Neuro-Optimal Control for Discrete Stochastic Processes via a Novel Policy Iteration Algorithm

Mingming Liang, Ding Wang, *Member, IEEE*, and Derong Liu, *Fellow, IEEE*

***Abstract*—In this paper, a novel policy iteration adaptive dynamic programming (ADP) algorithm is presented which is called "local policy iteration ADP algorithm" to obtain the optimal control for discrete stochastic processes. In the proposed local policy iteration ADP algorithm, the iterative decision rules are updated in a local space of the whole state space. Hence, we can significantly reduce the computational burden for the CPU in comparison with the conventional policy iteration algorithm. By analyzing the convergence properties of the proposed algorithm, it is shown that the iterative value functions are monotonically nonincreasing. Besides, the iterative value functions can converge to the optimum in a local policy space. In addition, this local policy space will be described in detail for the first time. Under a few weak constraints, it is also shown that the iterative value function will converge to the optimal performance index function of the global policy space. Finally, a simulation example is presented to validate the effectiveness of the developed method.**

***Index Terms*—Adaptive critic designs, adaptive dynamic programming (ADP), local policy iteration, neuro-dynamic programming, optimal control, stochastic processes.**

I. Introduction

MAIN control techniques include robust control [1]–[3], adaptive control [4], [5], intelligent control [6], [7], optimal control [8], [9] and stochastic control [10]. Wang *et al.* [1]–[3] showed the sliding mode technique has a strong robustness for uncertain parts of the dynamic systems. Adaptive dynamic programming (ADP) presented in Werbos' papers for the very first time [4], [5], has shown great effectiveness and feasibility in obtaining the optimal control policy for nonlinear systems [6]–[9] and stochastic processes [10]. The biggest breakthrough of this algorithm lies in that it uses neural networks to represent the iterative value function and the iterative control law. Hence, ADP can successfully avoid the so called "curse of dimensionality." Meanwhile, this breakthrough also enable the algorithm to update the value function and decision rule forward-in-time. Because of the superiority of ADP in obtaining the optimal decision rule for dynamic systems, many researchers have paid close attention to related research of this algorithm [11], [12]. In [13], depending on its implementation method, ADP could be categorized into six classes: 1) heuristic dynamic programming (HDP); 2) action-dependent HDP; 3) dual HDP (DHP) [14]–[17]; 4) action-dependent DHP; 5) globalized DHP (GDHP); and 6) ADGDHP. Iterative method is a powerful route to analyze the convergence and other properties of ADP. Policy iteration and value iteration are two frequently used methods in the ADP algorithms [18].

Bertsekas and Tsitsiklis [10] for the first time investigated the ADP algorithm with value iteration method which was applied to discrete-time dynamic systems. In [19], it was shown that if the value iteration algorithm was initialized by a zero value function, then the obtained decision rule sequence and value function sequence would converge to the optimal decision rule and the optimal performance index function, respectively. In [20], the condition that the value iteration algorithm must start with a zero value function was successfully released. The authors provided an effective method to prove that the obtained decision rule sequence and value function sequence would converge to the optimal decision rule and the optimal performance index function, respectively, no matter how the initial value function was chosen. Bertsekas and Tsitsiklis [10] for the first time investigated the ADP algorithm with policy iteration method which was applied to discrete-time nonlinear systems. Murray *et al.* [22] applied the ADP algorithm with policy iteration method to the continuous-time systems. Wei and Liu [23] proposed a novel policy iteration algorithm which was called "θ-ADP." The restriction that the policy iteration algorithm must start with an admissible decision rule was successfully released. It is also shown that each iterative decision rule obtained by the proposed θ-ADP algorithm could be guaranteed to stabilize the nonlinear system. Liu *et al.* [24] established the error bounds for the ADP algorithm with policy iteration method. And the sequence of iterative decision rules obtained by the algorithm would finally converge to within a finite neighborhood of the optimal value function under a few weak constraints. In [25],

Manuscript received August 21, 2018; revised January 15, 2019; accepted March 14, 2019. Date of publication April 12, 2019; date of current version October 15, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61773373, Grant U1501251, and Grant 61533017, in part by the Young Elite Scientists Sponsorship Program by the China Association for Science and Technology, and in part by the Youth Innovation Promotion Association of the Chinese Academy of Sciences. This paper was recommended by Associate Editor H. R. Karimi. *(Corresponding author: Ding Wang.)*

M. Liang is with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: liangmingming2015@ia.ac.cn).

D. Wang is with the Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China, and also with the Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing University of Technology, Beijing 100124, China (e-mail: dingwang@bjut.edu.cn).

D. Liu is with the School of Automation, Guangdong University of Technology, Guangzhou 510006, China (e-mail: derongliu@foxmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2019.2907991

an effective off-policy learning-based policy iteration algorithm was proposed to solve the optimal control problem for completely unknown continuous-time systems with unknown disturbances. In [26], a generalized policy iteration algorithm was presented, which adopted the technical advantages of both the policy iteration and value iteration algorithms. It is shown that the proposed algorithm is also able to obtain the optimal decision rule and the optimal performance index function for the dynamic system. Meanwhile, the generalized framework is of great significance for the development of ADP theory. Zhang *et al.* [29] made great achievements in solving the coupled Hamilton–Jacobi equations via constructing a novel relax adaptive matrix for the nonzero-sum games. Meanwhile, the strict restriction of initially stabilized control policies had been successfully removed by proposing a new switch operator. These results provide an important theoretical foundation for the optimal control and differential games of nonlinear systems.

Bertsekas [30] for the first time proved that the proper policy iteration method could not only be applied to solve the shortest path problems (SSPs) with finite state space (which can be viewed as Markov decision processes), but also could be applied to solve SSP with infinite state space. For Markov decision processes, Bertsekas presented the method of "approximate policy iteration" (policy iteration ADP algorithm) to obtain the optimal performance index function for discrete stochastic processes with large state space using neural-structure to approximate the corresponding control law or value function in [34]. Bertsekas [35] discussed several issues related to approximate policy iteration method (policy iteration ADP algorithm) such as convergence and convergence rate for the sequence of polices. Bertsekas [36] presented the implementations of the method of combining the λ-policy iteration algorithm and the approximate policy iteration algorithm to review the issues such as the bias issue and the exploration issue. Approximate policy iteration method is also applied to many actual Markov decision processes such as missile defense and interceptor allocation problems [38] and retailer inventory management problems [39].

However, to obtain the optimal decision rule and the optimal performance index function for dynamic systems, the policy iteration algorithms mentioned above require that the iterative decision rule and the iterative value function should be renewed for every state in the global system state space in each iteration epoch [10], [12], [22]–[24], [26]. We call these policy iteration algorithms the conventional global policy iteration algorithms. However, in practical projects, it is very difficult to meet these requirements for the conventional policy iteration algorithms. First, for many practical applications, the existence of noise makes it very hard to implement the policy iteration algorithms mentioned above. Second, it is usually difficult to gather the state data for the global system state set in each iteration epoch, since the practical dynamic system is generally operating in a local space of the global system state space. To implement the conventional global policy iteration algorithms, the computer must stop to wait until the program has explored all the data of the global state set and all the information needed has been loaded in RAM. That is to say, the CPU in the computer has a great chance of being in idle state. Hence, the time-efficiency of the conventional global policy iteration algorithm will be very low. Third, in numerical implementation of the ADP algorithm, the computer usually computes the refreshed decision rule and value function in a parallel manner. When we want to refresh the iterative decision rule and the iterative value function for every state in the global state set, the computer will calculate all the values for the whole state set simultaneously. Hence, if the global state set is too large, the computer will have a great chance of being overloaded. That is to say, the computational load of the conventional policy iteration algorithm is too excessive for the computer. It is required to propose some methods to overcome these shortcomings of the conventional policy iteration algorithm.

In this paper, we present a novel policy iteration ADP algorithm which is called "local policy iteration ADP algorithm" to solve optimal control problems for the stochastic processes. Here, we summarize our main contributions and novelties of our paper as follows.

1) For the first time, we apply the local policy iteration ADP algorithm to stochastic processes in order to obtain the optimal performance index function.
2) For the first time, we combine the local policy iteration method with neural networks to overcome the problems arisen in stochastic models with large state space such as the curse of dimensionality and the heavy burden in computation.
3) We show that the sequence of value functions generated by our algorithm can finally reach an optimum in a local policy space. Furthermore, this local policy space is described theoretically in detail for the first time.
4) Our proposed algorithm is able to reach the optimum in the global policy space under some weak conditions which is validated in our simulations.
5) Our proposed algorithm can significantly reduce the burden in computation which is validated in simulations.

This paper is organized as follows. In Section II, we present the system dynamic characteristics and some assumptions. The proposed local policy iteration ADP algorithm for stochastic processes is derived. In Section III, the monotonicity and the convergence properties of the iterative value function is developed. In Section IV, we illustrate how our algorithm utilize the neural networks to approximate the corresponding value functions and decision rules. In Section V, a simulation example is presented to validate the effectiveness of the developed method. Finally, in Section VI, we will make a simple summary about points of this paper.

II. Problem Formulation

A. System Description

In this paper, we will focus on the following discrete-time dynamic system:

$$x(k+1) = F(x(k), a(k), \omega(k), k), \quad k = 0, 1, 2, 3, \ldots \tag{1}$$

where $x(k) \in \mathbf{R}^n$ is the system state, $a(k)$ is the action made by the decision maker observing the system state $x(k)$, and

$\omega(k)$ is the environment disturbance which is independent of $\omega(\tau)$ for all $\tau < k$.

Here, we denote the set of possible system states by X and the allowable actions in state x by A_x. Given the current system state $x(k)$ and the current action $a(k)$, according to [27], the next system state $x(k+1)$ is determined by a probability distribution $p(\cdot \mid x(k), a(k))$.

We denote decision rule as function $d(x) : X \to A_x$, which specify the action choice a when the system occupies state x. Let policy $\pi = (d(x,0), d(x,1), d(x,2), \ldots, d(x,k), \ldots)$ be an arbitrary sequence of decision rules where $d(x,k)$ denotes the decision rule at time k. The expected total discounted reward for state x_0 under the policy π is defined as

$$J^{\pi}(x_0) = E\left\{\sum_{k=0}^{\infty} \lambda^k U(x(k), d(x,k))\right\} \tag{2}$$

where $U(x,a) = U(x(k), d(x,k))$ is the utility function, λ is a discount factor, and the mathematical symbol $E\{\cdot\}$ represents the expectation of the random variable inside the bracket.

The goal of the presented algorithm is to find an optimal policy to minimize the performance index function (2). For convenience of analysis, results of this paper are based on the following assumption.

Assumption 1 (See [27]): The function $F(x,a,\omega,k)$ is Lipschitz continuous on its dominate. The system state set X and the action set A_x are discrete (finite or countably infinite). Function $U(x,a)$ and $p(\cdot \mid x,a)$ do not vary with time. The utility function $U(x,a)$ satisfies $|U(x,a)| \leq M < \infty$ for all $a \in A_x$ and $x \in X$. The discount factor λ satisfies $0 < \lambda < 1$.

Remark 1: There are many practical cases which meet our assumptions.

1) Wireless Communication Systems [31]: Zhou *et al.* [31] proposed an optimal dynamic multicast scheduling to minimize the delay, power, and fetching costs for cache-enabled content-centric wireless networks. Here, the random process $\{\mathbf{Q}(k)\}$ is a Markov decision process, the system state is the request queues sent from users to base stations. The state space is denoted by $\mathcal{Q} \triangleq \prod_{m \in \mathcal{M}} Q_m$, where $Q_m \triangleq \{0, 1, \ldots, N_m\}$. Hence, the system state space is countable. And the immediate cost is the weighted sum of current delay cost, fetching cost, and current power cost which is obviously finite according to [31].

2) Thermal Management for High Performance Processors [32]: Jung and Pedram [32] presented a dynamic thermal management technique to minimize the power dissipation and on-chip temperature for processors. Here, the on-chip temperature change is a Markov decision process. The system state is the current CPU temperature at time k defined to be one of three states: s_1, s_2, or s_3 as shown in [32] which is obviously countable. The immediate cost is the sum of current power dissipation and current on-chip temperature of the CPU which is obviously finite according to [32].

3) Energy Control for Sustainable Manufacturing Systems [33]: In [33], the complex interaction between the adopted energy control decisions and system state evolutions is actually a Markov decision process. The energy states and the operation states are obviously countable according to [33]. The immediate cost is the energy consumption incurred before the system reaching the next decision epoch which is finite according to [33]. Other applications, such as queuing control problems, inventory problems, and SSPs [27], are all in line with our assumptions. Hence, our assumptions here are made based on actual applications and reasonable.

Define the set of decision rules at time k as D_k, and let Ξ denote the set of all policies from time 0 to infinity, i.e., $\Xi = D_0 \times D_1 \times D_2 \times D_3 \ldots$ Then, we can express the optimal performance index function as

$$J^*(x) = \inf_{\pi \in \Xi} \left\{J^{\pi}(x)\right\}. \tag{3}$$

To obtain this optimal index function, a new policy iteration algorithm will be developed.

B. Derivation of the Local Policy Iteration ADP Algorithm

In this section, we develop a local iterative algorithm to obtain the optimal policy and the optimal index function for the stochastic processes. For the proposed algorithm, we refresh the iterative decision rule in a local space of the whole system state set in each iteration epoch. We denote the sequence of the local spaces of the whole system state set as $\{B_x^i\}$, $i = 0, 1, 2, 3 \ldots$ For any local space of the whole system state, we have $B_x^i \subseteq X \ \forall i$.

For all $x \in X$, let $\widetilde{\pi}_0 \equiv (d_0(x), d_0(x), d_0(x), \ldots)$ denote the initial stationary policy which uses the same decision rule $d_0(x) \in D = D_0 = D_1 = D_2 \ldots$ at each time k. For all $x \in X$, let $V_0(x)$ be the initial iterative value function that satisfies the following equation:

$$V_0(x) = U(x, d_0(x)) + \sum_{j \in X} \lambda p(j \mid x, d_0(x)) V_0(j) \tag{4}$$

where $p(\cdot \mid x, d_0(x))$ denotes the transition probability under decision rule $d_0(x)$.

Then, for all $x \in B_x^0$, the iterative decision rule $d_1(x)$ which forms the stationary policy $\widetilde{\pi}_1 \equiv (d_1(x), d_1(x), d_1(x), \ldots)$ is computed as

$$d_1(x) = \arg\min_{a \in A_x} \left\{U(x,a) + \sum_{j \in X} \lambda p(j \mid x, a) V_0(j)\right\}. \tag{5}$$

Let $d_1(x) = d_0(x)$, for all $x \in X \backslash B_x^0$. For all $i = 1, 2, 3, \ldots$, let $V_i(x)$ be the iterative value function that satisfies the following equation:

$$V_i(x) = U(x, d_i(x)) + \sum_{j \in X} \lambda p(j \mid x, d_i(x)) V_i(j). \tag{6}$$

For all $x \in B_x^i$, the iterative decision rule $d_{i+1}(x)$ which forms the stationary policy $\widetilde{\pi}_{i+1} \equiv (d_{i+1}(x), d_{i+1}(x).d_{i+1}(x), \ldots)$ is computed as

$$d_{i+1}(x) = \arg\min_{a \in A_x} \left\{U(x,a) + \sum_{j \in X} \lambda p(j \mid x, a) V_i(j)\right\}. \tag{7}$$

Let $d_{i+1}(x) = d_i(x)$, for all $x \in X \backslash B_x^i$.

III. Local Policy Iteration ADP Algorithm

A. Properties of the Local Policy Iteration ADP Algorithm

In this section, we will show that the sequence of the iterative value functions obtained by the proposed algorithm have a very good property in terms of monotonicity. And the sequence of the value functions obtained by the proposed algorithm will finally converge to the optimal performance index function in a local policy space or in the global policy space depending on different conditions.

In the remainder of this paper, we will let $|X|$ denote the number of elements in X, $\mathbf{U}_{d(x)}$ denote the $|X|$-dimensional vector with component $U(x, d(x))$ for each $x \in X$, $\mathbf{V}$ denote the $|X|$-dimensional vector with xth component $V(x)$, and $\mathbf{P}_{d(x)}$ denote the $|X| \times |X|$ matrix with (x, j)th entry given by $p(j \mid x, d(x))$.

Let $\mathcal{V}$ denote the set of bounded real-valued functions on X, for each $V(x) \in \mathcal{V}$, define the norm of $V(x)$ by

$$\|V(x)\| = \sup_{x \in X} |V(x)|. \tag{8}$$

For discrete set X, we refer to elements of $\mathcal{V}$ as vectors. Define the norm of vector $\mathbf{U}_{d(x)}$, denoted by $\|\mathbf{U}_{d(x)}\|$ as

$$\left\|\mathbf{U}_{d(x)}\right\| = \sup_{x \in X} |U(x, d(x))|. \tag{9}$$

Define the norm of vector $\mathbf{V}$, denoted by $\|\mathbf{V}\|$ as

$$\|\mathbf{V}\| = \sup_{x \in X} |V(x)|. \tag{10}$$

According to [28], $\mathbf{P}_{d(x)}$ is a bounded linear transformation on $\mathcal{V}$. Define the norm of $\mathbf{P}_{d(x)}$, denoted by $\|\mathbf{P}_{d(x)}\|$ as

$$\left\|\mathbf{P}_{d(x)}\right\| = \sup\left\{\left\|\mathbf{P}_{d(x)}\mathbf{V}\right\|, \|\mathbf{V}\| \le 1, \mathbf{V} \in \mathcal{V}\right\}. \tag{11}$$

When X is discrete, so that $\mathbf{P}_{d(x)}$ is a matrix with components $p(j \mid x, d(x))$, this definition implies that

$$\left\|\mathbf{P}_{d(x)}\right\| = \sup_{x \in X} \sum_{j \in X} p(j \mid x, d(x)). \tag{12}$$

When $\mathbf{P}_{d(x)}$ is a probability matrix, $\|\mathbf{P}_{d(x)}\| = 1$.

Now, let us present the following lemma.

Lemma 1: Suppose $0 \le \lambda < 1$, and let the stationary policy be $\widetilde{\pi} = (d(x), d(x), d(x), \ldots)$, then there exists a real-valued and bounded function $V(x)$ that satisfies the following equation:

$$V(x) = U(x, d(x)) + \sum_{j \in X} \lambda p(j \mid x, d(x)) V(j) \tag{13}$$

or in vector notation

$$\mathbf{V} = \mathbf{U}_{d(x)} + \lambda \mathbf{P}_{d(x)} \mathbf{V}. \tag{14}$$

Proof: According to the definition of the value function of stationary policy $\widetilde{\pi}$, we have

$$\mathbf{V}^{\widetilde{\pi}} = \sum_{k=0}^{\infty} \lambda^k \mathbf{P}^k_{d(x)} \mathbf{U}_{d(k)} \tag{15}$$

where $\mathbf{P}^0_{d(x)} = \mathbf{I}$.

Then, we can derive

$$\begin{aligned} \left\|\mathbf{P}^k_{d(x)}\right\| &= \left\|\mathbf{P}^k_{d(x)}\right\| \\ &\le \left\|\mathbf{P}_{d(x)}\right\|^k \\ &= 1. \end{aligned} \tag{16}$$

Hence

$$\left\|\mathbf{V}^{\widetilde{\pi}}\right\| = \left\|\sum_{k=0}^{\infty} \lambda^k \mathbf{P}^k_{d(x)} \mathbf{U}_{d(x)}\right\| \tag{17}$$

$$\le \sum_{k=0}^{\infty} \left\|\lambda^k \mathbf{P}^k_{d(x)} \mathbf{U}_{d(x)}\right\| \tag{18}$$

$$\le \sum_{k=0}^{\infty} \left|\lambda^k\right| \left\|\mathbf{P}^k_{d(x)}\right\| \left\|\mathbf{U}_{d(x)}\right\| \tag{19}$$

$$\le \sum_{k=0}^{\infty} \lambda^k M \tag{20}$$

$$= \frac{M}{1-\lambda} \tag{21}$$

$$< \infty \tag{22}$$

thus $\mathbf{V}^{\widetilde{\pi}}$ is a bounded real-valued function. Let $\mathbf{V} = \mathbf{V}^{\widetilde{\pi}}$, the proof is completed. ■

According to Lemma 1, for any stationary $\pi \in \Pi$, there exists a real-valued and bounded function $V_0(x)$ that satisfies (4). Thus, the iterative value function $V_0(x)$ can be defined. Next, we will shed some light on the monotonicity property of the sequence of the iterative value functions obtained by the proposed algorithm.

Theorem 1: For $i = 0, 1, 2, 3, \ldots$, let $V_i(x)$ and $d_i(x)$ be obtained by (4)–(7). If Assumption 1 holds, then for all $x \in X$ and any $i = 0, 1, 2, 3, \ldots$, we have the following conclusion:

$$\begin{cases} V_{i+1}(x) \le V_i(x) \\ \|\mathbf{V}_{i+1}\| < \infty. \end{cases} \tag{23}$$

Proof: Equation (23) will be proven by mathematical induction. First, we consider $i = 0$. Let $V_1^q(x)$, $q = 0, 1, 2, 3, \ldots$, be an iterative value function that satisfies

$$V_1^{q+1}(x) = U(x, d_1(x)) + \sum_{j \in X} \lambda p(j \mid x, d_1(x)) V_1^q(j) \tag{24}$$

where $V_1^0(x) = V_0(x)$ and the iterative decision rules $d_1(x)$ is obtained by (5). From Lemma 1, the iterative value function $V_0(x)$ is bounded and real-valued. Next, we will prove

$$V_1^{q+1}(x) \le V_0(x) \tag{25}$$

$$\left\|\mathbf{V}_1^{q+1}\right\| \le \frac{M}{1-\lambda} < \infty \tag{26}$$

holds for any $q = 0, 1, 2, 3, \ldots$

We use the mathematical induction. First, for $q = 0$ and for all $x \in B_x^0$, according to (5), we can get

$$\begin{aligned} V_1^1(x) &= U(x, d_1(x)) + \sum_{j \in X} \lambda p(j \mid x, d_1(x)) V_1^0(j) \\ &= \min_{a \in A_x} \left\{ U(x, a) + \sum_{j \in X} \lambda p(j \mid x, a) V_0(j) \right\} \\ &\le U(x, d_0(x)) + \sum_{j \in X} \lambda p(j \mid x, d_0(x)) V_0(j) \\ &= V_0(x). \end{aligned} \tag{27}$$

For $q = 0$ and for all $x \in X \backslash B_x^0$, we obtain

$$\begin{aligned} V_1^1(x) &= U(x, d_1(x)) + \sum_{j \in X} \lambda p(j \mid x, d_1(x)) V_1^0(j) \\ &= U(x, d_0(x)) + \sum_{j \in X} \lambda p(j \mid x, d_0(x)) V_0(j) \\ &= V_0(x). \end{aligned} \tag{28}$$

According to (27) and (28), for all $x \in X$, we derive that

$$V_1^1(x) \le V_0(x). \tag{29}$$

For $q = 0$ and for all $x \in X$, according to Lemma 1, we have the conclusion

$$\begin{aligned} \left| V_1^1(x) \right| &= \left| U(x, d_1(x)) + \sum_{j \in X} \lambda p(j|x, d_1(x)) V_1^0(j) \right| \\ &\le |U(x, d_1(x))| + \lambda \sum_{j \in X} |p(j \mid x, d_1(x)) V_0(j)| \\ &\le |U(x, d_1(x))| + \lambda \|\mathbf{V}_0\| \\ &\le M + \frac{\lambda M}{1 - \lambda} \\ &= \frac{M}{1 - \lambda}. \end{aligned} \tag{30}$$

Considering the supremum over x in the above expression, we can get

$$\left\| \mathbf{V}_1^1 \right\| \le \frac{M}{1 - \lambda}. \tag{31}$$

Assume (25) and (26) hold for $q = l - 1, l = 2, 3, 4, 5, \ldots$, that is

$$V_1^l(x) \le V_0(x) \tag{32}$$

$$\left\| \mathbf{V}_1^l \right\| \le \frac{M}{1 - \lambda}. \tag{33}$$

Then, for $q = l$ and $x \in B_x^0$, we obtain

$$\begin{aligned} V_1^{l+1}(x) &= U(x, d_1(x)) + \sum_{j \in X} \lambda p(j \mid x, d_1(x)) V_1^l(j) \\ &\le U(x, d_1(x)) + \sum_{j \in X} \lambda p(j \mid x, d_1(x)) V_0(j) \\ &= \min_{a \in A_x} \left\{ U(x, a) + \sum_{j \in X} \lambda p(j \mid x, a) V_0(j) \right\} \\ &\le U(x, d_0(x)) + \sum_{j \in X} \lambda p(j \mid x, d_0(x)) V_0(j) \\ &= V_0(x). \end{aligned} \tag{34}$$

For $x \in X \backslash B_x^0$, we can derive

$$\begin{aligned} V_1^{l+1}(x) &= U(x, d_1(x)) + \sum_{j \in X} \lambda p(j \mid x, d_1(x)) V_1^l(j) \\ &= U(x, d_0(x)) + \sum_{j \in X} \lambda p(j \mid x, d_0(x)) V_1^l(j) \\ &\le U(x, d_0(x)) + \sum_{j \in X} \lambda p(j \mid x, d_0(x)) V_0(j) \\ &= V_0(x). \end{aligned} \tag{35}$$

According to (34) and (35), for all $x \in X$, we have the conclusion

$$V_1^{l+1}(x) \le V_0(x). \tag{36}$$

For $q = l$ and for $x \in X$, according to (33), we have

$$\begin{aligned} \left| V_1^{l+1}(x) \right| &= \left| U(x, d_1(x)) + \sum_{j \in X} \lambda p(j \mid x, d_1(x)) V_1^l(j) \right| \\ &\le |U(x, d_1(x))| + \lambda \sum_{j \in X} \left| p(j \mid x, d_1(x)) V_1^l(j) \right| \\ &\le |U(x, d_1(x))| + \lambda \left\| \mathbf{V}_1^l \right\| \\ &\le M + \frac{\lambda M}{1 - \lambda} \\ &= \frac{M}{1 - \lambda}. \end{aligned} \tag{37}$$

Taking the supremum over x in the above expression, we can get

$$\left\| \mathbf{V}_1^{l+1} \right\| \le \frac{M}{1 - \lambda}. \tag{38}$$

Then, according to (29), (31), (36), and (38), we can get the conclusion that

$$V_1^{q+1}(x) \le V_0(x) \tag{39}$$

$$\left\| \mathbf{V}_1^{q+1} \right\| \le \frac{M}{1 - \lambda} \tag{40}$$

holds for any $q = 0, 1, 2, 3, \ldots$

Now, we define the operator $L_{d_1} : \mathcal{V} \to \mathcal{V}$ as

$$L_{d_1}(V(x)) = U(x, d_1(x)) + \sum_{j \in X} \lambda p(j \mid x, d_1(x)) V(j). \tag{41}$$

Setting $V_1^0(x) = V_0(x)$, as the operator L_{d_1} is a contraction mapping, the sequence $\{V_1^{q+1}(x)\}$ defined by

$$V_1^{q+1}(x) = L_{d_1}\left(V_1^q(x)\right) = L_{d_1}^{q+1}(V_0(x)) \tag{42}$$

converges to a unique $V_1^\infty(x)$ which satisfies

$$L_{d_1}\left(V_1^\infty(x)\right) = V_1^\infty(x). \tag{43}$$

That is

$$V_1^\infty(x) = U(x, d_1(x)) + \sum_{j \in X} \lambda p(j \mid x, d_1(x)) V_1^\infty(j). \tag{44}$$

According to the uniqueness of $V_1^\infty(x)$ and the definition of $V_1(x)$ in (6), we can obtain that

$$\lim_{q \to \infty} V_1^{q+1}(x) = V_1^\infty(x) = V_1(x). \tag{45}$$

According to (39), (40), and (44), we can obtain that

$$V_1(x) \le V_0(x) \tag{46}$$

$$\|\mathbf{V}_1\| \le \frac{M}{1 - \lambda}. \tag{47}$$

Assume the conclusion holds for $i = l-1, l = 2, 3, 4, 5, \ldots$, that is

$$V_l(x) \le V_{l-1}(x) \tag{48}$$

$$\|\mathbf{V}_l\| \le \frac{M}{1 - \lambda}. \tag{49}$$

For $i = l$, let $V_{l+1}^{q}(x), q = 0, 1, 2, 3, \ldots$ be an iterative value function that satisfies

$$V_{l+1}^{q+1}(x) = U(x, d_{l+1}(x)) + \sum_{j \in X} \lambda p(j \,|\, x, d_{l+1}(x)) V_{l+1}^{q}(j) \quad (50)$$

where $V_{l+1}^{0}(x) = V_l(x)$. According to the idea of the mathematical induction from (24)–(40), for all $x \in X$, we can obtain that

$$V_{l+1}^{q+1}(x) \leq V_l(x) \quad (51)$$

$$\left\| \mathbf{V}_{l+1}^{q+1} \right\| \leq \frac{M}{1-\lambda} \quad (52)$$

holds for any $q = 0, 1, 2, 3, \ldots$

Define the operator $L_{d_{l+1}} : \mathcal{V} \to \mathcal{V}$ as

$$L_{d_{l+1}}(V(x)) = U(x, d_{l+1}(x)) + \sum_{j \in X} \lambda p(j \,|\, x, d_{l+1}(x)) V(j). \quad (53)$$

Using the same technique in (41)–(47), we can obtain

$$V_{l+1}(x) \leq V_l(x) \quad (54)$$

$$\|\mathbf{V}_{l+1}\| \leq \frac{M}{1-\lambda}. \quad (55)$$

The mathematical induction is complete. ■

As for $i = 0, 1, 2, 3, \ldots$, the iterative decision rule is updated in B_x^i. For a state $x \in X$, it can be located in several state sets, i.e., $x \in \{B_x^{i_0} \cap B_x^{i_1} \cap \ldots\}, i_\eta \in \{0, 1, 2, 3, \ldots\}$ and $\eta = 0, 1, 2, 3, \ldots$

Let the set $\mathcal{T}(x)$ of i_η be expressed as

$$\mathcal{T}(x) = \Big\{ i_\eta \,|\, x \in B_x^{i_\eta}, i_\eta \in \{0, 1, 2, 3, \ldots\} \quad \eta = 0, 1, 2, 3, \ldots, i_0 \leq i_1 \leq \cdots \leq i_\eta \leq \cdots \Big\}. \quad (56)$$

Let $\phi(x)$ denotes the number of the elements in $\mathcal{T}(x)$, which is expressed as

$$\phi(x) = |\mathcal{T}(x)|. \quad (57)$$

Let G denotes the subset of X, which is expressed as

$$G = \{x \,|\, x \in X, \phi(x) < \infty\}. \quad (58)$$

Then, we can derive the following theorem.

Theorem 2 (Local Optimality Property): For $i = 0, 1, 2, 3, \ldots$, let $V_i(x)$ and $d_{i+1}(x)$ be obtained by (4)–(7), then we can get the conclusion that

$$\lim_{\eta \to \infty} V_{i_\eta}(x) = \min_{\pi \in \Theta} V^{\pi}(x) = \inf_{\pi \in \Theta} V^{\pi}(x) \quad (59)$$

where Θ is a subspace of the policy space Ξ which will be explained later in the following proof.

Proof: Based on Theorem 1, we have the conclusion that the bounded sequence of the iterative value functions obtained by the proposed algorithm are monotonically decreasing and will finally reach convergence. Now, we choose any $\widetilde{x} \in X \backslash G$, then $\phi(\widetilde{x})$ in (57) satisfies $\phi(\widetilde{x}) \to \infty$, and we can also get

$$\widetilde{x} \in \cap_{\eta=0}^{\infty} B_x^{i_\eta}, i_\eta \in \{0, 1, 2, 3, \ldots\}. \quad (60)$$

Then, for the sequence $\{V_{i_\eta}(x)\}$, we have the conclusion that

$$\lim_{i \to \infty} V_i(x) = \lim_{\eta \to \infty} V_{i_\eta}(x). \quad (61)$$

Here, we denote the limit of the sequence $\{V_{i_\eta}(x)\}$ as $V_\infty^L(x)$.

For any $\eta = 0, 1, 2, 3, \ldots$ and $x \in X$, according to (50), the iterative value function $V_{i_\eta+1}^{q+1}$ is defined as

$$V_{i_\eta+1}^{q+1}(x) = U\big(x, d_{i_\eta+1}(x)\big) + \sum_{j \in X} \lambda p\big(j \,|\, x, d_{i_\eta+1}(x)\big) V_{i_\eta+1}^{q}(j) \quad (62)$$

where $V_{i_\eta+1}^{0}(x) = V_{i_\eta}(x)$ and $d_{i_\eta+1}(x)$ is defined in (7). For $q = 0$, according to (34) and (35), we obtain

$$\begin{cases} V_{i_\eta+1}^{1}(x) \leq V_{i_\eta}(x) \ \forall x \in B_x^{i_\eta} \\ V_{i_\eta+1}(x) = V_{i_\eta}(x) \ \forall x \in X \backslash \left(B_x^{i_\eta} \right) \end{cases} \quad (63)$$

which means

$$V_{i_\eta+1}^{1}(x) \leq V_{i_\eta}(x) \ \forall x \in X. \quad (64)$$

For $q = 1$ and for all $x \in X$, we get

$$\begin{aligned} V_{i_\eta+1}^{2}(x) &= U\big(x, d_{i_\eta+1}(x)\big) + \sum_{j \in X} \lambda p\big(j \,|\, x, d_{i_\eta+1}(x)\big) V_{i_\eta+1}^{1}(j) \\ &\leq U\big(x, d_{i_\eta+1}(x)\big) + \sum_{j \in X} \lambda p\big(j \,|\, x, d_{I_\eta+1}(x)\big) V_{i_\eta}(j) \\ &= V_{i_\eta+1}^{1}(x). \end{aligned} \quad (65)$$

By mathematical induction, we can derive that

$$V_{i_\eta+1}^{q+1}(x) \leq V_{i_\eta+1}^{q}(x) \quad (66)$$

holds for all $x \in X$, and $q = 0, 1, 2, 3, \ldots$

Letting $q \to \infty$, based on Theorem 1, for any $\eta = 0, 1, 2, 3, \ldots$ and all $x \in X$, we can get

$$V_{i_\eta+1}^{1}(x) \geq \lim_{q \to \infty} V_{i_\eta+1}^{q}(x) = V_{i_\eta+1}(x). \quad (67)$$

According to (62)–(64), for all $x \in B_x^{i_\eta}$, we have

$$\begin{aligned} V_{i_\eta+1}^{1}(x) &= U\big(x, d_{i_\eta+1}(x)\big) + \sum_{j \in X} \lambda p\big(j \,|\, x, d_{i_\eta+1}(x)\big) V_{i_\eta+1}^{0}(j) \\ &= U\big(x, d_{i_\eta+1}(x)\big) + \sum_{j \in X} \lambda p\big(j \,|\, x, d_{i_\eta+1}(x)\big) V_{i_\eta}(j) \\ &= \min_{a \in A_x} \left\{ U(x, a) + \sum_{j \in X} \lambda p(j \,|\, x, a) V_{i_\eta}(j) \right\} \\ &\leq V_{i_\eta}(x). \end{aligned} \quad (68)$$

Letting $\eta \to \infty$, we get

$$\min_{a \in A_x} \left\{ U(x, a) + \sum_{j \in X} \lambda p(j \,|\, x, a) V_\infty^L(j) \right\} \leq V_\infty^L(x) \quad (69)$$

holds for all $x \in \cap_{\eta=0}^{\infty} B_x^{i_\eta}$. It is very obvious that (69) holds for $\widetilde{x}$. Then, since $\widetilde{x}$ is chosen arbitrarily in the set $X \backslash G$, we can get the conclusion that (69) holds for all $x \in X \backslash G$.

Suppose for all $x \in X$, we have

$$d_\infty^L = \lim_{i \to \infty} d_i(x) \quad (70)$$

where $d_i(x)$ is defined in (7). Then, according to (4)–(7) we can get

$$V_\infty^L(x) = U\big(x, d_\infty^L(x)\big) + \sum_{j\in X} \lambda p\big(j \,|\, x, d_\infty^L(x)\big) V_\infty^L(j). \quad (71)$$

Now, let us set the decision rule $\widetilde{d}_\infty^L$ as follows. For $x \in G$, define $\widetilde{d}_\infty^L(x)$ as

$$\widetilde{d}_\infty^L(x) = d_\infty^L(x). \quad (72)$$

For $x \in X\backslash G$, define $\widetilde{d}_\infty^L(x)$ as

$$\widetilde{d}_\infty^L(x) = \min_{a\in A_x} \left\{ U(x,a) + \sum_{j\in X} \lambda p(j \,|\, x, a) V_\infty^L(j) \right\}. \quad (73)$$

Then, define the set D_k^L of the decision rules at time k as follows:

$$D_k^L = \big\{ d(x,k) \,|\, d(x,k) \in D_k \,:\, d(x,k) = d_\infty^L(x) \ \forall x \in G \big\}. \quad (74)$$

Besides, we set the subspace Θ of the policy space Ξ as follows:

$$\Theta = \big\{ D_0^L \times D_1^L \times D_2^L \times D_3^L \cdots \big\}. \quad (75)$$

Then, according to (71)–(74), we can transform (69) into

$$\begin{aligned} &\min_{d\in D^L} \left\{ U(x, d(x)) + \sum_{j\in X} \lambda p(j \,|\, x, d(x)) V_\infty^L(j) \right\} \\ &\quad = \inf_{d\in D^L} \left\{ U(x, d(x)) + \sum_{j\in X} \lambda p(j \,|\, x, d(x)) V_\infty^L(j) \right\} \\ &\quad \le V_\infty^L(x) \end{aligned} \quad (76)$$

where $D^L = D_0^L = D_1^L = D_2^L \ldots$

Writing (76) in vector form, we can get

$$\inf_{d\in D^L} \big\{ \mathbf{U}_{d(x)} + \lambda \mathbf{P}_{d(x)} \mathbf{V}_\infty^L \big\} \le \mathbf{V}_\infty^L. \quad (77)$$

According to (77), we have that for arbitrary $\varepsilon > 0$, there exists a policy which is expressed as $\pi = (d(x,0), d(x,1), d(x,2), \ldots, d(x,k), \ldots) \in \Theta$, such that

$$\begin{aligned} \mathbf{V}_\infty^L &\ge \mathbf{U}_{d(x,0)} + \lambda \mathbf{P}_{d(x,0)} \mathbf{V}_\infty^L - \varepsilon e \\ &\ge \mathbf{U}_{d(x,0)} + \lambda \mathbf{P}_{d(x,0)} \big\{ \mathbf{U}_{d(x,1)} + \lambda \mathbf{P}_{d(x,1)} \mathbf{V}_\infty^L - \varepsilon e \big\} - \varepsilon e \\ &= \mathbf{U}_{d(x,0)} + \lambda \mathbf{P}_{d(x,0)} \mathbf{U}_{d(x,1)} + \lambda^2 \mathbf{P}_{d(x,0)} \mathbf{P}_{d(x,1)} \mathbf{V}_\infty^L \\ &\quad - (\varepsilon e + \lambda \varepsilon e) \\ &\ge \mathbf{U}_{d(x,0)} + \lambda \mathbf{P}_{d(x,0)} \mathbf{U}_{d(x,1)} + \lambda^2 \mathbf{P}_{d(x,0)} \mathbf{P}_{d(x,1)} \\ &\quad \times \big\{ \mathbf{U}_{d(x,2)} + \lambda \mathbf{P}_{d(x,2)} \mathbf{V}_\infty^L - \varepsilon e \big\} - (\varepsilon e + \lambda \varepsilon e) \\ &= \mathbf{U}_{d(x,0)} + \lambda \mathbf{P}_{d(x,0)} \mathbf{U}_{d(x,1)} + \lambda^2 \mathbf{P}_{d(x,0)} \mathbf{P}_{d(x,1)} \mathbf{U}_{d(x,2)} \\ &\quad + \lambda^3 \mathbf{P}_{d(x,0)} \mathbf{P}_{d(x,1)} \mathbf{P}_{d(x,2)} \mathbf{V}_\infty^L - \Big(\varepsilon e + \lambda \varepsilon e + \lambda^2 \varepsilon e \Big) \\ &\ge \mathbf{U}_{d(x,0)} + \sum_{l=1}^{n-1} \left\{ \lambda^l \left\{ \prod_{m=0}^{l-1} \mathbf{P}_{d(x,m)} \right\} \mathbf{U}_{d(x,l)} \right\} \\ &\quad + \lambda^n \left\{ \prod_{m=0}^{n-1} \mathbf{P}_{d(x,m)} \right\} \mathbf{V}_\infty^L \\ &\quad - \Big(\varepsilon e + \lambda \varepsilon e + \lambda^2 \varepsilon e + \cdots + \lambda^{n-1} \varepsilon e \Big). \end{aligned} \quad (78)$$

Then, letting $n \to \infty$, we have

$$\lim_{n\to\infty} \lambda^n \left\{ \prod_{m=0}^{n-1} \mathbf{P}_{d(x,m)} \right\} \mathbf{V}_\infty^L = \mathbf{0} \quad (79)$$

$$\lim_{n\to\infty} \left\{ \mathbf{U}_{d(x,0)} + \sum_{l=1}^{n-1} \left\{ \lambda^l \left\{ \prod_{m=0}^{l-1} \mathbf{P}_{d(x,m)} \right\} \mathbf{U}_{d(x,l)} \right\} \right\} = \mathbf{V}^\pi. \quad (80)$$

Then, when $n \to \infty$, (78) becomes

$$\mathbf{V}_\infty^L \ge \mathbf{V}^\pi - (1-\lambda)^{-1} \varepsilon e. \quad (81)$$

Since ε was arbitrary, we can easily get

$$\mathbf{V}_\infty^L \ge \inf_{\pi\in\Theta} \mathbf{V}^\pi. \quad (82)$$

On the other hand, according to (67), for all $x \in B_x^{i_\eta}$, we can obtain

$$\begin{aligned} V_{i_\eta+1}^1(x) &= U\big(x, d_{i_\eta+1}(x)\big) + \sum_{j\in X} \lambda p\big(j \,|\, x, d_{i_\eta+1}(x)\big) V_{i_\eta+1}^0(j) \\ &= U\big(x, d_{i_\eta+1}(x)\big) + \sum_{j\in X} \lambda p\big(j \,|\, x, d_{i_\eta+1}(x)\big) V_{i_\eta}(j) \\ &= \min_{a\in A_x} \left\{ U(x,a) + \sum_{j\in X} \lambda p(j \,|\, x, a) V_{i_\eta}(j) \right\} \\ &\ge V_{i_\eta+1}(x). \end{aligned} \quad (83)$$

Letting $\eta \to \infty$, for all $x \in \cap_{\eta=0}^\infty B_x^{i_\eta}$, we can get

$$\min_{a\in A_x} \left\{ U(x,a) + \sum_{j\in X} \lambda p(j \,|\, x, a) V_\infty^L(j) \right\} \ge V_\infty^L(x). \quad (84)$$

It is obvious that (84) holds also for $\widetilde{x}$. Then, since $\widetilde{x}$ is chosen arbitrarily in the set $X\backslash G$, we can get the conclusion that (84) holds for all $x \in X\backslash G$.

Then, according to (71)–(74), we can transform (84) into

$$\begin{aligned} &\min_{d\in D^L} \left\{ U(x, d(x)) + \sum_{j\in X} \lambda p(j \,|\, x, d(x)) V_\infty^L(j) \right\} \\ &\quad = \inf_{d\in D^L} \left\{ U(x, d(x)) + \sum_{j\in X} \lambda p(j \,|\, x, d(x)) V_\infty^L(j) \right\} \\ &\quad \ge V_\infty^L(x). \end{aligned} \quad (85)$$

Writing (85) in vector format, we can get

$$\inf_{d\in D^L} \big\{ \mathbf{U}_{d(x)} + \lambda \mathbf{P}_{d(x)} \mathbf{V}_\infty^L \big\} \ge \mathbf{V}_\infty^L. \quad (86)$$

Then, choosing $\pi = (d(x,0), d(x,1), d(x,2), \ldots, d(x,k), \ldots) \in \Theta$, according to (86), we have

$$\begin{aligned} \mathbf{V}_\infty^L &\le \mathbf{U}_{d(x,0)} + \lambda \mathbf{P}_{d(x,0)} \mathbf{V}_\infty^L \\ &\le \mathbf{U}_{d(x,0)} + \lambda \mathbf{P}_{d(x,0)} \big\{ \mathbf{U}_{d(x,1)} + \lambda \mathbf{P}_{d(x,1)} \mathbf{V}_\infty^L \big\} \\ &= \mathbf{U}_{d(x,0)} + \lambda \mathbf{P}_{d(x,0)} \mathbf{U}_{d(x,1)} + \lambda^2 \mathbf{P}_{d(x,0)} \mathbf{P}_{d(x,1)} \mathbf{V}_\infty^L. \end{aligned} \quad (87)$$

By mathematical induction, it follows that, for $n \geq 1$:

$$\mathbf{V}_{\infty}^{L} \leq \mathbf{U}_{d(x,0)} + \sum_{l=1}^{n-1}\left\{\lambda^{l}\left\{\prod_{m=0}^{l-1}\mathbf{P}_{d(x,m)}\right\}\mathbf{U}_{d(x,l)}\right\} + \lambda^{n}\left\{\prod_{m=0}^{n-1}\mathbf{P}_{d(x,m)}\right\}\mathbf{V}_{\infty}^{L}. \tag{88}$$

Thus

$$\mathbf{V}_{\infty}^{L} - \mathbf{V}^{\pi} \leq \lambda^{n}\left\{\prod_{m=0}^{n-1}\mathbf{P}_{d(x,m)}\right\}\mathbf{V}_{\infty}^{L} - \sum_{l=1}^{\infty}\left\{\lambda^{l}\left\{\prod_{m=0}^{l-1}\mathbf{P}_{d(x,m)}\right\}\mathbf{U}_{d(x,l)}\right\}. \tag{89}$$

Choosing $\varepsilon > 0$, since

$$\left\|\lambda^{n}\left\{\prod_{m=0}^{n-1}\mathbf{P}_{d(x,m)}\right\}\mathbf{V}_{\infty}^{L}\right\| \leq \lambda^{n}\left\|\mathbf{V}_{\infty}^{L}\right\| \tag{90}$$

and $0 \leq \lambda < 1$, if n is sufficiently large, we have

$$-(\varepsilon/2)\mathbf{e} \leq \lambda^{n}\left\{\prod_{m=0}^{n-1}\mathbf{P}_{d(x,m)}\right\}\mathbf{V}_{\infty}^{L} \leq (\varepsilon/2)\mathbf{e} \tag{91}$$

where $\mathbf{e}$ denotes a vector of 1's. As result of the assumption

$$-\frac{\lambda^{n}M\mathbf{e}}{1-\lambda} \leq \sum_{l=1}^{\infty}\left\{\lambda^{l}\left\{\prod_{m=0}^{l-1}\mathbf{P}_{d(x,m)}\right\}\mathbf{U}_{d(x,l)}\right\} \leq \frac{\lambda^{n}M\mathbf{e}}{1-\lambda}. \tag{92}$$

By choosing a sufficiently large n, the second expression on the right-hand side of (89) can be bounded above and below by $(\varepsilon/2)\mathbf{e}$. Since the left-hand side of (89) does not depend on n, it follows that:

$$\mathbf{V}_{\infty}^{L} \leq \mathbf{V}^{\pi} + \varepsilon e. \tag{93}$$

Since ε was arbitrary, then we can get

$$\mathbf{V}_{\infty}^{L} \leq \inf_{\pi\in\Theta}\mathbf{V}^{\pi}. \tag{94}$$

According to (82) and (94), we can get the conclusion that

$$\mathbf{V}_{\infty}^{L} = \inf_{\pi\in\Theta}\mathbf{V}^{\pi}. \tag{95}$$

This completes the proof. ■

Remark 2: From Theorem 2, it is shown that the sequence of iterative value functions is monotonically decreasing and will finally converge to the optimum in a local policy space with the local policy iteration ADP algorithm. We should also pay attention to the situation where the iterative decision rule $d_i(x)$ is only refreshed for finite times for some state $x \in \Omega$. Then under this situation, we could get the conclusion that the sequence of iterative value functions obtained by the proposed algorithm are monotonically nonincreasing and will converge to the optimal performance index function in a local policy space. Hence, if the sequence of the iterative decision rules $d_i(x)$ are refreshed only finite times for some system state $x \in X$, then it is not guaranteed that the last converged value function obtained by the proposed algorithm is optimal in the global policy space. Thus, to guarantee that the sequence of the iterative value functions finally converge to the global optimum, the iterative decision rule should be refreshed for every state which is within the whole system state set for infinite times. Then it is not guaranteed that the last converged value function obtained by the proposed algorithm is optimal in the global policy space. We will discuss this property further in the following theorem.

Theorem 3 (Global Optimality Property): For $i = 0, 1, 2, 3, \ldots$ and all $x \in X$, let $V_i(x)$ and $d_{i+1}(x)$ be obtained by (4)–(7). Let $\mathcal{N}_j, j = 01, 2, 3, \ldots,$ be non-negative integers, which satisfy $\mathcal{N}_0 \leq \mathcal{N}_1 \leq \ldots$ Assume that $i_{\mathcal{N}_j} \leq i_{\mathcal{N}_{j+1}}$ $\forall j = 0, 1, 2, 3, \ldots$ If for any $j = 0, 1, 2, 3, \ldots$, the state set $B_x^i, i = 0, 1, 2, 3, \ldots,$ satisfies

$$\cup_{l=i_{\mathcal{N}_{j-1}}}^{i_{\mathcal{N}_j}} B_x^i = X \tag{96}$$

where we let $i_{\mathcal{N}_{j-1}} = -1$ for $j = 0$. Then, the iterative value function $V_{i_{\mathcal{N}_j}}(x)$ satisfies

$$\lim_{j\to\infty} V_{i_{\mathcal{N}_j}}(x) = \inf_{\pi\in\Xi} V^{\pi}(x). \tag{97}$$

Proof: As $i_{\mathcal{N}_j} \leq i_{\mathcal{N}_{j+1}}$ $\forall j = 0, 1, 2, 3, \ldots$, we have $i_{\mathcal{N}_j} \to \infty$ with $j \to \infty$. Based on Theorem 1, the sequence of the iterative value functions will finally converge, that is,

$$V_{\infty}(x) = \lim_{j\to\infty} V_{i_{\mathcal{N}_j}}(x). \tag{98}$$

From (96), we can get the conclusion that given any $x \in X$, there always exists an i_{ρ_j}, where $i_{\rho_j} \in \{0, 1, 2, 3, \ldots\}, i_{\mathcal{N}_{j-1}+1} \leq i_{\rho_j} \leq i_{\mathcal{N}_j}, j = 0, 1, 2, 3, \ldots$ satisfying $x \in B_x^{i_{\rho_j}}$. That is to say, given any $x \in X$, we have

$$x \in \cap_{j=0}^{\infty} B_x^{i_{\rho_j}}. \tag{99}$$

Then, according to the definition of the set G, we can get the conclusion that

$$G = \emptyset. \tag{100}$$

Then, similar to Theorem 2, we can easily get the conclusion that

$$\lim_{\eta\to\infty} V_{i_\eta}(x) = \inf_{\pi\in\Theta} V^{\pi}(x) \tag{101}$$

where $\Theta = \Xi$. ■

B. Summary of the Local Policy Iteration ADP Algorithm

Based on the above preparations, we summarize the local policy iteration ADP algorithm as in Algorithm 1.

IV. Neural Network Implementation for Local Policy Iteration ADP Algorithm

In practical cases, the number of system states for the stochastic processes is usually very large. Hence, we need to utilize approximation structures such as neural networks to approximate the iterative value functions and the iterative decision rules obtained by the policy iteration algorithm. Here, we use three-layer BP neural networks to approximate $d_i(x)$ and $V_i(x)$ for $i = 0, 1, 2, 3, \ldots$ We use κ to define the hidden

Algorithm 1 Local Policy Iteration ADP Algorithm for Stochastic Processes

Initialization:

Choose parameter λ;
Choose a computation precision ε;
Give an initial decision rule $d_0(x)$;
Construct a sequence of the local spaces of the whole system state set as $\{B_x^i\}$, $i = 0, 1, 2, 3 \ldots$.

Iteration:

1: Let the iteration index $i = 0$;
2: Obtain the value function $V_0(x)$ by (4);
3: Let $i = i + 1$;
4: Do **Policy Improvement**
If $x \in B_x^{i-1}$, **then**

$$d_i(x) = \arg\min_{a \in A_x} \left\{ U(x, a) + \sum_{j \in X} \lambda p(j|x, a) V_{i-1}(j) \right\}$$

else

$$d_i(x) = d_{i-1}(x)$$

end if;
5: Obtain $V_i(x)$ by doing **Policy Evaluation**

$$V_i(x) = U(x, d_i(x)) + \sum_{j \in X} \lambda p(j|x, d_i(x)) V_i(j);$$

6: If $\|\mathbf{V}_i - \mathbf{V}_{i-1}\| < \varepsilon$, then the optimal performance index function and optimal decision rule are obtained. Goto Step 7. Else goto Step 3;
7: **return** $V_i(x)$ and $d_i(x)$.

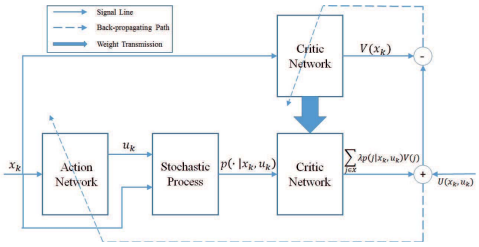

Fig. 1. Structure diagram of the algorithm.

layer size, W_h to define the weight matrix from the input layer to hidden layer, and W_o to define the weight matrix from hidden layer to output layer. We use b_h to define the bias vector for hidden layer and b_o to define the bias vector for output layer. Then, we can represent our neural network as

$$\hat{N}(W_h, W_o, b_h, b_o, x) = W_o^{\mathsf{T}} \left\{ f\left(W_h^{\mathsf{T}} x + b_h \right) \right\} + b_o \quad (102)$$

where $f(W_h^{\mathsf{T}} x + b_h)$ is a κ-dimensional vector, $[f(z)]_i = 2/(1 + e^{(-2z_i)}) - 1$, $i = 1, 2, 3, \ldots, \kappa$. Here, we use f as the activation function.

In our algorithm, we utilize two neural networks [also known as critic network $\hat{V}_i(x)$ and action network $\hat{d}_i(x)$] to approximate the performance index function $V_i(x)$ and the decision rule $d_i(x)$, respectively. We demonstrate the overall structure diagram in Fig. 1.

A. Critic Network

We utilize the critic network to approximate the iterative performance index function $V_i(x)$. The output of the critic network can be expressed as

$$\hat{V}_i^l(x) = W_{\text{oci}}^{l\mathsf{T}} \left\{ f\left(W_{hc}^{\mathsf{T}} x + b_{hc} \right) \right\} + b_{\text{oci}}^l \quad (103)$$

where $l = 0, 1, 2, 3, \ldots$. Let W_{oci}^0, W_{hc}, b_{hc}, and b_{oci}^0 be random weight matrices. In simulation, when we train the neural network, the hidden-output weight matrix W_{oci}^l and the bias b_{oci}^l will be refreshed, while the input-hidden weight matrix W_{hc} and the bias b_{hc} are fixed. Here, we can present the target of the critic network as

$$V_i(x) = U\left(x, \hat{d}_i(x) \right) + \sum_{j \in X} \left(\lambda p(j|x, \hat{d}_i(x)) \hat{V}_i^l(j) \right). \quad (104)$$

Then, we define the error function for the critic network as

$$e_{ci}^l(x) = V_i(x) - \hat{V}_i^l(x). \quad (105)$$

Then, applying (103) and (104) into (105), we can obtain

$$e_{ci}^l(x) = \left\{ U\left(x, \hat{d}_i(x) \right) + \sum_{j \in X} \left(\lambda p\left(j|x, \hat{d}_i(x) \right) \hat{V}_i^l(j) \right) \right\} - \left\{ W_{\text{oci}}^{l\mathsf{T}} \left\{ f\left(W_{hc}^{\mathsf{T}} x + b_{hc} \right) \right\} + b_{\text{oci}}^l \right\}. \quad (106)$$

Combining (103) with (106), we can define ∇_c as

$$\begin{aligned} \nabla_c &\triangleq \frac{\partial e_{ci}^l(x)}{\partial W_{\text{oci}}^l} \\ &= \left\{ \sum_{j \in X} \left\{ \lambda p\left(j|x, \hat{d}_i(x) \right) f\left(W_{hc}^{\mathsf{T}} j + b_{hc} \right) \right\} \right\} \\ &\quad - f\left(W_{hc}^{\mathsf{T}} x + b_{hc} \right). \end{aligned} \quad (107)$$

The objective function to be minimized in the critic network is

$$E_{ci}^l(x) = \frac{1}{2} \left(e_{ci}^l(x) \right)^2. \quad (108)$$

The gradient-based weight update rule can be applied to train the critic network

$$\begin{aligned} W_{\text{oci}}^{l+1} &= W_{\text{oci}}^l + \Delta W_{\text{oci}}^l \\ &= W_{\text{oci}}^l - \rho_{cw} \left\{ \frac{\partial E_{ci}^l(x)}{\partial e_{ci}^l(x)} \frac{\partial e_{ci}^l(x)}{\partial W_{\text{oci}}^l} \right\} \\ &= W_{\text{oci}}^l - \rho_{cw} e_{ci}^l(x) \nabla_c \end{aligned} \quad (109)$$

$$\begin{aligned} b_{\text{oci}}^{l+1} &= b_{\text{oci}}^l + \Delta b_{\text{oci}}^l \\ &= b_{\text{oci}}^l - \rho_{cb} \left\{ \frac{\partial E_{ci}^l(x)}{\partial \hat{V}_{i+1}^l(x)} \frac{\partial \hat{V}_{i+1}^l(x)}{\partial b_{\text{oci}}^l} \right\} \\ &= b_{\text{oci}}^l - \rho_{cb} e_{ci}^l(x) \end{aligned} \quad (110)$$

where $\rho_{cw} > 0$ and $\rho_{cb} > 0$ are the learning rates of the critic network. If the training precision is achieved, then we say that the performance index function $V_i(x)$ can be approximated by the critic network.

B. Action Network

Here, we utilize the action network to approximate the iterative decision rule $d_i(x)$. The output of the action network can be expressed as

$$\hat{d}_i^l(x) = W_{oai}^{l\mathsf{T}}\left\{f\left(W_{ha}^{\mathsf{T}}x + b_{ha}\right)\right\} + b_{oai}^l \tag{111}$$

where $l = 0, 1, 2, 3, \ldots$ Let W_{oai}^0, W_{ha}, b_{ha}, and b_{oai}^0 be random weight matrices. In simulation, when we train the neural network, the hidden-output weight matrix W_{oai}^l and the bias b_{oai}^l will be refreshed, while the input-hidden weight matrix W_{ha} and the bias b_{ha} are fixed. Here, we present the target function $d_i(x)$ as follows. If $x \in B_x^{i-1}$, then

$$d_i(x) = \arg\min_{a\in A_x}\left\{U(x,a) + \sum_{j\in X}\lambda p(j|x,a)\hat{V}_{i-1}(j)\right\}. \tag{112}$$

If $x \in X\backslash B_x^{i-1}$, then

$$d_i(x) = \hat{d}_{i-1}(x). \tag{113}$$

Then, we define the error function for the action network for all $x \in X$ as

$$e_{ai}^l(x) = \hat{d}_i^l(x) - d_i(x). \tag{114}$$

The objective function to be minimized in the action network is

$$E_{ai}^l(x) = \frac{1}{2}\left(e_{ai}^l(x)\right)^{\mathsf{T}}\left(e_{ai}^l(x)\right). \tag{115}$$

The gradient-based weight update rule can be applied here to train the action network

$$\begin{aligned} W_{oai}^{l+1} &= W_{oai}^l + \Delta W_{oai}^l \\ &= W_{oai}^l - \rho_{aw}\left\{\frac{\partial E_{ai}^l(x)}{\partial e_{ai}^l(x)}\frac{\partial e_{ai}^l(x)}{\partial \hat{d}_i^l(x)}\frac{\partial \hat{d}_i^l(x)}{\partial W_{oai}^l}\right\} \\ &= W_{oai}^l - \rho_{aw} f\left(W_{ha}^{\mathsf{T}}x + b_{ha}\right)\left(e_{ai}^l(x)\right)^{\mathsf{T}} \end{aligned} \tag{116}$$

$$\begin{aligned} b_{oai}^{l+1} &= b_{oai}^l + \Delta b_{oai}^l \\ &= b_{oai}^l - \rho_{ab}\left\{\frac{\partial E_{ai}^l(x)}{\partial e_{ai}^l(x)}\frac{\partial e_{ai}^l(x)}{\partial \hat{d}_i^l(x)}\frac{\partial \hat{d}_i^l(x)}{\partial b_{oai}^l}\right\} \\ &= b_{oai}^l - \rho_{ab}e_{ai}^l(x) \end{aligned} \tag{117}$$

where $\rho_{aw} > 0$ and $\rho_{ab} > 0$ are the learning rates of the action network. If the training precision is achieved, then we say that the decision rule $d_i(x)$ can be approximated by the action network.

Remark 3: With the above algorithm implemented with the help of neural networks, we actually get a sequence of approximate decision rules and a sequence of approximate value functions. Hence, there will be errors between the approximate decision rules and the "true" decision rules. And there will also be errors between the approximate value functions and the "true" value functions. In the following section, we will establish the error bounds between the approximate performance index function and the optimal performance index function under some conditions.

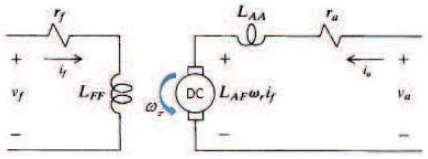

Fig. 2. Direct-current machine.

C. Error Bounds of Local Policy Iteration ADP Algorithm

We consider the local policy iteration ADP algorithm which generates a sequence of approximate iterative decision rules $\hat{d}_i(x)$ and a corresponding sequence of approximate iterative value functions $\hat{V}_i(x)$ satisfying

$$\left\|\hat{\mathbf{V}}_i - \mathbf{V}^{\hat{\pi}_i}\right\| \le \epsilon \ \forall i = 0, 1, 2, 3, \ldots \tag{118}$$

and

$$\sup\left|L_{\hat{d}_{i+1}}\left(\hat{V}_i(x)\right) - L\left(\hat{V}_i(x)\right)\right| \le \delta \ \forall i = 0, 1, 2, 3, \ldots \tag{119}$$

where ϵ and δ are some positive scalars, stationary policy $\hat{\pi}_0 = \pi_0 = (d_0(x), d_0(x), d_0(x), \ldots, d_0(x), \ldots)$ is the given initial policy, $\hat{\pi}_i = (\hat{d}_i(x), \hat{d}_i(x), \hat{d}_i(x), \ldots, \hat{d}_i(x), \ldots)$ is the approximate iterative policy, and the operator $L : \mathcal{V} \to \mathcal{V}$ is defined as $L(V(x)) = \min_{a\in A_x}\{U(x,a) + \sum_{j\in X}\lambda p(j|x,a)V_i(j)\}$.

Then, we can establish the error bounds between the approximate performance index function and the optimal performance index function.

Lemma 2 (See [10]): The sequence of the approximate value functions generated by the local policy iteration ADP algorithm satisfies

$$\limsup_{i\to\infty}\left\|\hat{V}_i - \inf_{\pi\in\Theta}\mathbf{V}^{\pi}\right\| \le \frac{\delta + 2\lambda\epsilon}{1-\lambda^2}. \tag{120}$$

V. Simulation Example

We consider the system of the direct-current machine, which is shown in Fig. 2. i_f is the current of the field winding; r_a is the resistance of the rotor coil; L_{AA} is the self-inductance of the armature winding; and L_{AF} is the mutual inductance between the field and the rotating armature coils. A random exogenous load torque T_L is applied to the system. $\mathcal{J}$ is the inertia of the rotor. The constant B_m is a damping coefficient associated with the mechanical rotational system of the machine. Let armature current i_a and rotor speed ω_r be the system states and let armature voltage v_a be the control input. This direct-current machine system can be derived as

$$\begin{aligned} \frac{\mathrm{d}i_a}{dt} &= -\frac{L_{AF}}{L_{AA}}i_f\omega_r - \frac{r_a}{L_{AA}}i_a + \frac{1}{L_{AA}}v_a \\ \frac{\mathrm{d}\omega_r}{dt} &= \frac{L_{AF}}{\mathcal{J}}i_f i_a - \frac{B_m}{\mathcal{J}}\omega_r - \frac{1}{\mathcal{J}}T_L. \end{aligned} \tag{121}$$

We will discretize the above system using the sampling interval ΔT. This yields

$$\begin{aligned} i_a(k+1) &= \frac{-\Delta T L_{AF}}{L_{AA}}i_f\omega_r(k) + \frac{L_A - \Delta T r_a}{L_{AA}}i_a(k) + \frac{\Delta T}{L_{AA}}v_a(k) \\ \omega_r(k+1) &= \frac{\Delta T L_{AF}}{\mathcal{J}}i_f i_a(k) + \frac{\mathcal{J} - \Delta B_m}{\mathcal{J}}\omega_r(k) - \frac{\Delta T}{\mathcal{J}}T_L(k). \end{aligned} \tag{122}$$

TABLE I
PARAMETERS SETTING UP

Parameters	Value
r_a	$11.2ohms$
L_{AA}	$0.1215H$
L_{AF}	$1.976H$
$\mathcal{J}$	$0.02215kg.m^2$
B_m	$0.002953N.m.s$
ΔT	$0.005s$
k_1	0.001
k_2	0.001
k_3	1
i_f	$1amp(fixed)$
ω_c	$100rad/s$
λ	0.7
T_L	$Gaussian\ Noise(N.m)$

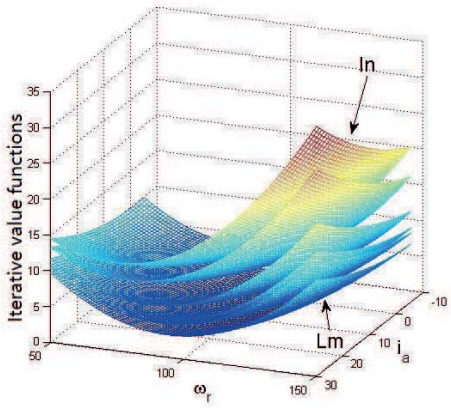

Fig. 3. Iterative value functions by local policy iteration ADP algorithm.

The utility function is chosen as

$$U(i_a, \omega_r, v_a) = U_1(i_a, \omega_r, v_a) + U_2(i_a, \omega_r, v_a) + U_3(i_a, \omega_r, v_a) \tag{123}$$

where

$$\begin{aligned} U_1(i_a, \omega_r, v_a) &= k_1(\omega_r - \omega_c)^2 \\ U_2(i_a, \omega_r, v_a) &= k_2 i_a^2 \\ U_3(i_a, \omega_r, v_a) &= k_3 \left\{ i_a - \left\{ \frac{-\Delta T L_{AF}}{L_{AA}} i_f \omega_r + \frac{L_A - \Delta T r_a}{L_{AA}} i_a + \frac{\Delta T}{L_{AA}} v_a \right\} \right\}^2. \end{aligned} \tag{124}$$

In the utility function, $U_1(i_a, \omega_r, v_a)$ is to make the rotors speed track the arbitrarily given speed command ω_c; $U_2(i_a, \omega_r, v_a)$ is to minimize the energy consumed by the system; and $U_3(i_a, \omega_r, v_a)$ is to ensure that the armature current changes as smooth as possible.

The parameters for the model are given in Table I.

We denote system state x as

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} i_a \\ \omega_r \end{bmatrix}. \tag{125}$$

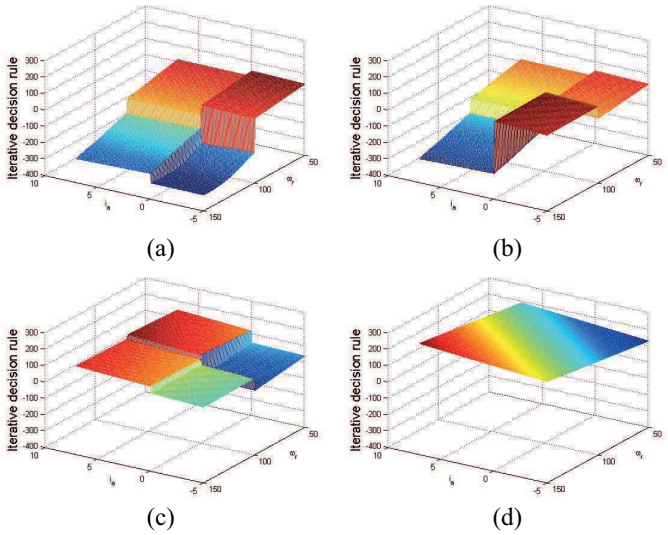

Fig. 4. Iterative decision rules by local policy iteration ADP algorithm. (a) $d_0(x)$. (b) $d_1(x)$. (c) $d_3(x)$. (d) $d_\infty(x)$.

Let the state space be expressed as $\Omega = \{(x_1, x_2) \mid -5 \le x_1 \le 30, 0 \le x_2 \le 200\}$. Let the initial state be $x_0 = [0, 0]^{\mathsf{T}}$. Neural networks are used to implement the developed local iterative ADP algorithm. For $y \ge 0$ and $z \ge 0$, let $\mathrm{rem}(y, z)$ denote the remainder of y/z. For $i = 0, 1, \ldots,$ the state set $B_x^{\mathrm{rem}(i,4)}$ is defined as

$$B_x^{\mathrm{rem}(i,4)} \in \left\{ B_x^0, B_x^1, B_x^2, B_x^3 \right\} \tag{126}$$

where we let $B_x^0 = \{0 \le x_1 \le 30, 100 \le x_2 \le 200\}$, $B_x^1 = \{-5 \le x_1 \le 0, 100 \le x_2 \le 200\}$, $B_x^2 = \{-5 \le x_1 \le 0, 0 \le x_2 \le 100\}$, and $B_x^3 = \{0 \le x_1 \le 30, 0 \le x_2 \le 100\}$. From (126), we know that $\cup_{j=0}^3 B_x^j = \Omega$. For $i = 0, 1, 2, 3, \ldots$, the iterative decision rule is refreshed in local space of the whole system state set. And for any $x \in X$, these local spaces of the whole system state set satisfy

$$x \in B_x^{\mathrm{rem}(i,4)}, i = 0, 1, 2, 3, \ldots \tag{127}$$

According to [10], we can build the initial decision rule using action network, where the initial decision rule can be expressed as $v_{a0}(x) = W_{a2,\mathrm{initial}}\sigma(Y_{a1,\mathrm{initial}}x + b_{a1,\mathrm{initial}}) + b_{a2,\mathrm{initial}}$. Here, the function $\sigma(\cdot)$ is chosen as the sigmoid function. We set up the values for the parameters randomly.

In numerical implementation, we refresh the iterative decision rule and iterative value function for 100 times until the computation precision $\epsilon = 0.01$ is satisfied. Fig. 3 demonstrates the iterative value functions using local policy iteration ADP algorithm, where "In" indicates the initial iteration item and "Lm" indicates the limiting iteration item. Fig. 4 demonstrates the iterative decision rules $d_i(x)$ using local policy iteration ADP algorithm. As in each iteration, the iterative decision rule $d_i(x)$ is only renewed in a local space of the global state set, i.e., $B_x^{\mathrm{rem}(i,4)\in\Omega}$. From Figs. 3 and 4, we can see that the sequence of iterative value functions obtained by the proposed algorithm are monotonically nonincreasing and will reach certain convergency. Fig. 6 demonstrates the iterative control trajectories using local policy iteration ADP algorithm. Fig. 5 demonstrates the iterative system state trajectories using

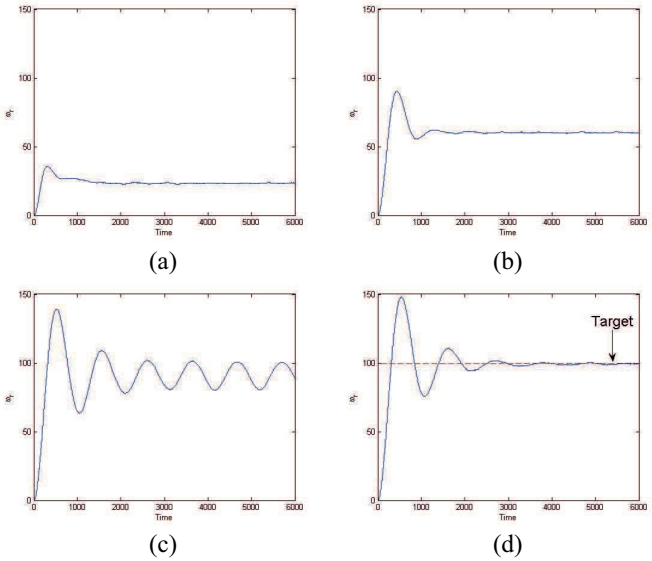

Fig. 5. Iterative trajectories of rotor speed. (a) ω_r with $i = 0$. (b) ω_r with $i = 4$. (c) ω_r with $i = 11$. (d) ω_r with $i = \infty$.

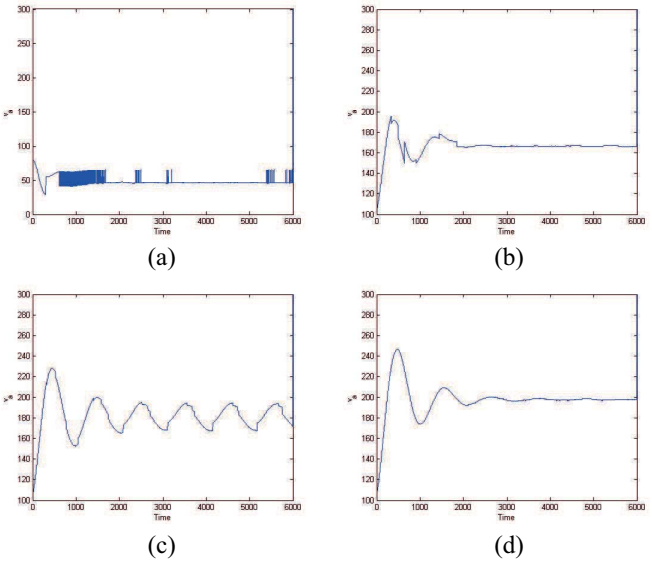

Fig. 6. Iterative trajectories of armature voltage. (a) v_a with $i = 0$. (b) v_a with $i = 8$. (c) v_a with $i = 11$. (d) v_a with $i = \infty$.

local policy iteration algorithm. From Figs. 5 and 6, it is obvious that the last converged system rotor speed trajectory (also known as the system state trajectory) using the proposed algorithm is able to effectively track the given speed target curve.

Now, if we let $B_x^i \equiv \Omega, i = 0, 1, \ldots$, then the local policy iteration ADP algorithm will become conventional global policy iteration algorithm. Here, we refresh the iterative decision rule and iterative value function using the conventional global policy iteration algorithm for 100 times until the computation precision $\epsilon = 0.01$ is satisfied. Fig. 7(a) demonstrates the iterative value functions using the conventional global policy iteration algorithm. From Fig. 7(a), it is shown that the sequence of iterative value functions obtained by the conventional global policy iteration algorithm are monotonically

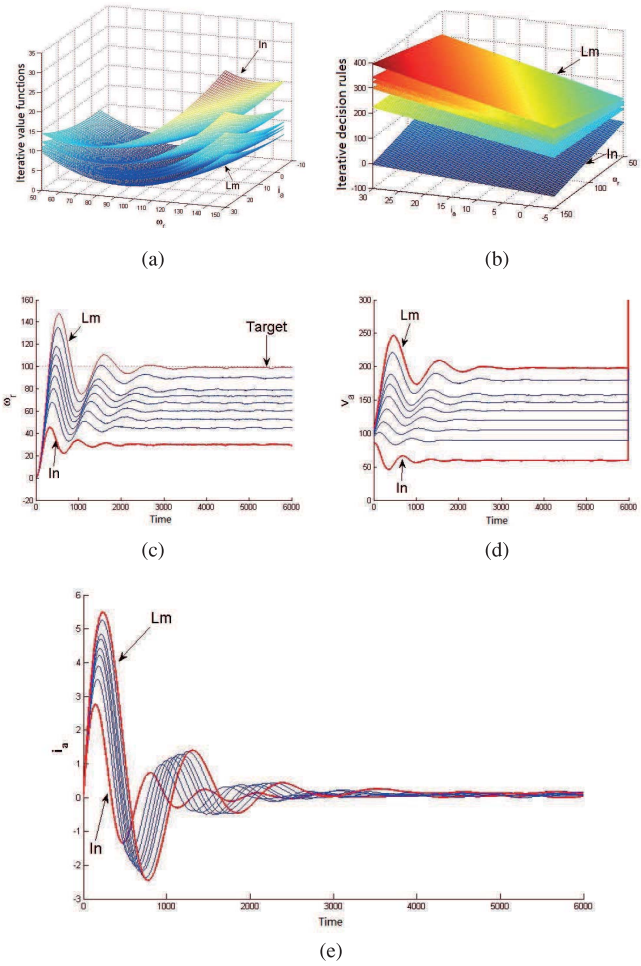

Fig. 7. Traditional policy iteration algorithm. (a) $V_i(x)$ by traditional policy iteration algorithm. (b) $d_i(x)$ by traditional policy iteration algorithm. Iterative trajectories of states: (c) speed, (d) voltage, and (e) current.

nonincreasing and the last converged value function obtained by the conventional global policy iteration algorithm is the optimal performance index function in the global policy space. Compare Fig. 7(a) with Fig. 3, we can see that the last converged value function obtained by the local policy iteration ADP algorithm is also the optimal performance index function in the global policy space. Fig. 7(b) demonstrates the iterative decision rules using the conventional global policy iteration algorithm. Fig. 7(c) and (e) demonstrates the iterative system state trajectories using the conventional global policy iteration algorithm. Fig. 7(d) demonstrates the iterative control trajectories using the conventional global policy iteration algorithm. Hence, if the local spaces used in the proposed algorithm satisfy the condition in Theorem 3, then the sequence of value functions obtained in the proposed algorithm and the sequence obtained in the conventional global policy iteration algorithm will reach the same convergence which is exactly the optimal performance index function in the global policy space of the stochastic processes. However, in the conventional global policy iteration algorithm, we refresh the iterative decision rule and iterative value function for every state data in the global system state simultaneously. Meanwhile, in the local policy iteration ADP algorithm, we only refresh the iterative decision

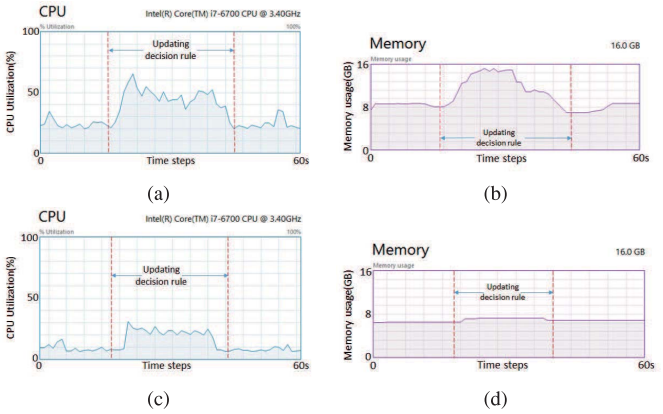

Fig. 8. System load. (a) CPU utilization when system is running conventional policy iteration algorithm for. (b) Memory usage when system is running conventional policy iteration algorithm. (c) CPU utilization when system is running local policy iteration ADP algorithm for. (d) Memory usage when system is running local policy iteration ADP algorithm.

rule and iterative value function in a local space of the system state. This means that the calculation load for the computer could be significantly released using the proposed algorithm.

Fig. 8 demonstrates the system load using conventional policy iteration algorithm and local policy iteration ADP algorithm, respectively. It is shown that the CPU utilization is around 50% and the maximum memory usage is nearly 100% when using the conventional policy iteration algorithm. At the same time, the CPU utilization is around 25% and the memory usage is around 50% when using local policy iteration algorithm.

Therefore, the simulation validates that the proposed algorithm is able to release the massive calculation load for the computer while achieving the goal of finding the optimal performance index function for the stochastic processes.

VI. Conclusion

In this paper, to obtain the optimal performance index function for the stochastic processes, a novel local policy iteration ADP algorithm is presented. We present simulations to validate that the proposed algorithm is able to release the massive calculation load for the computer CPU which is caused by the conventional global policy iteration algorithm while achieving the goal of finding the optimal performance index function for the stochastic processes. However, our proposed algorithm evaluates policies on the basis of expected total discounted reward of the system. When decisions are made frequently, so that the discount factor is very close to 1, or when performance criterion can easily be described, the decision maker may prefer to compare policies on the basis of the average expected reward instead of the expected total discounted reward of the system. Our proposed algorithm may not be suitable for these models. We will focus on this issue in our future work.

References

[1] Y. Wang, H. Shen, H. R. Karimi, and D. Duan, "Dissipativity-based fuzzy integral sliding mode control of continuous-time T–S fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 3, pp. 1164–1176, Jun. 2018.

[2] Y. Wang, Y. Gao, H. R. Karimi, H. Shen, and Z. Fang, "Sliding mode control of fuzzy singularly perturbed systems with application to electric circuit," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 10, pp. 1–9, Oct. 2018.

[3] Y. Wang, Y. Xia, H. Shen, and P. Zhou, "SMC design for robust stabilization of nonlinear Markovian jump singular systems," *IEEE Trans. Autom. Control*, vol. 63, no. 1, pp. 219–224, Jan. 2018.

[4] P. J. Werbos, "Advanced forecasting methods for global crisis warning and models of intelligence," *Gen. Syst. Yearbook*, vol. 22, no. 12, pp. 25–38, 1977.

[5] P. J. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control*, W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA, USA: MIT Press, 1991, pp. 67–95.

[6] D. Wang, H. He, and D. Liu, "Adaptive critic nonlinear robust control: A survey," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3429–3451, Oct. 2017.

[7] D. Wang, C.-X. Mu, and D.-R. Liu, "Data-driven nonlinear near-optimal regulation based on iterative neural dynamic programming," *Acta Automatica Sinica*, vol. 43, no. 3, pp. 366–375, Mar. 2017.

[8] D. Liu, Y. Xu, Q. Wei, and X. Liu. "Residential energy scheduling for variable weather solar energy based on adaptive dynamic programming," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 36–46, Jan. 2018.

[9] D. Wang and C. Mu, "Developing nonlinear adaptive optimal regulators through an improved neural learning mechanism," *Sci. China Inf. Sci.*, vol. 60, no. 5, Nov. 2017, Art. no. 058201.

[10] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA, USA: Athena Sci., 1996.

[11] D. Wang, C. Mu, H. He, and D. Liu, "Event-driven adaptive robust control of nonlinear systems with uncertainties through NDP strategy," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1358–1370, Jul. 2017.

[12] D. Wang, D. Liu, Q. Zhang, and D. Zhao, "Data-based adaptive critic designs for nonlinear robust optimal control with uncertain dynamics," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 11, pp. 1544–1555, Nov. 2016.

[13] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, Sep. 1997.

[14] H. Zhang, C. Qin, and Y. Luo, "Neural-network-based constrained optimal control scheme for discrete-time switched nonlinear system using dual heuristic programming," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 839–849, Jul. 2014.

[15] Q. Wei, F. L. Lewis, Q. Sun, P. Yan, and R. Song, "Discrete-time deterministic Q-learning: A novel convergence analysis," *IEEE Trans. Cybern.*, vol. 47, no. 5, pp. 1224–1237, May 2017.

[16] Z. Ni, H. He, X. Zhong, and D. V. Prokhorov, "Model-free dual heuristic dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1834–1839, Aug. 2015.

[17] Z. Ni, H. He, D. Zhao, X. Xu, and D. V. Prokhorov, "GrDHP: A general utility function representation for dual heuristic dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 614–627, Mar. 2015.

[18] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Syst.*, vol. 32, no. 6, pp. 76–105, Dec. 2012.

[19] Q. Wei, F.-Y. Wang, D. Liu, and X. Yang, "Finite-approximation-errorbased discrete-time iterative adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2820–2833, Dec. 2014.

[20] Q. Wei, D. Liu, and H. Lin, "Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 840–853, Mar. 2016.

[21] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, *Adaptive Dynamic Programming With Applications in Optimal Control*. Cham, Switzerland: Springer, 2017.

[22] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, "Adaptive dynamic programming," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 2, pp. 140–153, May 2002.

[23] Q. Wei and D. Liu, "A novel iterative θ-adaptive dynamic programming for discrete-time nonlinear systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 4, pp. 1176–1190, Oct. 2014.

[24] D. Liu, H. Li, and D. Wang, "Error bounds of adaptive dynamic programming algorithms for solving undiscounted optimal control problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1323–1334, Jun. 2015.

[25] R. Song, F. L. Lewis, Q. Wei, and H. Zhang, "Off-policy actor-critic structure for optimal control of unknown systems with disturbances," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1041–1050, May 2016.

[26] D. Liu, Q. Wei, and P. Yan, "Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 12, pp. 1577–1591, Dec. 2015.
[27] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 1994.
[28] E. Kreyszig, *Introductory Functional Analysis With Applications*. New York, NY, USA: Wiley, 1989.
[29] H. Zhang, L. Cui, and Y. Luo, "Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network ADP," *IEEE Trans. Cybern.*, vol. 43, no. 1, pp. 206–216, Feb. 2013.
[30] D. P. Bertsekas, "Proper policies in infinite-state stochastic shortest path problems," Lab. Inf. Decis. Syst., Massachusetts Inst. Technol., Cambridge, MA, USA, Rep. LIDS-3507, May 2017.
[31] B. Zhou, Y. Cui, and M. Tao, "Optimal dynamic multicast scheduling for cache-enabled content-centric wireless networks," *IEEE Trans. Commun.*, vol. 65, no. 7, pp. 2956–2970, Jul. 2017.
[32] H. Jung and M. Pedram, "Stochastic dynamic thermal management: A Markovian decision-based approach," in *Proc. 25th Int. Conf. Comput. Design*, San Jose, CA, USA, Nov. 2007, pp. 452–457.
[33] L. Li and Z. Sun, "Dynamic energy control for energy efficiency improvement of sustainable manufacturing systems using Markov decision process," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1195–1205, Sep. 2013.
[34] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vols. 1–2. Belmont, MA, USA: Athena Sci., 1995.
[35] D. P. Bertsekas, "Approximate policy iteration: A survey and some new methods," *J. Control Theory Appl.*, vol. 9, no. 3, pp. 310–335, Aug. 2011.
[36] D. P. Bertsekas, "Lambda-policy iteration: A review and a new implementation," Lab. Inf. Decis. Syst., Massachusetts Inst. Technol., Cambridge, MA, USA, Rep. LIDS-P-2874, Oct. 2011.
[37] H. Yu and D. P. Bertsekas, "*Q*-learning and policy iteration algorithms for stochastic shortest path problems," *Ann. Oper. Res.*, vol. 208, no. 1, pp. 95–132, Sep. 2013.
[38] D. P. Bertsekas, M. L. Homer, D. A. Logan, S. D. Patek, and N. R. Sandell, "Missile defense and interceptor allocation by neuro-dynamic programming," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 30, no. 1, pp. 42–51, Jan. 2000.
[39] B. Van Roy, D. P. Bertsekas, Y. Lee, and J. N. Tsitsiklis, "A neuro-dynamic programming approach to retailer inventory management," in *Proc. IEEE Conf. Decis. Control*, San Diego, CA, USA, Aug. 1997, pp. 4052–4057.
[40] H. Zhang, T. Feng, H. Liang, and Y. Luo, "LQR-based optimal distributed cooperative design for linear discrete-time multiagent systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 599–611, Mar. 2017.
[41] H. Jiang and H. Zhang, "Iterative ADP learning algorithms for discrete-time multi-player games," *Artif. Intell. Rev.*, vol. 50, no. 1, pp. 75–91, Jun. 2018.

Mingming Liang received the B.S. degree in automation from the Dalian University of Technology, Dalian, China, in 2015. He is currently pursuing the Ph.D. degree in control theory and control engineering with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

He is also with the University of Chinese Academy of Sciences, Beijing. His current research interests include neural networks, reinforcement learning, and stochastic processes.

Ding Wang (M'15) received the B.S. degree in mathematics from the Zhengzhou University of Light Industry, Zhengzhou, China, in 2007, the M.S. degree in operations research and cybernetics from Northeastern University, Shenyang, China, in 2009, and the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2012.

From 2015 to 2017, he was a Visiting Scholar with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, USA. He was an Associate Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. He is currently a Professor with the Faculty of Information Technology, Beijing University of Technology, Beijing. He has authored or coauthored over 120 journal and conference papers and three monographs. His current research interests include adaptive and learning systems, computational intelligence, and intelligent control.

Dr. Wang was a recipient of the Excellent Doctoral Dissertation Award of Chinese Academy of Sciences in 2013, and a nomination of the Excellent Doctoral Dissertation Award of Chinese Association of Automation in 2014. He was selected for the Young Elite Scientists Sponsorship Program by the China Association for Science and Technology in 2017, and also selected for the Youth Innovation Promotion Association of the Chinese Academy of Sciences in 2018. He was the Finance Chair of the 12th World Congress on Intelligent Control and Automation in 2016, and the Publications Chair of the 24th International Conference on Neural Information Processing in 2017. He currently or formerly serves as an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Neurocomputing*, and *Acta Automatica Sinica*.

Derong Liu (S'91–M'94–SM'96–F'05) received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, USA, in 1994.

From 1993 to 1995, he was a Staff Fellow with the General Motors Research and Development Center, Detroit, MI, USA. From 1995 to 1999, he was an Assistant Professor with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, USA. He joined the University of Illinois at Chicago, Chicago, IL, USA, in 1999, where he became a Full Professor of Electrical and Computer Engineering and Computer Science in 2006. He was the Associate Director with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. He has authored or coauthored over 19 books.

Dr. Liu was a recipient of the Faculty Early Career Development Award from the National Science Foundation in 1999, the University Scholar Award from University of Illinois from 2006 to 2009, the Overseas Outstanding Young Scholar Award from the National Natural Science Foundation of China in 2008, and the Outstanding Achievement Award from Asia–Pacific Neural Network Assembly in 2014. He was selected for the 100 Talents Program by the Chinese Academy of Sciences in 2008. He is the Editor-in-Chief of *Artificial Intelligence Review* (Springer). From 2010 to 2015, he was the Editor-in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS. He is a fellow of the International Neural Network Society and the International Association of Pattern Recognition.