Neural Computing and Applications (2020) 32:14311–14320
https://doi.org/10.1007/s00521-019-04180-2

EXTREME LEARNING MACHINE AND DEEP LEARNING NETWORKS

Reinforcement learning and adaptive optimization of a class of Markov jump systems with completely unknown dynamic information

Shuping He[1,2] · Maoguang Zhang[1] · Haiyang Fang[1] · Fei Liu[3] · Xiaoli Luan[3] · Zhengtao Ding[4]

Received: 29 November 2018 / Accepted: 29 March 2019 / Published online: 9 April 2019

Abstract

In this paper, an online adaptive optimal control problem of a class of continuous-time Markov jump linear systems (MJLSs) is investigated by using a parallel reinforcement learning (RL) algorithm with completely unknown dynamics. Before collecting and learning the subsystems information of states and inputs, the exploration noise is firstly added to describe the actual control input. Then, a novel parallel RL algorithm is used to parallelly compute the corresponding N coupled algebraic Riccati equations by online learning. By this algorithm, we will not need to know the dynamic information of the MJLSs. The convergence of the proposed algorithm is also proved. Finally, the effectiveness and applicability of this novel algorithm is illustrated by two simulation examples.

Keywords Markov jump linear systems (MJLSs) · Adaptive optimal control · Online · Reinforcement learning (RL) · Coupled algebraic Riccati equations (AREs)

1 Introduction

Markov jump linear systems (MJLSs), firstly proposed by Krasovskii and Lidskii [1] in 1961, can be considered as a kind of multi-model stochastic systems. In MJLSs, it contains two mechanisms, i.e., the modes and the states. The modes are jumping dynamics, modeled by finite-state Markov chains. The states are continuous or discrete, modeled by a set of differential or difference equations. With the development of control science and stochastic theory, MJLSs have been widely concerned and many research results are available, such as stochastic stability and stabilizability [2–4], controllability [5–9] and robust estimation and filtering [10–12].

In recent years, the adaptive optimal control problem has become a focused issue in controllers design and many related works have been published. For example, the authors in [13] studied the adaptive surface optimal control methods for strict-feedback systems. Then, the observer-based adaptive fuzzy control law was proposed for non-linear nonstrict-feedback systems [14]. A general method to solve the optimal control problem related to the related algebraic Riccati equations (AREs) was studied in [15]. In [16], Kleinman considered an offline iterative technique to solve the AREs. In [17, 18], the iterative algorithm and the temporal difference method were, respectively, proposed to solve the discrete-time coupled AREs. To get the solutions of a set of coupled AREs for MJLSs, a Lyapunov iteration method [19] was derived. For more related researches, we can refer to [20–24]. However, all these algorithms are

✉ Shuping He
shuping.he@ahu.edu.cn

Haiyang Fang
fangocean1996@gmail.com

Fei Liu
fliu@jiangnan.edu.cn

Xiaoli Luan
xiaoli_luan@126.com

Zhengtao Ding
zhengtao.ding@manchester.ac.uk

1 School of Electrical Engineering and Automation, Anhui University, Hefei 230601, China

2 Institute of Physical Science and Information Technology, Anhui University, Hefei 230601, China

3 Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Institute of Automation, Jiangnan University, Wuxi 214122, China

4 School of Electrical and Electronic Engineering, The University of Manchester, Manchester M13 9PL, UK

offline and have limitations to implement when the system dynamics are unknown.

Reinforcement learning (RL) and approximate/adaptive dynamic programming (ADP) algorithms are widely applied to solve the optimal control problems [25–28]. Moreover, we can use RL and ADP algorithm to obtain the optimal controller solutions through online learning. In [29], an online feedback control scheme is designed for linear systems by applying RL and ADP method. Then, it was developed to solve the zero-sum differential game [30]. In [31, 32], an adaptive policy iteration algorithm is considered for continuous-time and discrete-time nonlinear systems, respectively. Then, the reinforcement Q-learning for optimal tracking control problem was studied for linear discrete-time systems with unknown dynamics [33] and nonlinear discrete-time systems with unknown nonaffine dead-zone input [34]. Applying the H_∞ state feedback scheme, an online policy update algorithm was proposed [35]. For more results on this studying, we refer readers to [36–44].

It is worth pointing out the relevant methods related to RL and ADP are mostly used to solve the optimal controller design problems for general linear/nonlinear dynamics. Due to the complexity of MJLSs which contain states and modes, the mode-based switching subsystems have coupling relationships. Obviously, the RL and ADP algorithms proposed above can not directly solve the adaptive optimal control problems of MJLSs. To study the online control problem for MJLSs, the authors in [45] considered an online adaptive optimal control scheme by applying a novel policy iteration algorithm. Then, the relevant results were developed to the Itô stochastic systems [46] and the H_∞ optimal problem [47] of MJLSs. But in these results, we must partially know the system dynamics. Therefore, these published results can be considered as partial model-free ones.

In this paper, we propose a novel parallel RL algorithm to solve the optimal control problem of MJLSs with completely unknown dynamics. This algorithm is based on the subsystem transformation technique and the parallel RL algorithm. The subsystem transformation technique is used to obtain the N AREs. Then, we can parallelly solve the corresponding N AREs by collecting and learning the subsystems information of states and inputs. Comparing with the published results in [45, 46], we will not use the system dynamics in this paper and the exploration noise is added to describe the actual control input. Moreover, the solutions can be conducted directly in one step by repeated iteration and computation without using policy evaluation and improvement. It is clear that the iterative algorithm in this paper can be effective to study the online optimal control problems for MJLSs. The convergence has also been proved. In order to demonstrate the effectiveness of the designed method, two simulation examples are given at last.

The main contributions of this paper can be concluded as:

1. A novel parallel RL algorithm with completely unknown dynamic information is first proposed to solve the optimal control problem of MJLSs;
2. To deal with the jumping modes, we use the subsystem transformation technique to decompose the MJLSs.
3. Compared with the results in [19], we apply the online collecting and learning the subsystems information of states and inputs to obtain the dynamics information. Then, we optimize and update the iterative results to obtain the optimal control policy. The designed PI algorithm is online and does not require the dynamic information of the MJLSs.

This paper contains the following five sections. The problem formulation and preliminaries are introduced in Sect. 2. The novel parallel iteration algorithm is proposed in Sect. 3.1, and the online implementation of the algorithm is given in Sect. 3.2; Sect. 3.3 gives the convergence proof of the designed algorithm. Two simulation examples are shown in Sect. 4. A conclusion is given in Sect. 5.

Notations Throughout this paper, $\Re^n$ is used to denote an n-dimensional real matrix; $\lambda(A)$ denotes the eigenvalue of A; $\mathrm{rank}\,(A)$ denotes the rank of A; Z_+ denotes a non-negative–positive integer set; $\mathbf{E}\{\cdot\}$ shows the mathematical expectation of stochastic processes or vectors; $\|\cdot\|$ represents the Euclidean norm; $\otimes$ is used to represent the Kronecker product which has the following properties: (1) $(A\otimes B)^{\mathrm{T}} = A^{\mathrm{T}}\otimes B^{\mathrm{T}}$, (2) $x^{\mathrm{T}}Px = (x^{\mathrm{T}}\otimes x^{\mathrm{T}})\mathrm{vec}\,(P)$. For a matrix $A\in\Re^{n\times n}$, we denote $\mathrm{vec}\,(A) = [a_1^{\mathrm{T}},\ a_2^{\mathrm{T}},\ \ldots,\ a_n^{\mathrm{T}}]^{\mathrm{T}}$, where $a_i\in\Re^n$ is the ith column of A.

2 Backgrounds and preliminaries

2.1 Problem description

Consider a class of continuous-time Markov jump linear systems (MJLSs) described by:

$$\begin{cases} \dot{x}(t) = A(r(t))x(t) + B(r(t))u(t) \\ x(0) = x_0,\ r(0) = r_0, t = 0 \end{cases} \tag{1}$$

with the infinite horizon performance index:

$$J(x(t)) = \mathbf{E}\left\{\int_0^\infty \left[x^{\mathrm{T}}(t)Q_i x(t) + u^{\mathrm{T}}(t)R_i u(t)\right]\mathrm{d}t \middle| t = 0,\ x_0,\ r_0\right\} \tag{2}$$

where $x(t) \in \Re^n$ is the system state with the initial value x_0; $u(t) \in \Re^m$ is the control input; $A(r(t))$, $B(r(t))$ are the coefficient matrices, $Q(r(t))$ and $R(r(t))$ are the weight matrices which are mode-dependent and positive-definite; the pair $(A(r(t)),\ Q^{1/2}(r(t)))$ is supposed to be observable; $r(t)$ is a continuous-time and discrete-state right continuous Markov stochastic process which represents the system mode and takes on values in a discrete set $M = \{1, 2, \ldots, N\}$; r_0 represents the initial mode. Here, MJLSs (1) is assumed to be stochastically stabilizable.

For MJLSs (1), the transition probabilities of each mode satisfy:

$$P_r\{r(t+\Delta t) = j | r(t) = i\} = \begin{cases} \pi_{ij}\Delta t + o(\Delta t) & j \neq i \\ 1 + \pi_{ii}\Delta t + o(\Delta t) & j = i \end{cases} \tag{3}$$

where $i, j \in M$, $\Delta t > 0$ and $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$; $\pi_{ij} \geq 0\, (i \neq j)$ represents the transition rate from mode i to mode j at time $t \to t + \Delta t$ and has $\pi_{ii} = -\sum_{i \neq j} \pi_{ij}$. For simplicity, when $r(t) = i$, $A(r(t))$, $B(r(t))$, $Q(r(t))$, $R(r(t))$ can be labeled as A_i, B_i, Q_i, R_i, respectively.

For a given feedback control policy $u(t) = -K_i x(t)$, we define the following cost function as:

$$V(x(t)) = \mathbf{E}\left\{ \int_t^\infty \left[x^{\mathrm{T}}(t) Q_i x(t) + u^{\mathrm{T}}(t) R_i u(t)\right] \mathrm{d}t \middle| t, x(t), r(t) \right\} \tag{4}$$

The optimal control problem for MJLSs (1) is to find a mode-dependent admissible feedback control policy, which minimizes the performance index, i.e.

$$\begin{aligned} V^*(x(t)) &= \min_{u(t)} J(x(t)) \\ &= \min_{u(t)} \mathbf{E}\left\{ \int_0^\infty \left[x^{\mathrm{T}}(t) Q_i x(t) + u^{\mathrm{T}}(t) R_i u(t)\right] \mathrm{d}t \middle| t = 0,\ x_0,\ r_0 \right\} \end{aligned} \tag{5}$$

Then, the cost function (4) can be denoted as $V(x(t)) = x_{i,\,t}^{\mathrm{T}} P_i x_{i,\,t}$, where $x_{i,\,t}$ is the state of the ith jump mode; P_i is a positive-definite and mode-dependent matrix which can be obtained through solving the following coupled algebraic Riccati equations (AREs):

$$\hat{A}_i^{\mathrm{T}} P_i + P_i \hat{A}_i + Q_i + \sum_{j=1, i \neq j}^{N} \pi_{ij} P_j - P_i B_i R_i^{-1} B_i^{\mathrm{T}} P_i = 0 \tag{6}$$

where $\hat{A}_i = A_i + \frac{\pi_{ii}}{2} I$. Then, the optimal feedback control policy can be determined by:

$$u(t) = -K_i x(t) = -R_i^{-1} B_i^{\mathrm{T}} P_i x(t). \tag{7}$$

Definition 1 MJLSs (1) is stochastically stable (SS), if for any initial conditions x_0 and r_0, the following formula is satisfied:

$$\lim_{t \to \infty} \mathbf{E}\, \|x(t)\|^2 = 0. \tag{8}$$

Inspired by [48], according to the subsystem transformation technique, we can re-write MJLSs (1) as the following N re-constructed subsystems:

$$\begin{cases} \dot{x}_{i,\,t} = \hat{A}_i x_{i,\,t} + B_i u_{i,\,t} \\ x_i(0) = x_{i,0}, \quad t_0 = 0\ . \end{cases} \tag{9}$$

In order to guarantee the stabilizability of the N re-constructed subsystems in Eq. (9) by the optimal feedback control policy (7), the triples $(A_i,\ B_i,\ \sqrt{Q_i})$ are assumed to be stabilizable-detectable [19].

2.2 An offline parallel policy iteration algorithm (OPPIA)

According to the assumptions mentioned above, the exact solution P_i^* of the positive-definite and mode-dependent matrix P_i can be obtained by solving AREs (6). Notice that AREs (6) is a set of nonlinear algebraic equations and it is difficult to be solved directly, especially for large dimensional matrices. To tackle this problem, some algorithms have been proposed by Kleinman [16] and Gajic and Borno [19]. For MJLSs (1), we give the following Lemma 1.

Lemma 1 [19] For each subsystems, we give an initial stabilizable value $K_{i(0)}$. The solutions of the following decoupled algebraic Lyapunov equations are equivalent to the solutions of AREs (6).

$$\begin{aligned} &(\hat{A}_i - B_i K_{i(k)})^{\mathrm{T}} P_{i(k+1)} + P_{i(k+1)} (\hat{A}_i - B_i K_{i(k)}) \\ &\quad = -Q_{i(k)} - P_{i(k)} B_i R_i^{-1} B_i^{\mathrm{T}} P_{i(k)} \end{aligned} \tag{10}$$

In order to solve the N parallel decoupled algebraic Lyapunov equations in (10), we give the following Algorithm 1.

Remark 1 Due to the fact that MJLSs (1) can be decomposed into N independent subsystems, the solutions can be obtained by parallel solving N decoupled algebraic Lyapunov linear equations. Thus, the solutions to the N parallel nonlinear AREs (6) are numerically approximated. By Algorithm 1, the exact solutions of P_i^* can be obtained while $k \to \infty$. Notice that the method in Algorithm 1 is offline and the knowledge of the dynamics A_i, B_i are all needed.

Algorithm 1. OPPIA

Step 1. Subsystem Transformation. Re-construct MJLSs (1) into the N independent subsystems in equation (9).

Step 2. Give N initial stabilizable vaules, i.e., when $k=0$, let $\{P_{1(0)}, \cdots, P_{N(0)}\}$, $\lambda(\hat{A}_i - B_i K_{i(k)}) < 0$.

Step 3. Parallel Policy Iteration:
Solve the N parallel decoupled algebraic Lyapunov equations in (10).
Parallel Policy Update:
Derive $Q_{i(k)}$ and $K_{i(k)}$ by the following recursions:

$$\begin{cases} K_{i(k)} = R_i^{-1} B_i^T P_{i(k-1)} \\ Q_{i(k)} = Q_i + \sum_{j=1, i\neq j}^{N} \pi_{ij} P_{j(k-1)} \geq 0 \end{cases} \tag{11}$$

Step 4. Compute $P_{i(k)}$ and $K_{i(k)}$ by solving step 3, until $\|P_{(k+1)} - P_{(k)}\| \leq \varepsilon$ for $k \geq 1$; Otherwise, set $k = k+1$, go to Step 2 and continue the iteration process.

Step 5. Obtain the optimal control policy $u_{i,t} = -K_{i(k)} x_{i,t}$.

3 Main results

In this part, we will propose a parallel reinforcement learning (RL) algorithm to design the feedback control policy for MJLSs (1). This is a novel online learning strategy which will not require the knowledge of the coefficient matrices, i.e., A_i and B_i. For each solution step, we can solve N parallel decoupled algebraic Lyapunov equations in (10). In the end, the convergence of this new algorithm is also proved.

3.1 A parallel RL algorithm with completely unknown dynamics

For the offline Algorithm 1, the N subsystems dynamics are required to solve the decoupled algebraic Lyapunov equations in (10). For practical applications, this algorithm will be difficult to achieve if we do not know the exact knowledge in advance. To overcome these weaknesses, a new online parallel RL algorithm is developed to study the optimal solutions of MJLSs with completely unknown dynamics. For the ith subsystem, the unique symmetric positive-definite solutions of $P_{i(k)}$ and the state feedback gains $K_{i(k)}$ $(\forall k \in Z_+)$ should satisfy Eqs. (10), (11).

Recalling to Eq. (9), we rewrite the system state equation as:

$$\begin{cases} \dot{x}_{i,t} = \bar{A}_{i(k)} x_{i,t} + B_i(u_{i,t} + K_{i(k)} x_{i,t}) \\ x_i(0) = x_{i,0}, \quad t_0 = 0 \end{cases} \tag{12}$$

where $\bar{A}_i = \hat{A}_i - B_i K_{i(k)}$ and the pairs $(\hat{A}_i, B_i)$ are stabilizable.

Then, the novel algorithm is given as follows:

Algorithm 2 A parallel RL algorithm with completely unknown dynamics.

Step 1. Subsystem Transformation. Re-construct MJLSs (1) into the N independent subsystems in equation (12).

Step 2. Give a initial stabilizable value $K_{i(0)}$ and employ $u_{i,t} = -K_{i(0)} x_{i,t} + e_i$ as the control input in time interval $[t_0, t_l]$.

Step 3. Solve equations (10)-(11) by using the online parallel RL algorithm:

$$x_{i,t+T}^{\mathrm{T}} P_{i(k)} x_{i,t+T} - x_{i,t}^{\mathrm{T}} P_{i(k)} x_{i,t} = -\int_t^{t+T} x_{i,\tau}^{\mathrm{T}} \bar{Q}_{i(k)} x_{i,\tau} \mathrm{d}\tau + 2\int_t^{t+T} [(u_{i,\tau} + K_{i(k)} x_{i,\tau})^{\mathrm{T}} R_i K_{i(k+1)} x_{i,\tau}] \mathrm{d}\tau \tag{13}$$

where $\bar{Q}_{i(k)} = Q_i + \sum_{j=1, i\neq j}^{N} \pi_{ij} P_{j(k-1)} + K_{i(k)}^{\mathrm{T}} R_i K_{i(k)}$.

Step 4. Compute $P_{i(k)}$ and $K_{i(k)}$ in equation (13), until $\|P_{(k+1)} - P_{(k)}\| \leq \varepsilon$ for $k \geq 1$; Otherwise, set $k = k+1$, go to Step 2 and continue the iteration process.

Step 5. Obtain the optimal control policy $u_{i,t} = -K_{i(k)} x_{i,t}$.

Notice that the N subsystems in Eq. (9) are reconstructed into N new continuous-time subsystems in Eq. (12). Next, we will give an online implementation of the parallel RL algorithm to solve Eq. (13). In Eq. (12), A_i and π_{ii} are embedded in the reconstructed subsystem states; B_i is embedded in the reconstructed subsystem control inputs, which can be measured online. For real-time learning of the information of $x_{i,t}$ and $u_{i,t}$, we can choose $u_{i,t} = -K_{i(k)} x_{i,t} + e_i$ as a true control input, where e_i is an additional exploration noise.

Remark 2 Solving Eq. (13), we can get a series of solutions, where $P_{i(k)}$, $K_{i(k)}$ satisfy (10), (11), $x_{i,t}$ is the solution of Eq. (12) with the arbitrary control input $u_{i,t}$. This fact enables us to choose $u_{i,t} = -K_{i(k)} x_{i,t} + e_i$ as a real input signal for learning. Obviously, this operation will not affect the convergence of the learning process.

Remark 3 In Algorithm 2, we choose an exploration noise and assume that it is a nonzero measurable bounded signal. In fact, the choice of the exploration noise will not affect the algorithm. We add the exploration noise as an excitation signal to satisfy the persistence of the excitation conditions. For convenience, we always choose the exponentially decreasing probing noise and the sum of sinusoidal signals as the exploration noise.

Remark 4 In Eq. (13), the subsystems internal dynamics information is not needed, i.e., the dynamic matrices of A_i, B_i are all not included. In order to solve Eq. (13) by the online RL algorithm, we should continuously sample the states and the inputs information online from the N subsystems at each iteration step. In fact, the dynamics information is already included in the sample information. Therefore, the online Algorithm 2 did not require the dynamics information, i.e., Algorithm 2 as a completely model-free algorithm.

3.2 Online implementation of the novel parallel RL algorithm

To find a set of pairs $(P_{i(k)}, K_{i(k+1)})$ which satisfy Eqs. (10), (11), the novel online parallel RL implement approach is derived. It observes that the dynamics matrices A_i, B_i and the transfer matrix element π_{ii} are not included in Algorithm 2. According to the Kronecker product representation, the left side of Eq. (13) can be rewritten as:

$$x_{i,t+T}^{\mathrm{T}} P_{i(k)} x_{i,t+T} - x_{i,t}^{\mathrm{T}} P_{i(k)} x_{i,t} = \bar{P}_{i(k)}^{\mathrm{T}} \left[x_{i,t+T}^{\mathrm{T}} \otimes x_{i,t+T}^{\mathrm{T}} - x_{i,t}^{\mathrm{T}} \otimes x_{i,t}^{\mathrm{T}} \right] = \bar{P}_{i(k)}^{\mathrm{T}} \delta_{ix} \tag{14}$$

where $\bar{P}_{i(k)}$ is a set of positive-definite and mode-dependent matrices; $\bar{P}_i$ is a column vector-valued matrix whose internal elements are produced by P_i and can be given as:

$$\bar{P}_i = [p_{i11},\ 2p_{i12},\ \ldots,\ 2p_{i1n},\ p_{i22},\ 2p_{i23},\ \ldots,\ p_{inn}]^{\mathrm{T}}, \tag{15}$$

$$x_i^{\mathrm{T}} \otimes x_i^{\mathrm{T}} = [x_{i1}^2,\ x_{i1}x_{i2},\ \ldots,\ x_{i1}x_{in},\ x_{i2}^2,\ x_{i2}x_{i3},\ \ldots x_{i2}x_{in},\ \ldots,\ x_{in}^2]^{\mathrm{T}}. \tag{16}$$

Using the similar method as Eq. (14), we have:

$$x_{i,t}^{\mathrm{T}}\, \bar{Q}_{i(k)}\, x_{i,t} = (x_{i,t}^{\mathrm{T}} \otimes x_{i,t}^{\mathrm{T}})\ \mathrm{vec}\,(\bar{Q}_{i(k)}) \tag{17}$$

where $\bar{Q}_{i(k)} = Q_{i(k)} + K_{i(k)}^{\mathrm{T}} R_i K_{i(k)}$. Then, the right side of Eq. (13) can be rewritten as:

$$\begin{aligned}(u_{i,t} + K_{i(k)} x_{i,t})^{\mathrm{T}} R_i K_{i(k+1)} x_{i,t} &= [(x_{i,t}^{\mathrm{T}} \otimes x_{i,t}^{\mathrm{T}})(I_n \otimes K_{i(k)}^{\mathrm{T}} R_i) \\ &+ (x_{i,t}^{\mathrm{T}} \otimes u_{i,t}^{\mathrm{T}})(I_n \otimes R_i)]\ \mathrm{vec}(K_{i(k+1)}).\end{aligned} \tag{18}$$

Meanwhile, we define the following operators:

$$I_{ixx} = \begin{bmatrix} \int_{t_0}^{t_1} x_{i,\tau}^{\mathrm{T}} \otimes x_{i,\tau}^{\mathrm{T}} \mathrm{d}\tau \\ \int_{t_1}^{t_2} x_{i,\tau}^{\mathrm{T}} \otimes x_{i,\tau}^{\mathrm{T}} \mathrm{d}\tau \\ \vdots \\ \int_{t_{l-1}}^{t_l} x_{i,\tau}^{\mathrm{T}} \otimes x_{i,\tau}^{\mathrm{T}} \mathrm{d}\tau \end{bmatrix}, \tag{19}$$

$$I_{ixu} = \begin{bmatrix} \int_{t_0}^{t_1} x_{i,\tau}^{\mathrm{T}} \otimes u_{i,\tau}^{\mathrm{T}} \mathrm{d}\tau \\ \int_{t_1}^{t_2} x_{i,\tau}^{\mathrm{T}} \otimes u_{i,\tau}^{\mathrm{T}} \mathrm{d}\tau \\ \vdots \\ \int_{t_{l-1}}^{t_l} x_{i,\tau}^{\mathrm{T}} \otimes u_{i,\tau}^{\mathrm{T}} \mathrm{d}\tau \end{bmatrix}. \tag{20}$$

Thus, Eq. (13) can be described as:

$$\Theta_{i(k)} \begin{bmatrix} \bar{P}_{i(k)} \\ \mathrm{vec}(K_{i(k+1)}) \end{bmatrix} = \Xi_{i(k)} \tag{21}$$

where $\Theta_{i(k)} \in \Re^{l \times [\frac{1}{2}n(n+1)+mn]}$, $\Xi_{i(k)} \in \Re^{l}$ and they can be described as:

$$\Theta_{i(k)} = \begin{bmatrix} \delta_{ixx} & -2\,[I_{ixx}(I_n \otimes K_{i(k)}^{\mathrm{T}} R_i) + I_{ixu}(I_n \otimes R_i)] \end{bmatrix}, \tag{22}$$

$$\Xi_{i(k)} = -I_{ixx} \mathrm{vec}(\bar{Q}_{i(k)}). \tag{23}$$

For Eq. (21), if $\Theta_{i(k)}$ has full column rank, the least-squares solutions of Eq. (13) can be solved by:

$$\begin{bmatrix} \bar{P}_{i(k)} \\ \mathrm{vec}(K_{i(k+1)}) \end{bmatrix} = (\Theta_{i(k)}^{\mathrm{T}} \Theta_{i(k)})^{-1} \Theta_{i(k)}^{\mathrm{T}} \Xi_{i(k)} \tag{24}$$

By formula (24), we can obtain $P_{i(k)}$ and $K_{i(k+1)}$. To compute the matrices of I_{ixx} and I_{ixu}, we can use $\frac{n(n+1)}{2} + nm$ integrators to collect the information of the states and the control inputs of each subsystems. In order to guarantee that Algorithm 2 can be online implemented at each step, $\Theta_{i(k)}$ is required to have full column rank for all $k \in Z_+$, i.e., there is an integer $l_0 > 0$, for $\forall l > l_0$, the following condition holds:

$$\mathrm{rank}\ (\ [I_{ixx},\ I_{ixu}]\ ) = \frac{n(n+1)}{2} + nm. \tag{25}$$

3.3 The convergence proof

In order to prove the convergence of Algorithm 2, we establish the following Theorem 1.

Theorem 1 For the novel parallel RL Algorithm 2, we give a stabilizable value $K_{i(0)}$, where $\forall i \in M$ and assume that $\Theta_{i(k)}$ has the full column rank; then the solutions of Eq. (13) satisfy Eqs. (10), (11).

Proof Give a stabilizable value $K_{i(0)}$, i.e., $A_i - B_i K_{i(0)}$ is a Hurwitz. According to Algorithm 1, the exact solutions $P_i^* = \lim_{k\to\infty} P_{i(k)}$ can be obtained by the decoupled algebraic Lyapunov equations in (10), and the feedback gain $K_i^* = \lim_{k\to\infty} K_{i(k)}$ can be uniquely determined by Eq. (11). For $\forall t > 0$, we choose the cost function as $V(x(t)) = x_{i,t}^{T} P_i x_{i,t}$. Then, for the new re-constructed ith N independent subsystems in Eq. (12) and along the solutions of (12) by (10) and (11), the following is satisfied:

$$\begin{aligned} & x_{i,t+T}^{\mathrm{T}} P_{i(k)} x_{i,t+T} - x_{i,t}^{\mathrm{T}} P_{i(k)} x_{i,t} \\ & = \int_t^{t+T} \left[x_{i,\tau}^{\mathrm{T}} (\bar{A}_i^{\mathrm{T}} P_{i(k)} + P_{i(k)} \bar{A}_i) x_{i,\tau} + 2(u_{i,\tau} + K_{i(k)} x_{i,\tau})^{\mathrm{T}} B_i^{\mathrm{T}} P_{i(k)} \right] \mathrm{d}\tau \\ & = -\int_t^{t+T} x_{i,\tau}^{\mathrm{T}} \bar{Q}_{i(k)} x_{i,\tau} \mathrm{d}\tau + \int_t^{t+T} 2(u_{i,\tau} + K_{i(k)} x_{i,\tau})^{\mathrm{T}} R_i K_{i(k+1)} x_{i,\tau} \mathrm{d}\tau \end{aligned} \tag{26}$$

Since $\Theta_{i(k)}$ has full column rank, the least-squares solutions in Eq. (24) is an online implement of Eq. (13). Therefore, the solutions of Eq. (13) are equivalent to the iteration solutions of Eqs. (10) and (11). The proof is completed.

The flowchart of Algorithm 2 is given in Fig. 1.

Remark 5 By the online implementation in Algorithm 2, we can see the differences between the offline Algorithm 1 and the online Algorithm 2. In Algorithm 1, we apply the offline iterative algorithm to solve the Riccati functions and equations directly, while in Algorithm 2, we use the online sampling and learning method. Moreover, Eqs. (10), (11) in Algorithm 1 depend on the dynamics information. But in Algorithm 2, we do not need any dynamics information by Eq. (13).

4 Simulation examples

Example 1 To show the applicability of Algorithm 2, we consider a fourth order jump linear model with two jump modes as:

$$N = 2,\ \Pi = \begin{bmatrix} -3 & 3 \\ 2.5 & -2.5 \end{bmatrix},$$

$$A_1 = \begin{bmatrix} -2.2361 & -1.1358 & 1 & 0.6324 \\ -0.1024 & -3 & 0.3835 & 0.85 \\ 0.7112 & 11.2346 & -36.8199 & 4 \\ 1.0692 & 13.4230 & 20.1185 & -12.1801 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} -1.5326 & -1.2436 & 0.5458 & 0.7136 \\ -0.8 & -2.9346 & 0.0920 & 0.42 \\ 11.1634 & 23 & -26.4655 & -1.8347 \\ 25 & 8.3132 & -3.8714 & -31.4631 \end{bmatrix},$$

$$B_1 = B_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad Q_1 = Q_2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

In the simulation, we set $R_1 = R_2 = 1$, the error value ε as 10^{-15} and the initial stabilizing feedback gain as $K_{i(0)} = 0$. According to the OPPIA in Algorithm 1, AREs

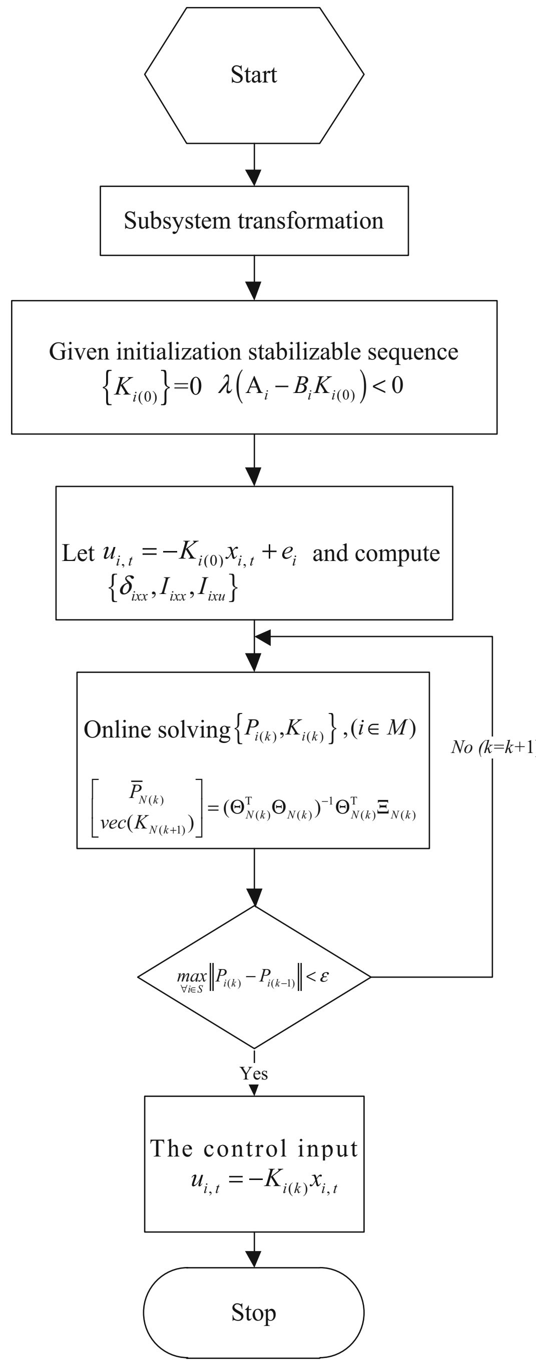

Fig. 1 The flowchart of Algorithm 2

(6) can be directly solved. After the 14th iterations, the exactly solutions are obtained:

$$P_1^* = \begin{bmatrix} 0.3505 & -0.0320 & 0.0180 & 0.0191 \\ -0.0320 & 0.5943 & 0.0572 & 0.0845 \\ 0.0180 & 0.0572 & 0.0268 & 0.0232 \\ 0.0191 & 0.0845 & 0.0232 & 0.0503 \end{bmatrix},$$

$$P_2^* = \begin{bmatrix} 0.5574 & -0.0483 & 0.0140 & 0.0227 \\ -0.0483 & 0.4057 & 0.0164 & 0.0111 \\ 0.0140 & 0.0164 & 0.0196 & -0.0003 \\ 0.0227 & 0.0111 & -0.0003 & 0.0178 \end{bmatrix}.$$

The OPPIA in Algorithm 1 is offline and the subsystems dynamics matrices are all needed. To show the effectiveness and applicability of Algorithm 2, we use the same conditions mentioned above. For the two subsystems, we select the initial states as $x_{10} = x_{20} = [0 \quad -0.1 \quad 0.10]^{\mathrm{T}}$ and the controller $u_{i(k)} = -K_{i(k)}x_i + e_i$ in the time interval $[0,\ 2]$ with the exploration noises as $e_1 = e_2 = 100\sum_{n=1}^{100}\sin(\omega_n t)$, where ω_n is a number and can be randomly selected in $[-500,\ 500]$. For the two subsystems, the information of the states and inputs is sampled every 0.01 s, and the optimal controller can be updated every 0.1 s. By the given iteration criterion, we can get the final calculation results of Algorithm 2 in step 4, after 7th iterations as:

$$P_{1(7)} = \begin{bmatrix} 0.3505 & -0.0320 & 0.0180 & 0.0191 \\ -0.0320 & 0.5943 & 0.0572 & 0.0845 \\ 0.0180 & 0.0572 & 0.0268 & 0.0232 \\ 0.0191 & 0.0845 & 0.0232 & 0.0503 \end{bmatrix},$$

$$P_{2(7)} = \begin{bmatrix} 0.5574 & -0.0483 & 0.0140 & 0.0227 \\ -0.0483 & 0.4057 & 0.0164 & 0.0111 \\ 0.0140 & 0.0164 & 0.0196 & -0.0003 \\ 0.0227 & 0.0111 & -0.0003 & 0.0178 \end{bmatrix}.$$

The optimal state feedback controller gains are:

$$K_{1(7)} = [0.3557 \quad 0.7040 \quad 0.1253 \quad 0.1771],$$

$$K_{2(7)} = [0.5458 \quad 0.3849 \quad 0.0497 \quad 0.0513].$$

Then, we get the simulation results by using MATLAB. Figures 2, 3, 4 and 5 show the convergence of Algorithm 2. The convergence of P_1 and P_2 to the optimal values are shown in Figs. 2, 3. The elements of matrices P_1 and P_2 are updated which are shown in Figs. 4, 5.

Example 2 In order to further study the proposed approach, we give the following comparative results. Applying the parallel RL algorithm proposed in this paper and simulating the experiment with the same parameters as [45], we can get the final solutions after 7 iterations:

$$P_{1(7)} = \begin{bmatrix} 0.2408 & 0.0705 & 0.0393 & 0.0182 \\ 0.0705 & 0.0308 & 0.0085 & 0.0064 \\ 0.0393 & 0.0085 & 0.0157 & 0.0025 \\ 0.0182 & 0.0064 & 0.0025 & 0.0016 \end{bmatrix},$$

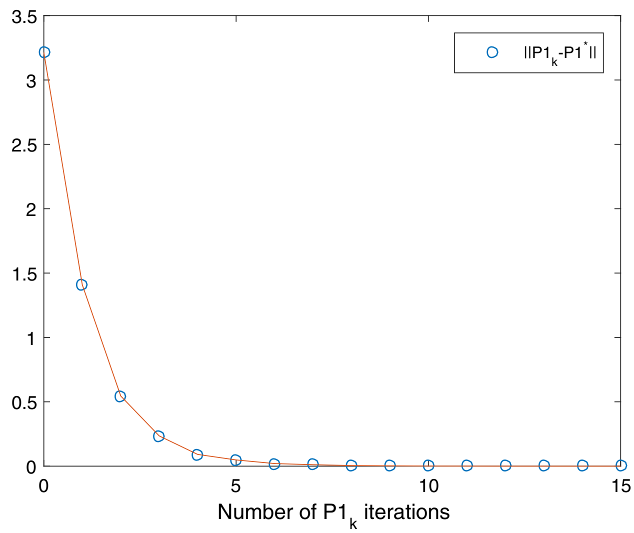

Fig. 2 The convergence of $P_{1(k)}$ to the optimal value P_1^*

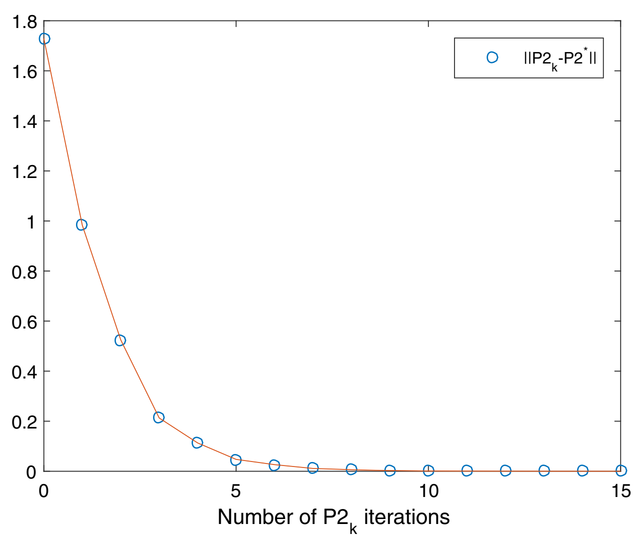

Fig. 3 The convergence of $P_{2(k)}$ to the optimal value P_2^*

$$P_{2(7)} = \begin{bmatrix} 0.5026 & 0.1343 & 0.0518 & 0.0097 \\ 0.1343 & 0.0485 & 0.0138 & 0.0026 \\ 0.0518 & 0.0138 & 0.0193 & 0.0002 \\ 0.0097 & 0.0026 & 0.0002 & 0.0003 \end{bmatrix}.$$

Comparing with the online algorithms proposed in [45], the simulation errors between the calculated results proposed in this paper can nearly converge to zero. It shows that the parallel RL algorithm in this paper gives a more accurate solution. It is also seen from Figs. 6 and 7 that the convergence rate of the parallel RL algorithm is also faster than the results in [45].

Remark 6 In the two simulation examples, we solve the positive-definite solution $P_{i(k)}$ by the online reinforcement

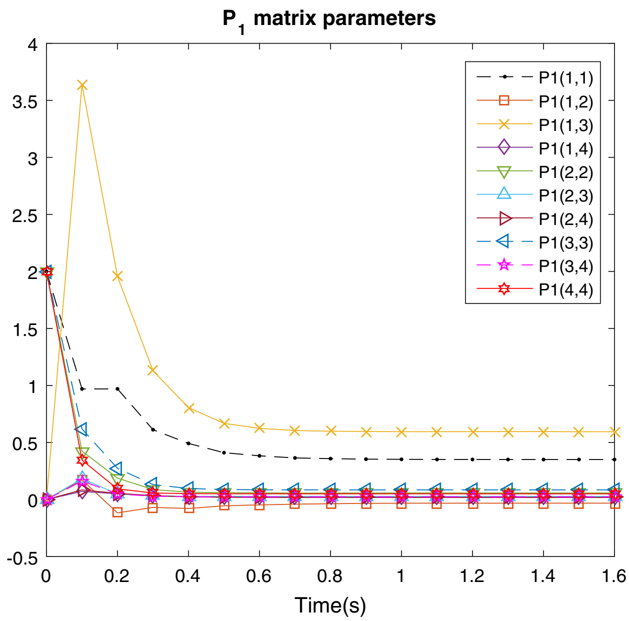

Fig. 4 The updated elements of $P_{1(k)}$ at each step in Algorithm 2

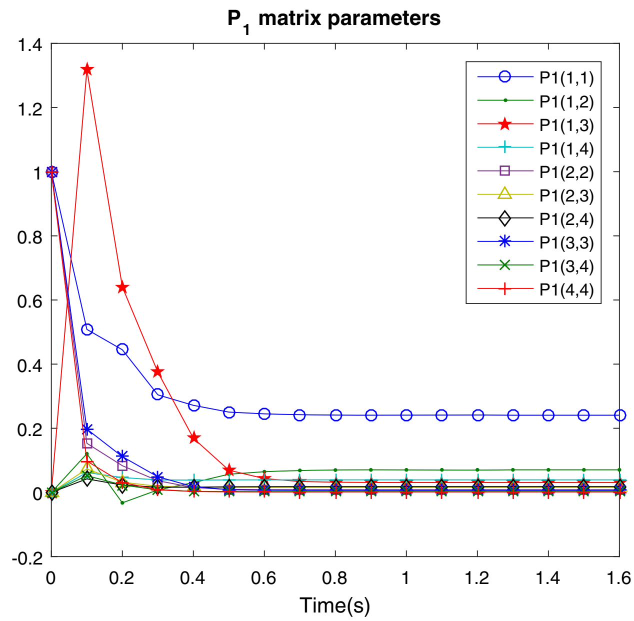

Fig. 6 The updated elements of $P_{1(k)}$ using the parallel RL algorithm

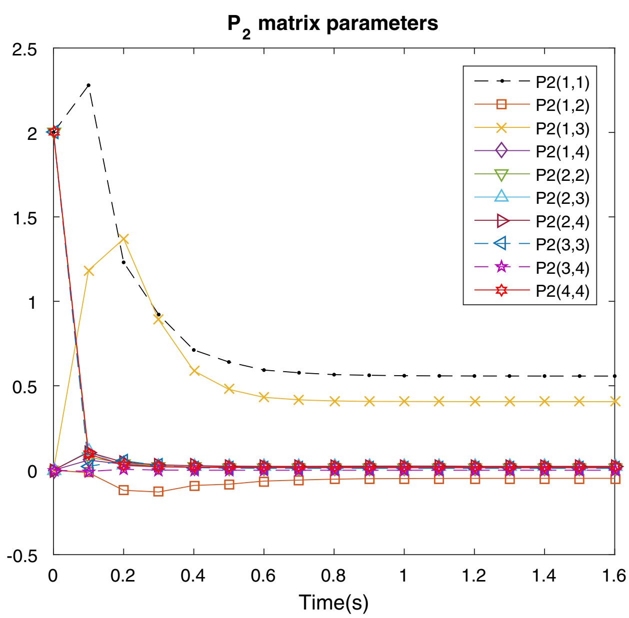

Fig. 5 The updated elements of $P_{2(k)}$ at each step in Algorithm 2

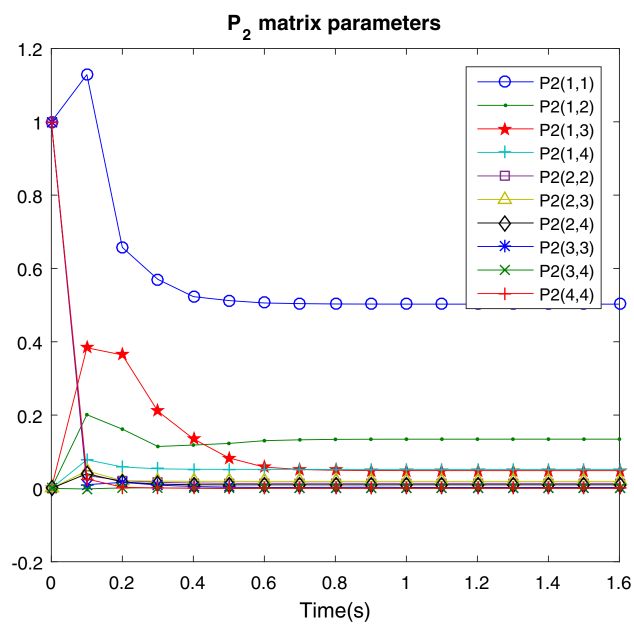

Fig. 7 The updated elements of $P_{2(k)}$ using the parallel RL algorithm

learning method. First, we obtain N coupled Riccati equations by decomposing the subsystems. According to the online implementation process, we need to collect the states and the inputs of each subsystem, and satisfy the condition of the full rank of the least square equation. Then, we iteratively solve the least square formula, until the final optimal solution satisfying the limited precision requirement is obtained.

5 Conclusion

A novel online parallel RL algorithm is proposed with completely unknown dynamics. It can solve the AREs by collecting the information of the states and the inputs of each of the subsystems. Therefore, this novel parallel RL algorithm can parallelly compute the corresponding N coupled AREs with completely unknown dynamics. The convergence of the parallel RL algorithm is also proved. Two simulation results illustrate the applicability and effectiveness of the designed algorithm. In the future, the RL algorithm and the relevant optimal algorithm can be used in other aspects, such as artificial intelligence, industrial informatics, pattern processing and wireless automation devices.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grant 61673001, 61722306, the Foundation for Distinguished Young Scholars of Anhui Province under Grant 1608085J05, the Key Support Program of University Outstanding Youth Talent of Anhui Province under Grant gxydZD2017001, the State Key Program of National Natural Science Foundation of China under Grant 61833007 and the 111 Project under Grant B12018.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflicts of interest.

References

1. Krasovskii NN, Lidskii EA (1961) Analysis design of controller in systems with random attributes—part 1. Autom Remote Control 22:1021–1025
2. Luan X, Huang B, Liu F (2018) Higher order moment stability region for Markov jump systems based on cumulant generating function. Automatica 93:389–396
3. Zhang L, Boukas EK (2009) Stability and stabilization of Markovian jump linear systems with partly unknown transition probabilities. Automatica 45(2):463–468
4. Shi P, Li F (2015) A survey on Markovian jump systems: modeling and design. Int J Control Autom Syst 13(1):1–16
5. Wang Y, Xia Y, Shen H, Zhou P (2017) SMC design for robust stabilization of nonlinear Markovian jump singular systems. IEEE Trans Autom Control. https://doi.org/10.1109/tac.2017.2720970 **(In Press)**
6. Li H, Shi P, Yao D, Wu L (2016) Observer-based adaptive sliding mode control for nonlinear Markovian jump systems. Automatica 64(1):133–142
7. Kao Y, Xie J, Wang C, Karimi HR (2015) A sliding mode approach to H_∞ non-fragile observer-based control design for uncertain Markovian neutral-type stochastic systems. Automatica 52:218–226
8. Shi P, Liu M, Zhang L (2015) Fault-tolerant sliding mode observer synthesis of Markovian jump systems using quantized measurements. IEEE Trans Industr Electron 62(9):5910–5918
9. Ma Y, Jia X, Liu D (2016) Robust finite-time H_∞ control for discrete-time singular Markovian jump systems with time-varying delay and actuator saturation. Appl Comput Math 286:213–227
10. Mao Z, Jiang B, Shi P (2007) H_∞ fault detection filter design for networked control systems modelled by discrete Markovian jump systems. IET Control Theory Appl 1(5):1336–1343
11. Shi P, Li F, Wu L, Lim CC (2017) Neural network-based passive filtering for delayed neutral-type semi-markovian jump systems. IEEE Trans Neural Netw Learn Syst 28(9):2101–2114
12. Li F, Wu L, Shi P, Lim CC (2015) State estimation and sliding mode control for semi-Markovian jump systems with mismatched uncertainties. Automatica 51:385–393
13. Ma H, Liang H, Zhu Q, Ahn CK (2018) Adaptive dynamic surface control design for uncertain nonlinear strict-feedback systems with unknown control direction and disturbances. IEEE Trans Syst Man Cybern Syst. https://doi.org/10.1109/tsmc.2018.2855170 **(In Press)**
14. Ma H, Zhou Q, Bai L, Liang H (2018) Observer-based adaptive fuzzy fault-tolerant control for stochastic nonstrict-feedback nonlinear systems with input quantization. IEEE Trans Syst Man Cybern Syst. https://doi.org/10.1109/tsmc.2018.2833872 **(In Press)**
15. Tao G (2003) Adaptive control design and analysis. Wiley-IEEE Press, Hoboken
16. Kleinman D (1968) On an iterative technique for Riccati equation computations. IEEE Trans Autom Control 13(1):114–115
17. Lu L, Lin W (1993) An iterative algorithm for the solution of the discrete-time algebraic Riccati equation. Linear Algebra Appl 188–189(1):465–488
18. Costa OLV, Aya JCC (1999) Temporal difference methods for the maximal solution of discrete-time coupled algebraic Riccati equations. In: Proceedings of the american control conference, San Diego. IEEE Press, pp 1791–1795
19. Gajic Z, Borno I (1975) Lyapunov iterations for optimal control of jump linear systems at steady state. IEEE Trans Autom Control 40(11):1971–1975
20. He W, Dong Y, Sun C (2016) Adaptive neural impedance control of a robotic manipulator with input saturation. IEEE Trans Syst Man Cybern Syst 46(3):334–344
21. Shen H, Men Y, Wu Z, Park JH (2017) Nonfragile H_∞ control for fuzzy Markovian jump systems under fast sampling singular perturbation. IEEE Trans Syst Man Cybern Syst. https://doi.org/10.1109/tsmc.2017.2758381 **(In Press)**
22. Xu Y, Lu R, Peng H, Xie K, Xue A (2017) Asynchronous dissipative state estimation for stochastic complex networks with quantized jumping coupling and uncertain measurements. IEEE Trans Neural Netw Learn Syst 28(2):268–277
23. Cheng J, Park JH, Karimi HR (2018) A flexible terminal approach to sampled-data exponentially synchronization of Markovian neural networks with time-varying delayed signals. IEEE Trans Cybern 48(8):2232–2244
24. Zhai D, An L, Li X, Zhang Q (2018) Adaptive fault-tolerant control for nonlinear systems with multiple sensor faults and unknown control directions. IEEE Trans Neural Netw Learn Syst 29(9):4436–4446
25. Zhai D, An L, Ye D, Zhang Q (2018) Adaptive reliable H_∞ static output feedback control against Markovian jumping sensor failures. IEEE Trans Neural Netw Learn Syst 29(3):631–644
26. Liu D, Wei Q, Yan P (2015) Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems. IEEE Trans Syst Man Cybern Syst 45(12):1577–1591
27. Wei Q, Liu D (2014) Stable iterative adaptive dynamic programming algorithm with approximation errors for discrete-time nonlinear systems. Neural Comput Appl 24(6):1355–1367
28. Liang Y, Zhang H, Xiao G, Jiang H (2018) Reinforcement learning-based online adaptive controller design for a class of unknown nonlinear discrete-time systems with time delays. Neural Comput Appl 30(6):1733–1745
29. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. IEEE Circuits Syst Mag 9(3):32–50
30. Vrabie D, Lewis FL (2011) Adaptive dynamic programming for online solution of a zero-sum differential game. IET Control Theory Appl 9(3):353–360
31. Vrabie D, Lewis FL (2009) Adaptive optimal control algorithm for continuous-time nonlinear systems based on policy iteration. In: Proceedings of the 48th IEEE conference on decision and control, Shanghai, pp 73–79
32. Guo W, Si J, Liu F, Mei S (2018) Policy approximation in policy iteration approximate dynamic programming for discrete-time

nonlinear systems. IEEE Trans Neural Netw Learn Syst 29(7):2794–2807
33. Kiumarsi B, Lewis FL, Modares H, Karimpour A, Naghibi-Sistani MB (2014) Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics. Automatica 50(4):1167–1175
34. Liu YJ, Li S, Tong CT, Chen CLP (2019) Adaptive reinforcement learning control based on neural approximation for nonlinear discrete-time systems with unknown nonaffine dead-zone input. IEEE Trans Neural Netw Learn Syst 30(1):295–305
35. Wu HN, Luo B (2013) Simultaneous policy update algorithms for learning the solution of linear continuous-time H_∞ state feedback control. Inf Sci 222(11):472–485
36. Mu C, Wang D, He H (2017) Data-driven finite-horizon approximate optimal control for discrete-time nonlinear systems using iterative HDP approach. IEEE Trans Cybern. https://doi.org/10.1109/tcyb.2017.2752845 **(In Press)**
37. He X, Huang T, Yu J, Li C, Zhang Y (2017) A continuous-time algorithm for distributed optimization based on multiagent networks. IEEE Trans Syst Man Cybern Syst. https://doi.org/10.1109/tsmc.2017.2780194 **(In Press)**
38. Yang X, He H, Liu Y (2017) Adaptive dynamic programming for robust neural control of unknown continuous-time nonlinear systems. IET Control Theory Appl 11(14):2307–2316
39. Xu W, Huang Z, Zuo L, He H (2017) Manifold-based reinforcement learning via locally linear reconstruction. IEEE Trans Neural Netw Learn Syst 28(4):934–947
40. Alipour MM, Razavi SN, Derakhshi MRF, Balafar MA (2018) A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. Neural Comput Appl 30(9):2935–2951
41. Zhu Y, Zhao D (2015) A data-based online reinforcement learning algorithm satisfying probably approximately correct principle. Neural Comput Appl 26(4):775–787
42. Tang L, Liu Y, Tong S (2014) Adaptive neural control using reinforcement learning for a class of robot manipulator. Neural Comput Appl 25(1):135–141
43. Mu C, Ni Z, Sun C, He H (2017) Air-breathing hypersonic vehicle tracking control based on adaptive dynamic programming. IEEE Trans Neural Netw Learn Syst 28(3):584–598
44. Xie X, Yue D, Hu S (2017) Fault estimation observer design of discrete-time nonlinear systems via a joint real-time scheduling law. IEEE Trans Syst Man Cybern Syst 45(7):1451–1463
45. He S, Song J, Ding Z, Liu F (2015) Online adaptive optimal control for continuous-time Markov jump linear systems using a novel policy iteration algorithm. IET Control Theory Appl 9(10):1536–1543
46. Song J, He S, Liu F, Niu Y, Ding Z (2016) Data-driven policy iteration algorithm for optimal control of continuous-time Itô stochastic systems with Markovian jumps. IET Control Theory Appl 10(12):1431–1439
47. Song J, He S, Ding Z, Liu F (2016) A new iterative algorithm for solving H_∞ control problem of continuous-time Markovian jumping linear systems based on online implementation. Int J Robust Nonlinear Control 26(17):3737–3754
48. Gajic Z, Borno I (2000) General transformation for block diagonalization of weakly coupled linear systems composed of N-subsystems. IEEE Trans Circuits Syst I Fundam Theory Appl 47(6):909–912