabilizability condition derived in this paper, an algorithm was developed. The algorithm can be used to calculate the optimization problem mentioned above and then solves the feedback stability problem of PFSMs. Since DQN avoids the limited capacity problem of Q learning, our algortithm can handle high-dimensional complex systems. Finally, an illustrative example is provided to show the effectiveness of our method.

## Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

## Author contributions

HT: Conceptualization, Formal analysis, Funding acquisition, Methodology, Project administration, Supervision, Writing—review & editing. XS: Conceptualization, Formal analysis, Investigation, Methodology, Validation, Writing—original draft. YH: Formal analysis, Investigation, Writing—review & editing.

## Funding

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. This research was funded by the National Key R&D Program of China (2021YFB3203202) and Chongqing Nature Science Foundation (cstc2020jcyj-msxmX0708).

## Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

## References

Acernese, A., Yerudkar, A., Glielmo, L., and Vecchio, C. D. (2020). Double deep-Q learning-based output tracking of probabilistic Boolean control networks. *IEEE Access* 8, 199254–199265. doi: 10.1109/ACCESS.2020.3035152

Ding, S., Wang, Z., and Zhang, H. (2019). Quasi-synchronization of delayed memristive neural networks via region-partitioning-dependent intermittent control. *IEEE Trans. Cybern.* 49, 4066–4077. doi: 10.1109/TCYB.2018.2856907

El-Maleh, A., and Al-Qahtani, A. (2014). A finite state machine based fault tolerance technique for sequential circuits. *Microelectron. Reliab.* 54, 654–661. doi: 10.1016/j.microrel.2013.10.022

Fadhil, A., Wang, Y., and Reiterer, H. (2019). Assistive conversational agent for health coaching: a validation study. *Methods Inf. Med.* 58, 9–23. doi: 10.1055/s-0039-1688757

Han, X., and Chen, Z. (2018). A matrix-based approach to verifying stability and synthesizing optimal stabilizing controllers for finite-state automata. *J. Franklin Inst.* 355, 8642–8663. doi: 10.1016/j.jfranklin.2018.09.009

Kobayashi, K. (2006). "Modeling of discrete dynamics for computational time reduction of model predictive control," in *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems* (Tokyo), 628–633.

Kobayashi, K., and Imura, J. (2007). "Minimalilty of finite automata representation in hybrid systems control," in *Hybrid Systems: Computation and Control* (Berlin Heidelberg: Springer), 343–356. doi: 10.1007/978-3-540-71493-4_28

Kobayashi, K., Imura, J., and Hiraishi, K. (2011). Stabilization of finite automata with application to hybrid systems control. *Discret. Event Dyn. Syst.* 21, 519–545. doi: 10.1007/s10626-011-0110-2

Li, J., and Tan, Y. (2019). A probabilistic finite state machine based strategy for multi-target search using swarm robotics. *Appl. Soft Comput.* 77, 467–483. doi: 10.1016/j.asoc.2019.01.023

Özveren, C., Willsky, A., and Antsaklis, P. (1991). Stability and stabilizability of discrete event dynamic systems. *J. ACM* 38, 729–751. doi: 10.1145/116825.116855

Passino, K., Michel, A., and Antsaklis, P. (1994). Lyapunov stability of a class of discrete event systems. *IEEE Trans. Automat. Contr.* 39, 269–279. doi: 10.1109/9.272323

Piccinini, A., Previdi, F., Cimini, C., Pinto, R., and Pirola, F. (2018). Discrete event simulation for the reconfiguration of a flexible manufactuing plant. *IFAC-PapersOnLine* 51, 465–470. doi: 10.1016/j.ifacol.2018.08.362

Ratsaby, J. (2019). On deterministic finite state machines in random environments. *Probab. Eng. Inf. Sci.* 33, 528–563. doi: 10.1017/S0269964818000451

Shah, S., Velardo, C., Farmer, A., and Tarassenko, L. (2017). Exacerbations in chronic obstructive pulmonary disease: identification and prediction using a digital health system. *J. Med. Internet Res.* 19:e69. doi: 10.2196/jmir.7207

Tarraf, D., Megretski, A., and Dahleh, M. (2008). A framework for robust stability of systems over finite alphabets. *IEEE Trans. Automat. Contr.* 53, 1133–1146. doi: 10.1109/TAC.2008.923658

Tian, H., and Hou, Y. (2019). State feedback design for set stabilization of probabilistic boolean control networks. *J. Franklin Inst.* 356, 4358–4377. doi: 10.1016/j.jfranklin.2018.12.027

Tian, H., Zhang, H., Wang, Z., and Hou, Y. (2017). Stabilization of k-valued logical control networks by open-loop control via the reverse-transfer method. *Automatica* 83, 387–390. doi: 10.1016/j.automatica.2016.12.040

Tian, Y., and Wang, Z. (2020). A new multiple integral inequality and its application to stability analysis of time-delay systems. *Appl. Math. Lett.* 105:106325. doi: 10.1016/j.aml.2020.106325

Vayadande, K., Sheth, P., Shelke, A., Patil, V., Shevate, S., Sawakare, C., et al. (2022). Simulation and testing of deterministic finite automata machine. *International Journal of Comput. Sci. Eng.* 10, 13–17. doi: 10.26438/ijcse/v10i1.1317

Vidal, E., Thollard, F., de la Higuera, C., Casacuberta, F., and Carrasco, R. (2005). Probabilistic finite-state machines - part I. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27, 1013–1025. doi: 10.1109/TPAMI.2005.147

Wang, L., Zhu, B., Wang, Q., and Zhang, Y. (2017). Modeling of hot stamping process procedure based on finite state machine (FSM). *Int. J. Adv. Manuf. Technol.* 89, 857–868. doi: 10.1007/s00170-016-9097-z

Xu, X., Zhang, Y., and Hong, Y. (2013). "Matrix approach to stabilizability of deterministic finite automata," in *2013 American Control Conference* (Washington, DC), 3242–3247.

Yan, Y., Chen, Z., and Liu, Z. (2015a). Semi-tensor product approach to controllability and stabilizability of finite automata. *J. Syst. Eng. Electron.* 26, 134–141. doi: 10.1109/JSEE.2015.00018

Yan, Y., Chen, Z., and Yue, J. (2015b). Stp approach to controlliability of finite state machines. *IFAC-PapersOnLine* 48, 138–143. doi: 10.1016/j.ifacol.2015.12.114

Zhang, X. (2018). Application of discrete event simulation in health care: a systematic review. *BMC Health Serv. Res.* 18, 1–11. doi: 10.1186/s12913-018-3456-4

Zhang, Z., Chen, Z., Han, X., and Liu, Z. (2020a). Stabilization of probabilistic finite automata based on semi-tensor product of matrices. *J. Franklin Inst.* 357, 5173–5186. doi: 10.1016/j.jfranklin.2020.02.028

Zhang, Z., Xia, C., and Chen, Z. (2020b). On the stabilization of nondeterministic finite automata via static output feedback. *Appl. Math. Comput.* 365:124687. doi: 10.1016/j.amc.2019.124687