https://doi.org/10.26599/AIR.2022.9150007 CAAI Artificial Intelligence Research

State of the Art of Adaptive Dynamic Programming and Reinforcement Learning

Derong Liu[1,2] ✉, Mingming Ha[3], and Shan Xue[4]

ABSTRACT

This article introduces the state-of-the-art development of adaptive dynamic programming and reinforcement learning (ADPRL). First, algorithms in reinforcement learning (RL) are introduced and their roots in dynamic programming are illustrated. Adaptive dynamic programming (ADP) is then introduced following a brief discussion of dynamic programming. Researchers in ADP and RL have enjoyed the fast developments of the past decade from algorithms, to convergence and optimality analyses, and to stability results. Several key steps in the recent theoretical developments of ADPRL are mentioned with some future perspectives. In particular, convergence and optimality results of value iteration and policy iteration are reviewed, followed by an introduction to the most recent results on stability analysis of value iteration algorithms.

KEYWORDS

adaptive dynamic programming; approximate dynamic programming; adaptive critic designs; neuro-dynamic programming; neural dynamic programming; reinforcement learning; intelligent control; learning control; optimal control

Reinforcement learning (RL) methods are a kind of goal-oriented learning approaches, which allow the agent to interact with the environment and obtain the corresponding rewards. The objective of RL methods is to maximize long-term cumulative return. In the past few decades, RL has enjoyed rather remarkable successes across a wide range of domains, involving game artificial intelligence (AI)[1–6], COVID-19 border testing[7], mobile robotics[8–10], autonomous driving[11,12], nuclear fusion[13], intelligent control[14–22], and so forth, which make the agents possess the strikingly successful machine intelligence previously thought to be impossible. Therefore, the term of RL has a broad coverage in areas such as psychology, computer science, economics, and control community[23,24].

In various game AI tasks, board games are the most representative kind of multistage decision task, where the ancient Chinese board game Go is most challenging. It has been intensively studied by AI researchers for many decades. Researchers have always hoped to develop a learning agent, which can defeat the human professional opponent in the game Go. The board of the game consists of a grid of 19 horizontal and 19 vertical lines. Black and white "stones" are alternately placed on unoccupied cross point by two players. The goal of Go is to occupy an area larger than that occupied by the other player. At the beginning of Go, there are roughly 360 options for each of the two players to place their stones. Since this game has a large number of potential board positions, the search space for Go grows exponentially and the number of legal moves per position quickly becomes larger than the total number of atoms in the whole universe[25]. With this many states result in so many outcomes any given game can move in, it is impossible for programs ever to experience more than a small fraction of them even if we have massive computing power.

The Monte Carlo tree search approach has been used often to solve the single-agent sequential decision problems. For computer Go, only some of the possible sequences at each step are sampled and the agent chooses an appropriate move between different possible moves rather than trying by brute force computation of every possible ones.

An AI company in London, namely Google DeepMind, has achieved remarkable results in applying RL techniques. In March 2016, the match of a program called AlphaGo, developed by DeepMind, vs. Lee Sedol made worldwide headline news at that time, and has been a milestone in the quest of AI. The defeat over a human opponent by a machine has also aroused huge public interests in AI technology around the world[26]. Instead of searching various sequences of moves to learn, AlphaGo makes a move by evaluating the value of the current position on the board. Such an evaluation of the current state was made possible by combining the deep learning capabilities of neural networks (NNs). Position evaluation plays a crucial role in the success of AlphaGo. It is used to estimate the optimal cost function. Such ideas have been applied usefully to computer games by many researchers, such as backgammon (TD-Gammon)[27,28], checkers[29], othello[30], and chess[31]. Many of these computer games involve the use of an RL technique called temporal-difference (TD) method, in particular, the TD(λ) method which was used in AlphaGo and TD-Gammon to evaluate the current position. An agent trained by TD-Gammon has possessed a grandmaster level in the backgammon[27,28]. Moreover, AlphaGo first defeated European Go champion Fan Hui (professional 2 dan) by 5 games to 0[2], then won world Go champion Lee Sedol (professional 9-dan) by 4 games to 1[26,32], and defeated world's no. 1 Go player Ke Jie

1 Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen 518055, China
2 Department of Electrical and Computer Engineering, University of Illinois at Chicago, IL 606071, USA
3 School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China
4 School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
Address correspondence to Derong Liu, derongliu@gmail.com

(professional 9-dan) by 3 games to 0[33,34].

The success of RL depended on NN's powerful representation capabilities[1,35–37]. The deep NN structure used in AlphaGo has 13 layers. Even though there were a large number of reports on the application of RL and related approaches for Go[38–42], it is only with AlphaGo[2] that deep NNs were adopted to establish the value networks to achieve high evaluation accuracy. Position evaluation[39,40,43,44] and deep learning[45,46] have been applied to programs to play the game of Go, however, none of them realized the level of success by AlphaGo[2]. The success of AlphaGo has a far-reaching impact on the research in AI. Based on the core structure of AlphaGo, AlphaGo Zero is only given the game rules in the sense that it learns from self-play, which has defeated AlphaGo by 100 games to 0. Afterwards, AlphaGo Zero was generalized into a single AlphaZero algorithm[4], which based solely on RL, without human data, guidance, or domain knowledge beyond game rules. In various challenging tasks such as chess and shogi (Japanese chess), as well as Go, AlphaZero can achieve a superhuman performance. The tasks of single-agent environments and two-player turn-based games have been intensively investigated by researchers. For the case containing multiple agents, different agents independently learn and act to cooperate and compete with each other. For example, it is demonstrated that the trained agent can achieve human-level and even grandmaster-level performance in some challenging three-dimensional multiplayer video games[5,6]. More exciting result is the accelerating fusion science through learned plasma control[13], which reveals that the RL technique possesses the potential to accelerate scientific and technological progress.

RL is widely considered as an effective technique in handling optimization problems by applying the principle of optimality derived from dynamic programming (DP). Especially, RL is used often in optimal control problems in control systems community. Significantly, DP provides a foundation for understanding RL. The great majority of RL algorithms can be considered as attempts to achieve the similar effect as DP, with less computation and without establishing a sufficiently accurate environment model. One class of RL methods is built upon the actor-critic structure, namely adaptive critic designs[47], where a critic component is used to evaluate the value of the current state and adopted action while an actor component applies an action or control policy to the environment. In the vast majority of control systems, since the state and control input spaces are continuous, it is necessary to introduce the function approximation techniques. The combination of DP, function approximators, and actor-critic structure results in the adaptive dynamic programming (ADP) algorithms.

Although both RL and ADP provide approximate solutions to DP with similar ideas and they have close relationship with each other, studies in these two directions have been somewhat independent[48] in the past. It has been a recent trend to regard the two together as ADPRL (ADP and RL)[49–51]. In this paper, starting from the basic Markov decision process, the classic RL algorithms are revisited. The connections and differences between Markov decision process and optimal control for discrete-time nonlinear systems are summarized. As mentioned in Ref. [49], RL is strongly connected from a theoretical point of view with direct and indirect adaptive optimal control methods. Moreover, compared with existing surveys[49,52–55], several key steps in the recent algorithm schemes and theoretical developments of ADPRL for discrete-time optimal control are mentioned with some new perspectives, which includes some new theoretical results of convergence rate, stability, and new iterative framework. A brief overview of RL will be provided in the next section, followed by a more detailed overview of ADP. Its classic frameworks and iterative schemes are also revisited.

1 Reinforcement Learning

RL methods mainly involve the value-based algorithms and the policy-based algorithms, where the representative algorithms of the value-based approaches include policy iteration (PI), value iteration (VI), TD learning, deep Q-network and so forth while the classic policy-based methods involve stochastic and deterministic policy gradient methods, REINFORCE, trust region policy optimization, proximal policy optimization, etc. Note that actor-critic methods belong to both value-based method and policy-based method, which combine the advantages of policy evaluation and policy gradient. The classic research results in RL can be found in the book by Sutton and Barto[47] and its references. The central idea of the value-based method in RL is certainly the TD method[47,56]. The typical algorithms of TD learning are the on-policy※ Sarsa and the off-policy† Q-learning[57,58]. The area of RL is more complete and more mature[59,60] than ADP.

In a typical RL problem, an agent sequentially takes an action to interact with its environment. In general, this process is formally modeled as Markov decision process. An RL system typically consists of the following four components: $\{\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R}, F\}$, where $\boldsymbol{S}$, $\boldsymbol{A}$, $\boldsymbol{R}$, and F: $\boldsymbol{S}\times\boldsymbol{A}\to\boldsymbol{S}$ are the set of states, the set of actions, the set of scalar rewards, and the state transition function (or probability), respectively. A policy is denoted as π, which represents a mapping π: $\boldsymbol{S}\to\boldsymbol{A}$. At any given time t, the agent in current state $s_t\in\boldsymbol{S}$ takes an action $a_t\in\boldsymbol{A}$ accordings to a deterministic or stochastic policy π, i.e., $a_t=\pi(s_t)$ or $\Pr(a_t|s_t)=\pi(a_t|s_t)$, transitions to the next state s' according to the state transition function $s_{t+1}=F(s_t,a_t)$ or probability $\Pr(s_{t+1}=s'|s_t=s,a_t=a)$, and at the same time, receives a reinforcement signal, which is also called the immediate reward, denoted by $r_{t+1}=r(s_t,a_t,s_{t+1})\in\boldsymbol{R}$. The goal of RL is to find a policy to maximize the discounted accumulated reward, namely the total return denoted by $G_t=\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}$, starting from an initial state s_0. The corresponding policy is the optimal policy.

Value-based RL methods always involve estimating some kind of value functions. Under a policy π, a value function estimates the value of a given state s, which is formulated as

$$V^{\pi}(s)=\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\big|_{s_t=s}=\sum_{k=0}^{\infty}\gamma^k r(s_{t+k},a_{t+k},s_{t+k+1})\big|_{s_t=s},$$

where $0<\gamma\leqslant 1$ is a discount factor, $a_k=\pi(s_k)$, and $s_{k+1}=F(s_k,a_k)$. $V^{\pi}(s)$ is called the state-value function for policy π. Another value function for policy π is often employed to evaluate the value of taking action a in a given state s, namely the action-value function, which is defined as

$$Q^{\pi}(s,a)=\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\big|_{s_t=s,a_t=a}=\sum_{k=0}^{\infty}\gamma^k r(s_{t+k},a_{t+k},s_{t+k+1})\big|_{s_t=s,a_t=a}.$$

In different RL tasks, the optimal policy denoted by π^* may not be unique. There might exist multiple optimal policies to make the accumulated reward achieve maximization. Their same optimal state-value function is given by

※On-policy learning estimates the value of a policy while using it for control.

† Off-policy learning is defined as evaluating one policy while following another.

$$V^*(s) = \sup_{\pi} \{V^{\pi}(s)\} .$$

And the same optimal action-value function is expressed as

$$Q^*(s,a) = \sup_{\pi} \{Q^{\pi}(s,a)\}.$$

According to the Bellman's principle of optimality, the state-value and action-value functions can be rewritten as their corresponding Bellman optimality equations (Bellman equations), respectively, given as follows:

$$V^*(s) = \max_a \{r(s,a,s') + \gamma V^*(s')\} \tag{1}$$

and

$$Q^*(s,a) = r(s,a,s') + \gamma \max_{a'} \{Q^*(s',a')\} \tag{2}$$

The optimal policy is determined by

$$\pi^*(s) = \arg\max_a \{r(s,a,s') + \gamma V^*(s')\}$$

or

$$\pi^*(s) = \arg\max_a Q^*(s,a).$$

Note that the above is described for deterministic systems.

In what follows, how to obtain the state-value function V^{π} for an arbitrary policy π is presented in detail. The process of estimating the value function of π is called policy evaluation, which is expressed as

$$\begin{aligned} V^{\pi}(s) =& r(s,a,s') + \gamma V^{\pi}(s') = \\ & r(s,\pi(s),F(s,\pi(s))) + \gamma V^{\pi}(F(s,\pi(s))) \end{aligned} \tag{3}$$

The goal is to solve for V^{π} from Eq. (3), i.e., to determine a function $V^{\pi}(\cdot)$ that can balance the above equation for all s. Initialized by an arbitrary initial estimation $V_0(\cdot)$, the value function can also be computed by iterative policy evaluation:

$$V_{i+1}(s) = r(s,a,s') + \gamma V_i(s') \tag{4}$$

where $i = 0,1,2,\ldots$ is the iteration index. The value function $V^{\pi}(s)$ is obtained as $V^{\pi}(s) = \lim_{i\to\infty} V_i(s)$, $\forall s$, under the assumption that the iteration in Eq. (4) is convergent.

After the policy evaluation is finished and $V^{\pi}(s)$ is obtained, an improved policy can be computed by

$$\pi'(s) = \arg\max_a \{r(s,a,s') + \gamma V^{\pi}(s')\} \tag{5}$$

The PI procedure involves the alternating iteration between policy evaluation in Eqs. (3) and (4) and policy improvement in Eq. (5). With this operation, the optimal policy can be determined.

Another algorithm to obtain the optimal policy is to use the VI scheme. For each iteration step i, the state-value functions V_i can be computed by the following value function update:

$$V_{i+1}(s) = \max_a \{r(s,a,s') + \gamma V_i(s')\} \tag{6}$$

and policy improvement

$$\pi_{i+1}(s) = \arg\max_a \{r(s,a,s') + \gamma V_{i+1}(s')\} \tag{7}$$

The iteration process shall continue until V_i converges. In general, the termination criterion $|V_{i+1}(s) - V_i(s)| \leqslant \varepsilon$, $\forall s$, is used to stop the iteration process, where ε is a small positive number. By doing this, the obtained state-value function satisfies $V_{i+1}(s) \approx V^*(s)$. Then, we have $\pi^*(s) \approx \pi_{i+1}(s)$.

Note that the PI and VI algorithms are usually model-based approaches. For the unknown environment's transition dynamics, the TD method[56] is a remarkable success to estimate the value function. The TD algorithm is given by

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right] \tag{8}$$

or

$$V_{i+1}(s_t) = V_i(s_t) + \alpha\left[r_{t+1} + \gamma V_i(s_{t+1}) - V_i(s_t)\right] \tag{9}$$

where $\alpha > 0$ is the step size. Actually, compared with TD(λ) introduced later, the algorithm described in Formula (8) and Eq. (9) is also called TD(0). Note that the update rule in Formula (8) is described as the following general formula:

$$\text{NewValue} \leftarrow \text{OldValue} + \text{StepSize} \times (\text{Target} - \text{OldValue})$$

which means a step of move towards the "Target".

In addition, the off-policy TD algorithm, Q-learning[57,58], is an early breakthrough of the RL methods[47]. A common on-policy version of TD method is called Sarsa[47,61,62], whose name comes from the fact that the approach employs the quintuple $\{s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}\}$ which was first introduced by Rummery and Niranjan [61], and called modified Q-learning.

TD(λ), a more general TD approach[56], has been very popular[2,27–31,39,40,44]. The idea of eligibility trace can be applied to Sarsa[47,61] to generate a new RL algorithm, namely Sarsa(λ). The eligibility trace can also be applied to Q-learning[57,58].

TD, Q-learning, Sarsa, TD(λ), Sarsa(λ), and Q(λ) estimate the value functions $V(s)$ or $Q(s,a)$ by using the state or action-state trajectories derived from the environment. After obtaining the estimation of the value function, an improved policy can be determined. This is the procedure how RL solves the Bellman equation (Eqs. (1) and (2)) with model-based and model-free approaches to obtain approximate solutions.

2 Dynamic Programming for Discrete-Time Nonlinear Systems

DP method is a classical optimization technique for multistage decision problems. The traditional DP approaches require a complete environment information. There are various schemes of DP[63–66] to handle different systems, such as linear systems or nonlinear systems, discrete-time systems or continuous-time systems, deterministic systems or stochastic systems, time-invariant systems or time-varying systems, etc. The simplest scheme is a backward computation in time. Since the digital implementation of manufacturing systems has become a large trend in the industrialization process and time-invariant nonlinear systems cover most of the application scenarios, in the following discussion, we focus attention on discrete-time nonlinear time-invariant dynamical systems.

Consider the following discrete-time nonlinear systems given by

$$x_{k+1} = F(x_k, u_k) \tag{10}$$

where $k = 0,1,2,\ldots$, $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ and F: $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are the discrete-time index, the state vector, the control vector, and the system function, respectively. Under a given initial state $x_0 \in \mathbb{R}^n$, an infinite-length control sequence $\underline{u}_0 = (u_0, u_1, \ldots)$ needs to be selected and applied to Eq. (10) to achieve certain objectives.

Define the performance index (or cost) of Eq. (10) as

$$J(x_k, \underline{u}_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, u_i) \tag{11}$$

where $\underline{u}_k$ indicates the control input sequence starting at time k, $U(\cdot,\cdot)$ is the positive definite utility function, and $\gamma \in (0,1]$ denotes the discount factor. The function J in Eq. (11) is the cost-to-go of state x_k under the control input sequence $\underline{u}_k$, which is determined by the initial time and the initial state. In this case, the cost will accumulate indefinitely. This problem is called the infinite horizon problem. If the cost accumulates over finite time steps, the corresponding problem is the finite horizon problem. In general, the objective of DP is to determine $\underline{u}_0 = (u_0, u_1, \dots)$ so that $J(x_0, \underline{u}_0)$ is maximized or minimized. In this article, $\underline{u}_0^* = (u_0^*, u_1^*, \dots)$ and $J^*(x_0)$ are used to represent the optimal control sequence and the optimal cost function, respectively.

More specifically, in this article, we consider the minimization problem of the cost function $J(x_0, \underline{u}_0)$ in the sense that a control sequence $\underline{u}_0^*$ is determined to minimize the function J in Eq. (11). According to the definition of the minimization problem, the optimal cost function is expressed as

$$J^*(x_0) = \inf_{\underline{u}_0} J(x_0, \underline{u}_0) = J(x_0, \underline{u}_0^*).$$

The control input may be considered as a function of the state, i.e., $u_k = \mu(x_k)$, $\forall k$. Such a mapping $\mu\colon \mathbb{R}^n \to \mathbb{R}^m$ is called control policy, or state feedback control, or control law. Therefore, the cost function in Eq. (11) for a given control policy μ can be rewritten as

$$J^{\mu}(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, \mu(x_i)).$$

The corresponding optimal cost starting at x_0 is obtained as

$$J^*(x_0) = \inf_{\mu} J^{\mu}(x_0) = J^{\mu^*}(x_0)$$

where μ^* is the optimal policy.

The Bellman's principle of optimality[63–66] is the foundation of DP method. It shows that no matter what previous decisions are, the remaining actions must constitute an optimal policy regarding the state resulting from those previous actions.

It is assumed that, for all possible states x_{k+1}, the optimal cost $J^*(x_{k+1})$ and the optimal control sequences $\underline{u}_{k+1}^*$ from time $k+1$ to the terminal time have been obtained. Note that the sequence $\underline{u}_{k+1}^*$ is determined by x_{k+1}. In this case, $J^*(x_{k+1})$ indicates that the optimal cost is generated by applying the optimal control sequence $\underline{u}_{k+1}^*$ to the system with initial state x_{k+1}. If an arbitrary control input u_k acts on the system at time k and then the optimal input sequence $\underline{u}_{k+1}^*$ is applied to the system from time $k+1$, the accumulated cost becomes

$$U(x_k, u_k) + \gamma U(x_{k+1}, u_{k+1}^*) + \gamma^2 U(x_{k+2}, u_{k+2}^*) + \cdots = U(x_k, u_k) + \gamma J^*(x_{k+1})$$

where $x_{k+1} = F(x_k, u_k)$. Therefore, the optimal cost from time k on is expressed as

$$J^*(x_k) = \min_{u_k}\{U(x_k, u_k) + \gamma J^*(x_{k+1})\} = \min_{u_k}\{U(x_k, u_k) + \gamma J^*(F(x_k, u_k))\} \tag{12}$$

At time k, the optimal control u_k^* is the control input that minimizes the cost, i.e.,

$$u_k^* = \arg\min_{u_k}\{U(x_k, u_k) + \gamma J^*(x_{k+1})\} \tag{13}$$

The principle of optimality for Eq. (10) is formulated as Eq. (12). Its key point lies in the fact that only one control vector is optimized at a time by the backward numerical process.

Considering the Markov decision process mentioned in Section 1, it is the basic task in sequential decision problems. Considerable research in RL has focused on this problem or its variants. For discrete-time nonlinear systems in Eq. (10), it can be observed that each state depends only on the state and control attained in the previous time step, which is similar to the Markov decision process. Therefore, the core idea of RL is applicable to solve the Bellman optimality equations (Eq. (1)) and the optimal cost (Eq. (12)). On the other hand, the main difference between the Markov decision process and discrete-time nonlinear system is that, in general, the Markov decision process is stochastic process while the discrete-time nonlinear system is a deterministic dynamics. More specifically, the definitions of some terms and functions corresponding the Markov decision process and discrete-time nonlinear system are different, which are summarized in Table 1.

Remark 1. Table 1 illustrates the different terms and definitions corresponding to Markov decision processes and optimal control for discrete-time nonlinear systems, where the stochastic case of Markov decision processes is considered. In this case, both state space and action space are discrete and the numbers of the state and action are finite, which is the classic problem formulation of early game AI. As the environment becomes more complex, the number of the state increases exponentially, such as the game Go, real-time strategy video game StarCraft II and so on. In the computer science community, since boundaries of tasks are clear, such as the boundaries of the board and video game, these tasks allow that the agent interacts with the dynamics to accumulate experience in the trial and error and improve the policy with a lower cost. Therefore, convergence of RL approaches is a key focus in the computer science community. By contrast, for discrete-time nonlinear dynamics, the system state and control spaces are continuous. Besides, in control systems community, considerable system dynamics are derived from the real world, which results in the huge cost of trial and error, such as the orbital maneuver problem, aircraft attitude control, robot control and navigation, and so forth. Therefore, for control policies deployed to the actual control systems, both convergence and stability are necessary and important topics in control systems community.

DP method is a quite effective technique for optimization and optimal control problems. Especially, whether or not the control and state variables are constrained, it is convenient to apply the DP method to nonlinear system dynamics. Note that Eq. (12) is called the Bellman equation or functional equation of DP. It is the

Table 1 Comparison between Markov decision process and optimal control for discrete-time nonlinear systems.

Term	Markov decision process	Discrete-time nonlinear system
Dynamics	$\Pr(a_t \mid s_t) = \pi(a_t \mid s_t)$	$x_{k+1} = F(x_k, u_k)$
Immediate reward/Utility function	$r_{t+1} = r(s_t, a_t, s_{t+1})$	$U(x_k, u_k)$
Return/Performance index	$G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_{i+1}$	$J(x_k, \underline{u}_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, u_i)$
Policy/Feedback control	$\pi(a \mid s)$	$\mu(x)$

foundation for implementing DP. Considering Eq. (12), if we have obtained the system function F and the cost function J, solving the Bellman equation can be regarded as a simple optimization problem. On the other hand, since the solutions of the Bellman equation requires calculations backward in time, actually, it is often impossible to implement the exact DP. The backward numerical process easily leads to the well-known "curse of dimensionality"[63–66]. Therefore, in general, the Bellman equation is difficult to solve analytically. In the past few decades, researchers have strived to obtain the numerical solution of the Bellman equation by establishing a module named critic. The critic component is employed to approximate the cost function. It acts as a function approximator.

3 Basic Frameworks of Adaptive Dynamic Programming

In 1975, Weinstein and Zeckhauser[67] probably first mentioned the term "adaptive dynamic programming", which was employed to design the optimal scheme for consuming natural resources. Researchers have realized in very early days that DP has been considered as a quite effective method to solve inventory control problems. Nevertheless, it is difficult to exactly solve the DP problems. Afterwards, ADP methods[68,69] were formally considered to solve the inventory control problem. In 1976, Shields'[70] survey of fault detection had also mentioned the ADP technique. After nearly twenty years, an algorithm closely related to ADP was developed by Barto et al.[71] Subsequently, in 2002, aiming at continuous-time systems, the ADP method was adopted to handle the optimal control problem. Then, a comprehensive theoretical analysis[72] of the developed ADP approach was given later. Therefore, numerical solutions were sought using "adaptive dynamic programming"[68,69] or using "approximate dynamic programming" scheme[73]. In 1987, the DP-based technique for optimal control was named "approximate dynamic programming" by Werbos[74].

Back in 1977, an approach, later called "adaptive critic designs" (ACD)[75–80], was presented by Werbos. Later, ACD frameworks were classified into three classes[76]: heuristic dynamic programming (HDP), dual heuristic programming (DHP), and globalized DHP (GDHP). In Ref. [78] published in 1992, "ACD", "approximate dynamic programming", and "RL" have been alternatively adopted by Werbos. The term ACDs have since appeared in many studies [38, 41, 48, 74, 76–88].

On the other hand, another widely used term is "neural dynamic programming"[89–91] or "neuro-dynamic programming"[92, 93]. Their corresponding acronym is "NDP". In terms of control applications[89–97], both NDP and ADP have been widely adopted to represent approximate approaches for DP problems. Bertsekas and Tsitsiklis's book[93] is the first one in NDP/ADP, in which the authors first systematically and comprehensively illustrated the methodology for intelligent control, optimal control, and operations research.

Since it is difficult to compute the exact solution of DP problems, in general, theoretical solutions to optimal control problems for nonlinear plants are approximated by using NDP/ADP methods. Therefore, the obtained approximate optimal control is usually called "suboptimal control" in the literature. Some papers studying optimal control problems[98–102] can be found, in which many viable NDP/ADP-based algorithms were developed to approximate the exact solution of the optimal control problem.

Here, ADP or ADPRL will be adopted to represent "approximate dynamic programming", "adaptive dynamic programming", "adaptive critic designs", "neuro-dynamic programming", "neural dynamic programming", as well as "reinforcement learning"[103–106]. No matter what we call it, in all terms, the objective is to obtain the approximate solutions of DP. For this reason, the term "approximate dynamic programming" has been quite popular in the past. A classic framework of ADPRL is designed as a structure including three modules, namely model, critic, and action[76,78], as given in Fig. 1. The critic component plays the role of estimating the cost function J, for some deterministic plants, which can be generally regarded as a Lyapunov function.

This paper concentrates on the case of each module with the NN structure[107–109]. As shown in Fig. 1, the output of the critic module in the ADP structure is the estimate of the function J in Eq. (11), denoted as $\hat{J}$. The procedure is implemented by minimizing the following square error over time.

$$\|E_h\| = \frac{1}{2}\sum_k E_k^2 = \frac{1}{2}\sum_k \left(\hat{J}_k - U_k - \gamma \hat{J}_{k+1}\right)^2 \tag{14}$$

where $\hat{J}_k = \hat{J}(x_k, W_c)$ and W_c indicates the parameters of the critic network, i.e., the function approximator of state-value function. The function U_k is the same utility function as the one in Eq. (11). Note that the function U_k given in Eq. (11) is usually a function of x_k and u_k, i.e., $U_k = U(x_k, u_k)$. If, for $\forall k$, $E_k = 0$, Eq. (14) results in

$$\hat{J}_k = U_k + \gamma \hat{J}_{k+1} = U_k + \gamma(U_{k+1} + \gamma \hat{J}_{k+2}) = \cdots = \sum_{i=k}^{\infty} \gamma^{i-k} U_i \tag{15}$$

which is equivalent to the cost function in Eq. (11). Obviously, in order to minimize the error function in Eq. (14), the critic NN is established so that its output $\hat{J}$ approximates the cost function J given in Eq. (11).

In the case where the system function in Eq. (10) is unknown, the function F given in Eq. (10) is learned by a model network as plotted in Fig. 1. The model network can be constructed and trained off-line[78,80] in advance or can also be identified in parallel with the training of critic and action networks[110]. Note that the model network is trained by minimizing $\|x_k - \hat{x}_k\|$.

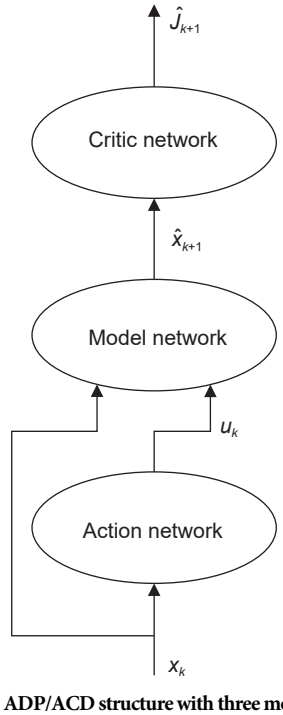

Fig. 1 ADP/ACD structure with three modules.

As mentioned in the previous paragraph, the learning process of the critic network is to realize the minimization of the error measure in Eq. (14). There are various standard NN training algorithms[111,112] to achieve this goal. As depicted in Fig. 1, $\hat{J}_{k+1} = \hat{J}(\hat{x}_{k+1}, W_c)$ is the estimation of the cost function J at time $k+1$. Note that the state $\hat{x}_{k+1}$ is a prediction of the state at time $k+1$ derived from the model network rather than a real state generated by the system dynamics.

Once the approximation of the cost function has been finished, the action network is employed to minimize $U_k + \gamma \hat{J}_{k+1}$ through the use of $u_k = u(x_k, W_a)$ while fixing the parameters of the critic and model networks, where W_a is the weights of the action network. Then, the learned action network will output an optimal, or at least, a suboptimal control signal. If the trained critic network possesses an ideal performance, the control signal will be close to the optimal control input in the sense that the performance of the action network is determined by the approximation ability of the critic network. The objective of DP is to determine an optimal action sequence. The core idea is to interactively establish a relationship between current actions and future consequences by an approximation of the function J.

After the action network's training process is complete, one may check its control performance, then determine whether or not to go back to the training cycle of the critic network again[79,80]. This loop will be iteratively performed until an acceptable performance is achieved. The interaction among the three networks is presented in Fig. 1. Recall that RL method is goal-directed learning by interacting with the environment or systems. Therefore, the control signal u_k generated by the action network is used to interact with the external environment or the target plants and then a state at next time x_{k+1} is obtained. Meanwhile, the approximate state $\hat{x}_{k+1}$ is given by the model network.

In general, the gradient descent algorithm is applied to train the three networks. The gradient information is propagated backward through the critic network to the model network and then to the action network. Then, the three networks can be regarded as one large feedforward network (Fig. 1). In the present implementation of ADP as given in Fig. 1, the model network is required to identify the unknown system dynamics. Even if the system function is known, the model network is still necessary to further facilitate the gradient backpropagation of action network.

Two algorithms[113] used to train the critic network were introduced by Liu et al., where one is a forward-in-time approach as presented in Fig. 2, and the other is a backward-in-time approach as given in Fig. 3. In the forward-in-time algorithm, $\hat{J}_k$ is the output of the critic network and $U_k + \gamma \hat{J}_{k+1}$ is the training target of the critic network. In Eq. (14), $\hat{J}_k$ and $\hat{J}_{k+1}$ represent the cost of different states at time k and $k+1$. On the other hand, in the backward-in-time approach, $\hat{J}_{k+1}$ in Eq. (14) is the output of the critic network and $(\hat{J}_k - U_k)/\gamma$ is chosen as the training target. In a word, both forward-in-time and backward-in-time schemes strive to achieve the minimization of the error measure in Eq. (14) and satisfy the requirement in Eq. (15).

According to the TD learning in Formula (8), $r_{t+1} + \gamma V(s_{t+1})$ is the learning target, also called the TD target. Then, the learning objective of TD method is to minimize $|r_{t+1} + \gamma V(s_{t+1}) - V(s_t)|$, which is identical to the key idea of the forward-in-time approach shown in Fig. 2. The main difference between the TD learning and the forward-in-time approach is the definition of reward function. The reward of TD learning is defined as $r_{t+1} = r(s_t, a_t, s_{t+1})$ while the reward of forward-in-time scheme, i.e., the utility function, is expressed as $U_k = U(x_k, u_k)$. The reason behind the difference between r_{t+1} and U_k will be discussed later.

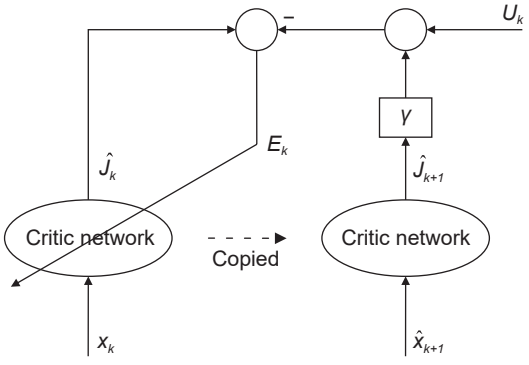

Fig. 2 Forward-in-time approach for critic network learning.

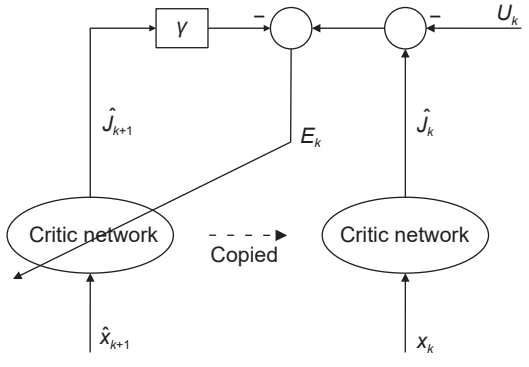

Fig. 3 Backward-in-time approach for critic network learning.

Note that the TD(λ) and the forward-in-time scheme depicted in Fig. 2 possess the same learning objective. Nevertheless, in TD and TD(λ), the update of value functions at each step only makes a move according to the step size towards the target, and presumably, it does not reach the target after a single step of move. In the present approaches shown in Figs. 2 and 3, the forward-in-time and backward-in-time states for certain number of steps, such as 3–5 steps[114] or 50 steps[115], are employed in learning or training. With this operation, the learning target may or may not be achieved. It is certain that the action taken by an agent will move in the direction of the target.

There were two greatest developments of ADP in control systems community[80,115]. The first one[80] summarized the main developments of ADP in detail. Until then, major results of ADP methods are mainly contributed by Werbos[74–78]. The second one[115] achieved important advances regarding the model-free ADP method. A more concise scheme was developed to make the model network given in Fig. 1 not required anymore. The improved ADP-based algorithm was applied to several practical examples[115], and the competitive performance of the developed ADP approach was demonstrated. As mentioned in Ref. [115], the present self-learning approach can be regarded as a backward-in-time algorithm. Another paper[113] also discussed the model-free ADP approach and its properties. As plotted in Fig. 4, the present diagram is a model-free, action-dependent adaptive critic design. In this design, the model network and the critic network together can be regarded as a new critic network.

The model-free ADP has been called action-dependent (AD) ACDs by Werbos[78]. Therefore, various AD versions of HDP, DHP, and GDHP have emerged to deal with the model-free control problem. The classic approach of ADP, namely HDP, is given in Fig. 1 or the left side of Fig. 4. As mentioned in Ref. [78], the learning objectives of HDP and TD are identical. Also, the equivalence of ADHDP and Q-learning was discussed[78]. The

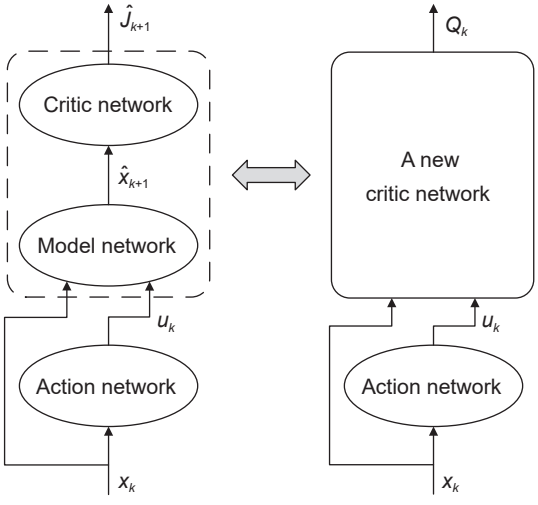

Fig. 4 A new critic network for model-free ADP.

action-value function is adopted in both ADHDP and Q-learning (involving Sarsa as well) to evaluate the value of state-action pair. As depicted in the right side of Fig. 4, the role of the critic network of ADHDP is to minimize the following square error measure.

$$\|E_q\| = \frac{1}{2}\sum_k E_{qk}^2 = \frac{1}{2}\sum_k \left(Q_{k-1} - U_k - \gamma Q_k\right)^2 \tag{16}$$

where $Q_k = Q(x_k, u_k, W_{qc})$ and W_{qc} indicates the parameters of the critic network. If $E_{qk} = 0, \forall k$, Eq. (16) leads to

$$Q_k = U_{k+1} + \gamma Q_{k+1} = U_{k+1} + \gamma(U_{k+2} + \gamma Q_{k+2}) = \cdots = \sum_{i=k+1}^{\infty} \gamma^{i-k-1} U_i \tag{17}$$

which is exactly $\hat{J}_{k+1}$ given in Eq. (15). From Fig. 4, it can be observed that $Q_k = Q(x_k, u_k, W_{qc})$ is equivalent to $\hat{J}_{k+1} = \hat{J}(\hat{x}_{k+1}, W_c) = \hat{J}(\hat{F}(x_k, u_k), W_c)$. If the same inputs x_k and u_k are given, the two outputs are identical. Note that the relationships are different. According to the definition of Q_k and the model network, it is explicitly a function of x_k and u_k while $\hat{F}$ in model network is totally internal. Considering the cost function, obviously, $\hat{J}_{k+1}$ is an explicit function of $\hat{x}_{k+1}$ while, through $\hat{x}_{k+1} = \hat{F}(x_k, u_k)$, $\hat{J}_{k+1}$ is also a function of x_k and u_k.

The one step time difference of functions $\hat{J}_k$ in Eq. (15) and Q_k in Eq. (17) has exactly the same argument. The HDP structure requires a model network to result in

$$\hat{J}_k \approx U_k + \gamma U_{k+1} + \gamma^2 U_{k+2} + \cdots$$

The ADHDP structure does not need a model network and the function Q_k satisfies

$$Q_k \approx U_{k+1} + \gamma U_{k+2} + \gamma^2 U_{k+3} + \cdots$$

Actually, the TD learning is a model-free method. In other words, it does not need the model network to evaluate the value function[47,56].

In addition to the present basic ADP schemes given above, there are also other various structures developed in Refs. [116, 117].

4 Model-Based and Model-Free Adaptive Dynamic Programming

Under a known external environment dynamics, the widely adopted approach to compute the (optimal) cost function is VI. In what follows, a different set of notation will be used. The value function V^μ represents the cost function under the control policy μ, i.e., $V^\mu(x_k) = J^\mu(x_k)$, $\forall x_k$. The corresponding optimal value function is expressed as $V^*(x_0) = \inf_\mu V^\mu(x_0)$.

Similarly, for the value function, there also exist three forms: (i) $V(x_k, \underline{u}_k)$ indicates the value of Eq. (10) starting at x_k and controlled by the control sequence $\underline{u}_k = (u_k, u_{k+1}, \ldots)$. (ii) $V^\mu(x_k)$ denotes the value function of Eq. (10) starting at x_k and controlled by the control policy $u_k = \mu(x_k)$. (iii) $V^*(x_k)$ is the optimal value function of Eq. (10) starting at x_k. For convenience, in this paper, the notation $V(x_k)$ has been used to represent $V(x_k, \underline{u}_k)$ and $V^\mu(x_k)$. This representation of value function is standard in the literature. There will also be cases where the value function is a function of x_k and u_k (not explicitly as a function of $\underline{u}_k$). Therefore, it is appropriate to denote the value function as $V(x_k, u_k)$. Besides, for time-varying systems, the value function is described as $V(x_k, u_k, k)$. As a convention, in these cases, it is also standard to represent value functions using $V(x_k)$ if the context is clear.

As given earlier in Eqs. (4), (6), and (9), the Bellman equation can be iteratively solved using the successive approximation approach.

The task of ADP is to obtain the optimal cost function J^* given in Eq. (12) and then derive the optimal control u_k^* in Eq. (13). One effective approach to approximate J^* is to adopt the following iterative algorithm. Let J^* in Eq. (12) be replaced by V. Then, Eq. (12) can be rewritten as

$$V(x_k) = \min_{u_k}\left\{U(x_k, u_k) + \gamma V(x_{k+1})\right\},$$

which can be further described as

$$V_i(x_k) = \min_{u_k}\left\{U(x_k, u_k) + \gamma V_{i-1}(x_{k+1})\right\} \tag{18}$$

where $i = 1, 2, \ldots$ is the iteration index. The iteration scheme is similar to the fixed point iteration. For example, the algebraic equation $z = g(z)$ is solved by using the iterative method, where $g(z)$ is called the fixed point equation. Then, the iteration scheme is implemented by establishing the iteration equation $z_{i+1} = g(z_i)$. Starting from z_0, the iteration equation is used to compute $z_1 = g(z_0), z_2 = g(z_1), \ldots$ If $g(z)$ is a contraction map, then the fixed point is unique and we have $z_\infty = g(z_\infty)$ starting from any initial value z_0.

For Eq. (18), the value function is initialized by V_0 and the iterative value functions V_1, V_2, and so on, can be iteratively derived from this equation. One would hope that $V_\infty(x_k)$ is the solution of the Eq. (18) when the iteration step i reaches ∞. Note that a solution can only be obtained if the sequence $\{V_i\}$ is convergent.

Considering the iteration process in the previous paragraph, we would expect to obtain a solution which can approximate the solution of DP with high accuracy. In the iteration process, a control policy corresponding to each iterative value function V_i is determined by

$$v_i(x_k) = \arg\min_{u_k}\left\{U(x_k, u_k) + \gamma V_i(F(x_k, u_k))\right\} \tag{19}$$

This sequence of control signals $\{v_0, v_1, \ldots\}$ is called the iterative control policy sequence.

The detailed VI-based ADP algorithm has been given. Next, the theoretical results of the iterative ADP approach will be introduced and discussed, which includes stability of the policy, convergence of value function and policy sequences, and optimality of the obtained solution. It is meaningful and essential to guarantee the convergence of the iterative solution process. In

addition, guaranteeing the convergence of the sequence $\{V_i\}$ is not sufficient. The key property of the sequence $\{V_i\}$ is that $\{V_i\}$ converges to the optimal cost J^* as $i \to \infty$.

The simplest initialization of VI is to start from $V_0(x_k) \equiv 0$, $\forall x_k$[118–120]. Considering the undiscounted optimal control problem in the sense that the discount factor is set as 1. Then, Eq. (18) is rewritten as

$$V_i(x_k) = \min_{u_k} \left\{ U(x_k, u_k) + V_{i-1}(x_{k+1}) \right\},\ i = 1, 2, \ldots \tag{20}$$

Rantzer and his coworkers proved the following proposition.

Proposition 1 (Convergence of VI[118, 119]). Suppose that, $\forall x_k$ and $\forall u_k$, the inequality $J^*(F(x_k, u_k)) \leqslant \rho U(x_k, u_k)$ holds uniformly for some $\rho < \infty$ and that $\eta J^*(x_k) \leqslant V_0(x_k) \leqslant J^*(x_k)$ for some $0 \leqslant \eta \leqslant 1$. Then, $\{V_i\}$ given by Eq. (20) approximates J^* according to the inequalities

$$\left[1 + \frac{\eta - 1}{(1 + \rho^{-1})^i}\right] J^*(x_k) \leqslant V_i(x_k) \leqslant J^*(x_k),\ \forall x_k.$$

It was shown by Proposition 1 that the convergence and optimality of VI can be guaranteed, i.e., $V_i(x_k) \to J^*(x_k)$ as $i \to \infty$.

The affine form of Eq. (10) has been largely studied, which is expressed as

$$x_{k+1} = f(x_k) + g(x_k)u(x_k),\quad k = 0, 1, 2, \ldots$$

where $f: \mathbb{R}^n \to \mathbb{R}^n$ and $g: \mathbb{R}^n \to \mathbb{R}^{n \times m}$ are nonlinear system dynamics. In general, the utility function is selected as the following quadratic form:

$$U(x_k, u_k) = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k$$

where $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ are positive definite matrices. In Ref. [120], an iterative ADP algorithm and its properties were derived as follows. The iterative ADP starts from a zero initial value function, i.e., $V_0(x_k) \equiv 0$, $\forall x_k$. Then, the initial control policy v_0 is solved by

$$v_0(x_k) = \arg\min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + V_0(x_{k+1}) \right\} \tag{21}$$

where $V_0(x_{k+1}) = 0$. The next value function can be determined by

$$\begin{aligned} V_1(x_k) =& x_k^{\mathrm{T}} Q x_k + v_0^{\mathrm{T}}(x_k) R v_0(x_k) + V_0(x_{k+1}) = \\ & x_k^{\mathrm{T}} Q x_k + v_0^{\mathrm{T}}(x_k) R v_0(x_k) + V_0\left(f(x_k) + g(x_k) v_0(x_k)\right) \end{aligned} \tag{22}$$

For the iteration step i, the successive approximation process will be implemented between control policies

$$\begin{aligned} v_i(x_k) =& \arg\min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + V_i(x_{k+1}) \right\} = \\ & \arg\min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + V_i\left(f(x_k) + g(x_k) u_k\right) \right\} = \\ & -\frac{1}{2} R^{-1} g^{\mathrm{T}}(x_k) \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}} \end{aligned} \tag{23}$$

and value functions

$$\begin{aligned} V_{i+1}(x_k) =& \min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + V_i(x_{k+1}) \right\} = \\ & x_k^{\mathrm{T}} Q x_k + v_i^{\mathrm{T}}(x_k) R v_i(x_k) + V_i\left(f(x_k) + g(x_k) v_i(x_k)\right) \end{aligned} \tag{24}$$

where $i = 1, 2, \ldots$.

We can rewrite the above Eqs. (21)–(24) as follows. Firstly $v_0(x_k)$ is obtained from $V_0(x_k) = 0$ as

$$v_0(x_k) = \arg\min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k \right\} \tag{25}$$

Then the iteration will be performed between value function update

$$\begin{aligned} V_i(x_k) =& \min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + V_{i-1}(x_{k+1}) \right\} = \\ & x_k^{\mathrm{T}} Q x_k + v_{i-1}^{\mathrm{T}}(x_k) R v_{i-1}(x_k) + V_{i-1}\left(f(x_k) + g(x_k) v_{i-1}(x_k)\right) \end{aligned} \tag{26}$$

and control policy improvement

$$\begin{aligned} v_i(x_k) =& \arg\min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + V_i(x_{k+1}) \right\} = \\ & \arg\min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + V_i\left(f(x_k) + g(x_k) u_k\right) \right\} = \\ & -\frac{1}{2} R^{-1} g^{\mathrm{T}}(x_k) \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}} \end{aligned} \tag{27}$$

where $i = 1, 2, \ldots$. It is noted that Eqs. (21)–(24) and Eqs. (25)–(27) are exactly the same algorithm.

The following results were shown in Ref. [120].

(1) The sequence $\{V_i(x_k)\}$ with $V_0(x_k) = 0$ will be monotonically non-decreasing and is bounded by a continuous function. Therefore, the limit of $V_i(x_k)$ when $i \to \infty$, i.e., $V_\infty(x_k)$, exists.

(2) $V_\infty(x_k) = J^*(x_k)$ in the sense that the iterative solution converges to the optimal cost.

(3) $v_\infty(x_k) = u^*(x_k)$ in the sense that the iterative control policy converges to the optimal control policy.

Along the line of Ref. [120], there are further results[121–130]. Along the line of Rantzer's work[118, 119], some of important developments can be also found[131–139].

Another effective method to numerically approximate $J^*(x_k)$ and $u^*(x_k)$ is policy iteration (PI). PI starts from an arbitrary admissible control policy $v_0(x_k)$. The PI scheme is implemented between the policy evaluation

$$V_i(x_k) = x_k^{\mathrm{T}} Q x_k + v_{i-1}^{\mathrm{T}}(x_k) R v_{i-1}(x_k) + V_i\left(f(x_k) + g(x_k) v_{i-1}(x_k)\right) \tag{28}$$

and the policy improvement

$$\begin{aligned} v_i(x_k) =& \arg\min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + V_i(x_{k+1}) \right\} = \\ & \arg\min_{u_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + V_i\left(f(x_k) + g(x_k) u_k\right) \right\} = \\ & -\frac{1}{2} R^{-1} g^{\mathrm{T}}(x_k) \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}} \end{aligned} \tag{29}$$

where $i = 1, 2, \ldots$. When the policy evaluation cannot be solved directly, the successive approximation approach is usually used to iteratively approximate its numerical solution at each iteration. The monotonicity, convergence and optimality of the value function sequence generated by PI have been investigated[125]. Later, a generalized policy iteration (GPI) algorithm[126] was presented to solve the infinite horizon optimal control problem. Moreover, the convergence of the PI and GPI algorithms were proved. Recently, some research has focused on the convergence rate of the iterative ADP algorithms. Luo et al.[140] found that PI converges to the optimal value function faster than VI while it requires an initial admissible control policy. A tradeoff between PI and VI was achieved to accelerate the convergence by introducing a balancing parameter. Inspired by the successive over relaxation method, Ha et al.[141] developed a novel VI scheme with a adjustable convergence rate and gave a practical accelerated ADP algorithm. Similar to the traditional iterative ADP, the present ADP algorithm is initialized by a positive semi-definite function $\widetilde{V}_0(x_k)$. The initial control policy $\tilde{v}_0(x_k)$ is also obtained. The novel ADP scheme achieves a balance between

$\min_{u_k}\{x_k^{\mathrm{T}}Qx_k+u_k^{\mathrm{T}}Ru_k+\widetilde{V}_{i-1}(x_{k+1})\}$ and $\widetilde{V}_{i-1}(x_k)$ by the use of a relaxation factor w. The iterative procedure is performed between the novel value function update

$$\widetilde{V}_i(x_k)=w\min_{u_k}\left\{x_k^{\mathrm{T}}Qx_k+u_k^{\mathrm{T}}Ru_k+\widetilde{V}_{i-1}(x_{k+1})\right\}+(1-w)\widetilde{V}_{i-1}(x_k) \tag{30}$$

and the policy improvement

$$\tilde{v}_i(x_k)=\arg\min_{u_k}\left\{x_k^{\mathrm{T}}Qx_k+u_k^{\mathrm{T}}Ru_k+\widetilde{V}_i(x_{k+1})\right\} \tag{31}$$

where the relaxation factor satisfies $w>0$. As mentioned[141], if $0<w<1$, the developed VI scheme is an under-relaxation method. If $w=1$, it is the traditional VI. If $w>1$, it is an over-relaxation method. Since the value-based ADPRL methods including TD, Sarsa, TD(λ), Sarsa(λ), and Q(λ) are based on the generalized PI framework, the new iterative ADP scheme can be extended to these ADPRL algorithms to accelerate convergence of the value function.

Note that the iterative ADP methods mentioned above consider the case of the known system dynamics. Based on the ADP scheme, considerable literatures[142–150] also focus attention on the case of unknown system dynamics and many researchers are devoted to developing the model-free and data-driven ADP technique. Luo et al.[151] developed a model-free policy-gradient-based ADP algorithm to achieve optimal control by using offline and online data. A model-free ADP controller[152] was designed to solve the optimal control problem by using the supplement information derived from the costate function, which does not require any plant information. Aiming at the data-driven control problem, Li et al.[153,154] developed a series of data-driven ADP methods and investigated the stability and domain of attraction of the closed-loop systems, which do not need the plant modeling process.

On the other hand, optimal tracking control is also a significant topic in the control community, which mainly aims at designing a controller to make the controlled plant track a reference trajectory. The literature on this problem is extensive[155–161] and reflects considerable current activity. Luo et al.[162] converted a tracking control problem to a regulation problem with a discounted performance index by establishing an augmented system. A multistep heuristic dynamic programming was developed to achieve the tradeoff between PI and VI by using multistep policy evaluation scheme. Afterwards, it is revealed that the tracking error cannot be eliminated if the traditional performance index is adopted[163]. To eliminate completely the tracking error as the number of time steps increases, Li et al.[163] designed a novel cost function. Besides, the convergence and monotonicity of the obtained new value function sequence were also investigated. Based on the novel cost function[163] and considering the effect of discount factor, a novel stability analysis method was developed to guarantee that the tracking error under the new control policy approaches zero as the number of time steps increases[164]. Moreover, the effect of the presence of the approximation errors of the value function is discussed. A novel inverse RL algorithm[165] was also developed to learns an unknown performance objective function for tracking control. In addition, some RL-based tracking algorithms have been developed and applied to various practical applications[166–169]. Aiming at the unmanned surface vehicle with complex unknowns[170], including dead-zone input nonlinearities, system dynamics, and disturbances, an RL-based neuro-tracking controller was designed to solve the optimal tracking problem. Cao et al.[169] established a fixed-time trajectory tracking control method for uncertain robotic manipulators with input saturation, which employs a new nonsingular fast terminal sliding mode approach to guarantee the convergence of tracking error in fixed time. Rizvi et al.[171] considered the a heating, ventilating, and air conditioning system in the presence of neither measurable nor manipulable disturbance and, based on PI and VI, proposed both state feedback and output feedback approaches to guarantee the convergence of the tracking error to zero. For unknown nonlinear input-affine discrete-time systems and based on Q-learning, a model-free dynamic inversion-based tracking control algorithm[172] was proposed to eliminate the tracking error, which is an off-policy approach. A model-free and off-policy adaptive critic design with experience replay were developed to improve the optimal tracking control performance by using the policy gradient technique[173]. In offshore oil and gas production, Li et al.[174] applied the model-free RL approaches based on state feedback and output feedback to the de-oiling hydrocyclone system. Output tracking control and ADP algorithms were integrated to track the desired yaw rate, mitigate the sideslip angle, roll angle, and roll rate of the vehicle[175]. With this operation, the lateral stability of vehicle dynamic systems can be improved.

5 Convergence and Optimality

In this section, for the traditional and new iterative ADP schemes, convergence and optimality properties are reviewed.

5.1 Convergence and optimality of the traditional iterative adaptive dynamic programming

The convergence and optimality of VI-based HDP algorithm have been proved[120]. The VI-based HDP scheme starts from a zero value function. Then, a monotonically non-decreasing value function sequence is established, as shown in Fig. 5. The value function will increase monotonically to get close to V^* as the iteration index increases. Since it is impossible to perform the iterative ADP infinite number of steps, an ε-optimal control method was developed for finite-horizon optimal control[121].

Aiming at the discounted optimal control problem, an iterative GHDP algorithm were considered[122,123]. In order to reduce the complexity of DP, a relaxation process based on the upper and lower bounds of the optimal cost function was developed[118,119]. For the constrained input control[124], finite-horizon optimal control[121] and optimal tracking control[137], the iterative ADP methods were applied to find their optimal control policies. Under these control problems, it can also be guaranteed that their value function

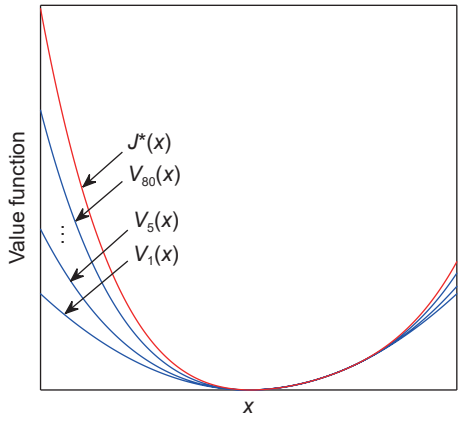

Fig. 5 Monotonically non-decreasing value function sequence.

sequences converge to the corresponding optimal value functions. Later, Wei and Liu[127] and Li and Liu[131] found that the value function can be initialized by a positive semidefinite function. The VI scheme with a positive semidefinite initial value function is called generalized value iteration (GVI). Using a positive semidefinite function to initialize VI is an extension to the zero initial value function. With this operation, the monotonicity of the $\{V_i\}$ sequence is determined by the selection of the initial value function. Under certain conditions[127, 131, 132, 138], the $\{V_i\}$ sequence can be monotonically non-increasing, which is more appropriate since the goal of our learning process is to minimize the value function and thus it is expected that the iterative learning process will decrease the values of V_i gradually. Besides, it is convenient to discuss the stability of the closed-loop system using the iterative control policy by constructing a monotonically non-increasing value function sequence. The relevant stability results will be given in the next section. Heydari[176] used an admissible control policy to initialize the VI algorithm, which makes the VI scheme possess the stability guarantee and is called the stabilizing VI. An iterative θ-ADP algorithm that can effectively avoid the initial admissible control was proposed[127]. A nearly optimal output feedback control algorithm was developed[128] for discrete-time systems. The Markov jump systems control problem was converted to a single objective optimal control problem[129], and then the optimal controller was obtained by using ADP method. Event-triggering mechanism was also considered for discrete-time control problems[130].

In the literature, there are various approaches to obtain a monotonically non-increasing $\{V_i\}$ sequence, such as iterative θ-ADP, stabilizing VI[176, 177], PI[125], GPI[126, 136] and so forth. The following monotonicity results of GVI and stabilizing VI have been developed.

Proposition 2 (Monotonicity of $\{V_i\}$[131, 176]). Let $V_i(x_k)$ and $v_i(x_k)$ be updated by the value function update (Eq. (26)) and policy improvement (Eq. (27)), respectively.

(1) If the initial value function is a positive semidefinite function and satisfies $V_0(x_k) \leqslant V_1(x_k)$, then the obtained sequence $\{V_i\}$ is monotonically non-decreasing, i.e., $V_i(x_k) \leqslant V_{i+1}(x_k) \leqslant J^*(x_k)$, $\forall i$. If $V_0(x_k) \geqslant V_1(x_k)$, the $\{V_i\}$ sequence is monotonically non-increasing, i.e., $V_i(x_k) \geqslant V_{i+1}(x_k) \geqslant J^*(x_k), \forall i$.

(2) If the VI algorithm is initialized by the value function of an admissible control policy, then the sequence $\{V_i\}$ is monotonically non-increasing.

For nonlinear optimal control problems with discount factor $0 < \gamma < 1$, similar to the undiscounted VI algorithm, the convergence analysis of the discounted value function sequence was given in Refs. [122, 123]. Note that the cost function without discount factor is a special case of the cost function with discount factor. The following results were given in Refs. [122, 123].

(1) The $\{V_i\}$ sequence with $V_0(\cdot) = 0$ is defined as in (18), $0 < \gamma \leqslant 1$. If the system is controllable, there exists an upper bound Y such that $V_i(x_k) \leqslant Y, \forall i$.

(2) The discounted value function sequence with $V_0(\cdot) = 0$ and control law are updated by (18) and (19), where $0 < \gamma \leqslant 1$. Then, the discounted value function sequence $\{V_i\}$ is monotonically non-decreasing.

(3) $\lim\limits_{i\to\infty} V_i(x_k) = J^*(x_k)$ and $\lim\limits_{i\to\infty} v_i(x_k) = v^*(x_k)$.

Based on the works by Rantzer and his coworkers, the following proposition is given[131, 132, 137] for the GVI algorithm.

Proposition 3 (Convergence of GVI[131, 132]). For all x_k and for all u_k, suppose that the inequality $J^*(F(x_k, u_k)) \leqslant \rho U(x_k, u_k)$ holds uniformly for some $\rho < \infty$ and that $V_1(x_k) \leqslant V_0(x_k) \leqslant \zeta J^*(x_k)$ for some $1 \leqslant \zeta < \infty$. Then, the sequence $\{V_i\}$ defined iteratively by (20) approaches J^* according to the inequalities

$$J^*(x_k) \leqslant V_i(x_k) \leqslant \left[1 + \frac{\zeta - 1}{(1+\rho^{-1})^i}\right] J^*(x_k), \ \forall x_k.$$

According to Proposition 3, the sequence $\{V_i\}$ resulting from GVI converges to the optimal cost as $i \to \infty$. As shown in Fig. 6, there are three ways to ensure that the iterative value function sequence derived from GVI converges to the optimal cost. When $V_0 \leqslant J^*$, the $\{V_i\}$ sequence converges to the optimal value function J^* from below. When $V_0 \geqslant J^*$, the $\{V_i\}$ sequence converges to the optimal value function J^* from above. When the relationship between V_0 and J^* is uncertain, the $\{V_i\}$ sequence still converges to J^* by squeezing the upper and lower bounds of V_i. Note that the monotonicity of the sequence $\{V_i\}$ cannot be guaranteed when $V_0(x_k) \leqslant J^*(x_k)$ for some x and $V_0(x_k) \geqslant J^*(x_k)$ for others. Similar convergence and optimality results of GVI for discrete-time optimal tracking control were also discussed and developed[137].

In the value function update (Eq. (26)) and policy evaluation (Eq. (28)), function approximation structures like NNs are usually employed to approximate the value function, which will introduce approximation errors. Since the value update and policy evaluation equations cannot be solved exactly[132], the obtained value functions are usually their estimations. The approximation errors of NN were considered and the convergence analysis of the approximate iterative value function was developed[131, 132, 137]. It is guaranteed that the approximate value function sequence $\{\hat{V}_i\}$ converges to a finite neighborhood of the optimal value function as the iteration index increases under some conditions. Based on iterative θ-ADP algorithm, the upper and lower bounds of the

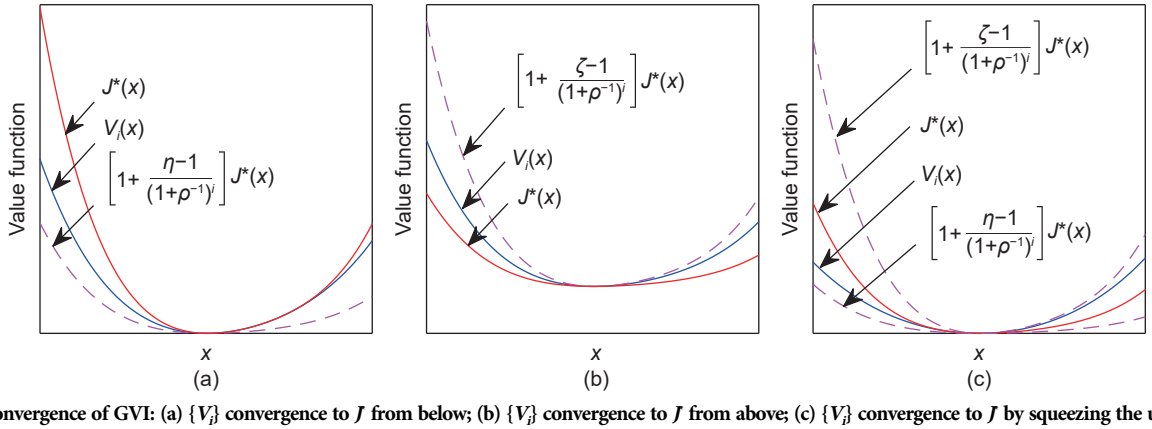

Fig. 6 Convergence of GVI: (a) $\{V_i\}$ convergence to J from below; (b) $\{V_i\}$ convergence to J from above; (c) $\{V_i\}$ convergence to J by squeezing the upper and lower bounds of V_i.

obtained approximate value function were given[133–135]. The iterative value function with approximation errors is shown to converge to a finite neighborhood of the optimal cost. For the GVI scheme, the relationship between the approximate value function $\hat{V}_i$ and the exact value function V_i was revealed[138].

On the other hand, the convergence and optimality of the $\{V_i\}$ sequence for PI and GPI have also been obtained[125, 126, 132, 136, 139]. The monotonicity, convergence and stability properties of PI for discrete-time nonlinear systems[125] were analyzed for the first time. It was proved that any of the iterative control policies derived from PI are admissible, which makes PI play a significant role in online ADP methods. The corresponding stability results are presented in the next section. The following monotonicity and convergence results of PI were obtained[125].

(1) Starting with an admissible control policy, $V_i(x_k)$ and $v_i(x_k)$ are updated by the policy evaluation and policy improvement. Then, the $\{V_i\}$ sequence generated by PI is monotonically non-increasing, i.e., $V_i(x_k) \geqslant V_{i+1}(x_k) \geqslant J^*(x_k), \forall i$.

(2) The iterative value function and control policy sequences resulting from PI converge to the optimal performance index and the optimal control policy, respectively, i.e., $\lim_{i\to\infty} V_i(x_k) = J^*(x_k)$ and $\lim_{i\to\infty} v_i(x_k) = v^*(x_k), \forall x_k$.

The error bounds for approximate value function of PI and GPI were established[132, 136]. The obtained $\{\hat{V}_i\}$ sequence also converges to a finite neighborhood of J^* as $i \to \infty$. Aiming at the discounted optimal control problem, the convergence of exact Q-function and error bound analysis of the Q-function for the PI-based action-dependent ADP were investigated[139]. Similar to the approximate value function, it has been proved that the approximate Q-function under the boundedness conditions converges to a finite neighborhood of the optimal Q-function.

5.2 Convergence and optimality of the new iterative adaptive dynamic programming algorithm

According to the analysis approach of the traditional iterative ADP methods, the convergence analysis of the new iterative ADP scheme as mentioned in Eqs. (30) and (31) has been given in Ref. [141], which obtained the following results.

Proposition 4 (Convergence of the new iterative ADP scheme[141]). Suppose that the condition $0 \leqslant J^*(F(x_k, u_k)) \leqslant \rho U(x_k, u_k)$ is satisfied and that $0 \leqslant \eta J^*(x_k) \leqslant \widetilde{V}_0(x_k) \leqslant \zeta J^*(x_k)$, where $0 < \rho < \infty$ and $0 \leqslant \eta \leqslant 1 \leqslant \zeta$. Let the value function and the control policy be iteratively updated by (30) and (31).

(1) If the relaxation factor satisfies $0 < w \leqslant 1$, then the iterative value function $\widetilde{V}_i$ approximates the optimal value function J^* according to the following inequalities:

$$\left[1-\left(1-\frac{w}{1+\rho}\right)^i(1-\eta)\right]J^*(x_k) \leqslant \widetilde{V}_i(x_k) \leqslant \left[1+\left(1-\frac{w}{1+\rho}\right)^i(\zeta-1)\right]J^*(x_k).$$

(2) If the relaxation factor satisfies

$$1 \leqslant w \leqslant 1+\frac{Ld_{\min}}{(1+\rho)(\zeta-\eta)},$$

where $L \in (0,1)$ is a constant and $d_{\min} = \min\{1-\eta, \zeta-1\}$, then $\widetilde{V}_i$ approximates J^* according to the following inequalities:

$$\left[1-\left(1-\frac{w-L}{1+\rho}\right)^i(1-\eta)\right]J^*(x_k) \leqslant \widetilde{V}_i(x_k) \leqslant \left[1+\left(1-\frac{w-L}{1+\rho}\right)^i(\zeta-1)\right]J^*(x_k).$$

(3) If the relaxation factor satisfies

$$0 \leqslant w \leqslant 1+\frac{Ld_{\min}}{(1+\rho)(\zeta-\eta)},$$

then $\lim_{i\to\infty} \widetilde{V}_i(x_k) = J^*(x_k)$ and $\lim_{i\to\infty} \widetilde{v}_i(x_k) = v^*(x_k), \forall x_k$.

The upper and lower bounds demonstrate that the bigger the w is, the faster the convergence.

6 Stability Result

Stability is the fundamental requirement of all control systems. In this section, the stability results of the iterative ADPs are revealed.

6.1 Stability of the the traditional iterative adaptive dynamic programming algorithms

Recently, there has been considerable literature[125, 127, 131, 135, 177–181] on the stability of closed-loop systems using control policies derived from the iterative ADP methods. For the PI algorithm, it can be guaranteed that all the iterative control policies are admissible[125]. However, GVI initialized by a positive semidefinite function does not possess similar stability properties to PI. Till now, the admissibility of control policies derived from the traditional VI and GVI has been paid considerable attentions[131, 134, 136, 178, 180, 181]. Various VI schemes such as iterative θ-ADP[127, 135], stabilizing VI[176, 177], and so forth were developed to guarantee the stability of closed-loop systems using the iterative control policy. The control policies generated by the iterative θ-ADP under some conditions of θ are asymptotically stable control laws[127]. A stabilizing VI algorithm was developed[176], which is initialized by the value function of an admissible policy. Such an initialization is similar to PI. By doing this, the obtained value function sequence is monotonically non-increasing, which ensures the stability of the iterative control policies and makes the implementations both online and offline feasible. The following stability results of stabilizing VI were provided in Ref. [176].

(1) Let the VI algorithm be initialized by the value function of an admissible control policy. Then, the closed-loop system using the iterative control policy v_i is asymptotically stable, $\forall i \in \mathbb{N}$.

(2) The compact set $\mathbb{O}_r^i \triangleq \{x \in \mathbb{R}^n : V_i(x) \leqslant r\} \subset \Omega_x$ for any $r > 0$ is a subset of the domain of attraction for the closed-loop system.

(3) Let each iterative control policy $v_i(x_k)$ in the $\{v_i\}$ sequence be applied to the controlled system for $L_i \in \mathbb{N}$ time steps. Then, every state trajectory initiated from Ω_x converges to the origin.

In summary, the objective of these different initialization approaches of VI is to establish a monotonically non-increasing value function sequence. For the value function sequence that the monotonically non-increasing property cannot be guaranteed, Wei et al.[178] gave the uniform ultimate boundedness (UUB) condition and the admissibility condition of the iterative control policy for the first time. The stability of the closed-loop system under offline and online VI schemes without approximation errors was investigated[180]. The main stability results of GVI are summarized in the following proposition.

Proposition 5 (Stability of GVI[131, 178, 180]). The $\{V_i\}$ sequence with an initial positive semidefinite value function and the $\{v_i\}$

sequence are updated by Eqs. (26) and (27), respectively.

(1) If there exists a constant ε such that

$$|V_{i+1}(x_k) - V_i(x_k)| \leqslant \varepsilon,\ \forall x_k,$$

then the closed-loop system using the iterative control policy $v_i(x_k)$ is UUB.

(2) If $V_0(x_k) \geqslant V_1(x_k)$ holds for all x_k, then the iterative function $V_i(x_k)$ is a Lyapunov function and the system using $v_i(x_k)$ is asymptotically stable, $\forall i \in \mathbb{N}$.

(3) If for all $x_k \neq 0$, the following condition

$$V_{i+1}(x_k) - V_i(x_k) < \kappa U(x_k, v_i(x_k)),\ \kappa \in (0,1) \tag{32}$$

holds, then the iterative control policy $v_i(x_k)$ is admissible.

(4) For all $x_k \neq 0$, there exists a finite $N > 0$ such that

$$V_{N+1}(x_k) - V_N(x_k) < \kappa U(x_k, v_N(x_k)),\ \kappa \in (0,1).$$

(5) If for any $x_k \neq 0$, the following condition

$$J^*(x_k) - V_i(x_k) < \kappa x_k^{\mathrm{T}} Q x_k,\ \kappa \in (0,1) \tag{33}$$

holds, then the iterative control policy $v_{i+j}(x_k)$ is admissible, $\forall j \in \mathbb{N}$.

According to the monotonicity of $\{V_i\}$ and considering the property (2) in Proposition 5, a monotonically non-increasing value function sequence can be obtained when $V_0 \geqslant V_1$. Then, the iterative control policies derived from GVI are asymptotically stable[131]. In addition, when the iterative value function sequence is monotonically non-decreasing, the stability criterion Eq. (32) in Proposition 5 can be used to determine the admissibility of the control policy at the current iteration. The stability condition (Eq. (32)) was modified as $V_{i+1}(x_k) - V_i(x_k) < \kappa x_k^{\mathrm{T}} Q x_k$, which is more appropriate since the term $x_k^{\mathrm{T}} Q x_k$ does not vary with the iteration index while the terms $U(x_k, v_i(x_k))$ at different iterations are different[180]. Besides, the corresponding domain of attraction for the closed-loop system can also be deduced. On the other hand, if the stability condition (Eq. (33)) is satisfied, then the iterative value functions V_{i+j+1} and V_{i+j} must satisfy the stability condition (Eq. (32)), $j \in \mathbb{N}$. According to the convergence result $\lim_{i\to\infty} V_i(x_k) = J^*(x_k)$, the property (5) in Proposition 5 reveals that, in the iterative process of GVI, there exists an iteration step such that the control policies obtained after this iteration are admissible. As shown in Fig. 7, if the iterative value function is located in the shadow, the corresponding control policy is admissible. Based on the stability condition $V_{i+1}(x_k) - V_i(x_k) < \kappa x_k^{\mathrm{T}} Q x_k$, the stabilizing VI was integrated with the traditional VI[180], which employed the traditional VI to generate the admissible control policy and applied the obtained admissible control policy to initialize the stabilizing VI.

It was further discussed the stability of the linear system under the iterative control policy and obtained the global asymptotic stability results[180], which greatly facilitated the stability analysis of the closed-loop linear systems under the online HDP algorithm. For the GVI algorithm with discount factor, the effect of the discount factor on the stability of the iterative control policy has been investigated[181]. As mentioned in Ref. [181], unlike the undiscounted GVI, the monotonically non-increasing $\{V_i\}$ sequence of the discounted GVI does not always result in the stabilizing iterative control policy due to the introduction of the discount factor. Similar to the undiscounted VI, the monotonicity of the discounted iterative value function sequence can be determined by the relationship between $V_0(x_k)$ and $V_1(x_k)$[181]. The relevant stability results with respect to the discounted GVI is presented in Proposition 6.

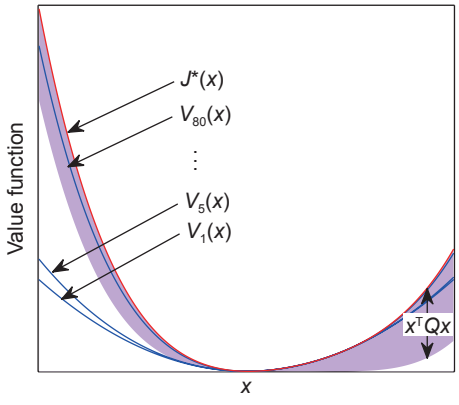

Fig. 7 Stability results of GVI.

Proposition 6. (Stability of the discounted GVI[181]). The iterative value function sequence with an initial positive semidefinite function is defined as in Eq. (18), $0 < \gamma \leqslant 1$.

(1) If $V_0(x_k) \geqslant V_1(x_k)$ and for any $x_k \neq 0$, the following condition

$$(1-\gamma) V_i(x_k) < U(x_k, v_i(x_k)) \tag{34}$$

holds, then the closed-loop system using the iterative control policy $v_i(x_k)$ is asymptotically stable.

(2) If $V_0(x_k) \geqslant V_1(x_k)$ and for any $x_k \neq 0$,

$$(1-\gamma) V_i(x_k) < x_k^{\mathrm{T}} Q x_k \tag{35}$$

then the iterative control policy $v_{i+j}(x_k)$ is asymptotically stable, $\forall j \in \mathbb{N}$.

From Proposition 6, the selection of the discount factor is significant. If an inappropriate discount factor is selected, the stability conditions may not be satisfied in the iteration process. This even leads to an unstable optimal control policy. Then, any ADP methods to numerically approximate the optimal cost and the optimal control policy may be meaningless. Note that the stability condition (Eq. (34)) can only be used to determine the stability of the current iterative control policy while the condition (Eq. (35)) guarantees that all the iterative control policies after the current iteration are asymptotically stable.

Since the function approximation structure is used to approximate the value function and the control policy, the stability of the system could be at risk due to the approximation errors of the value function and control policy. Based on the θ-ADP method, it is proved that the numerical iterative control law under some conditions is an asymptotically stable control policy[134,135]. For the stabilizing VI, effects of the presence of approximation errors were presented on the stability of the controlled systems[177]. Also, estimations of the domain of attraction, under the approximation errors of the value function, were given. For the traditional VI algorithm, the stability conditions of the approximation error of the actor were provided[179], which ensure that the obtained approximate control policies can asymptotically stabilize the controlled systems.

As described in the previous paragraph, NN-based adaptive critic scheme is employed in the implementation of the iterative ADP scheme[122,123,182,183]. The stability of NN-based controllers has been systematically and fully investigated[97,130,149,182,184]. For affine systems, the H_∞ state feedback controller was designed and the stability analysis was elaborated[185]. It is proved that the NN identifier can make the state estimation error dynamics asymptotically stable[122]. Sokolov et al.[186] extended previous results to the case of multi-layer NNs across all layers. Additionally, the

proposed control method based on ADHDP is UUB under some mild conditions. According to Ref. [183], the UUB stability of the parameter estimation errors can be guaranteed when the weights and biases across all layers are updated by the gradient descent algorithm. For the model-free ADHDP approach with an eligibility trace long-term prediction parameter λ, the UUB conditions[184, 187] were investigated by using Lyapunov stability. Considering GDHP algorithm, Kim et al.[188] provided the elementwise error bound of the costate function sequence and, for the general multi-layer NNs, developed UUB stability of weights.

6.2 Stability of the new iterative adaptive dynamic programming algorithm

For the new iterative ADP algorithm in Eqs. (30) and (31), the stability of the closed-loop system using $\tilde{v}_i$ has also been revealed[141]. The positive definiteness of $\{\widetilde{V}_i\}$ and the stability results are summarized in the following propositions.

Proposition 7. (Positive definiteness of $\{\widetilde{V}_i\}$[141]). Suppose that $0 \leqslant J^*(F(x_k,u_k)) \leqslant \rho U(x_k,u_k)$ and $0 \leqslant \eta J^*(x_k) \leqslant \widetilde{V}_0(x_k) \leqslant \zeta J^*(x_k)$, where $0<\rho<\infty$ and $0<\eta\leqslant 1<\zeta$. Let the value function and the control policy be iteratively updated by Eqs. (30) and (31). If the relaxation factor is selected to satisfy

$$0 < w \leqslant 1+\frac{\zeta-\eta-d_{\max}}{\rho(\zeta-\eta)+d_{\max}} \tag{36}$$

where $d_{\max}=\max\{\zeta-1,1-\eta\}$, then the iterative value function $\widetilde{V}_i$ is positive definite.

Proposition 8. (Stability of the new iterative ADP scheme[141]). Assume that $J^*(F(x_k,u_k)) \leqslant \rho U(x_k,u_k)$ and $0 \leqslant \widetilde{V}_0(x_k) \leqslant J^*(x_k)$ hold. Let $\widetilde{V}_i(x_k)$ and $\tilde{v}_i(x_k)$ be updated by Eqs. (30) and (31), respectively.

(1) If the difference between $\widetilde{V}_{i+1}(x_k)$ and $\widetilde{V}_i(x_k)$, $\forall x_k \neq 0$, satisfies

$$\widetilde{V}_{i+1}(x_k)-\widetilde{V}_i(x_k) < wcx_k^{\mathrm{T}}Qx_k$$

where the relaxation factor satisfies the condition (36) and $c\in(0,1)$ is a constant, then the control policy $\tilde{v}_i(x_k)$ is admissible.

(2) If the relaxation factor satisfies

$$0 < w \leqslant 1+\frac{Ld_{\min}}{(1+\rho)(\zeta-\eta)},$$

then there must exist an iteration index i_s such that the control policy $\tilde{v}_{i_s+j}(x_k)$ is admissible, $j\in\mathbb{N}$.

7 Conclusion

In this article, the state-of-the-art developments of ADPRL, involving various algorithms and theoretical analyses, have been revisited. Especially, in control systems community, ADPRL methods have been developed considerably and widely applied to different engineering problems. The adaptive feature and general learning capability of ADPRL methods have aroused huge interests of researchers. Werbos[74, 105, 189, 190] pointed out that the core idea of ADP may be the most likely to realize truly brain-like intelligence. The increasing evidence has demonstrated that optimality is an organizing principle for understanding brain intelligence[105, 106, 190]. In recent years, the brain research around the world has been extensive and reflects a considerable interest. ADPRL has a lot of potential to make contributions to brain research and brain-like intelligence. Besides, with increasing developments regarding the understanding of brain operating styles, more intelligent ADPRL approaches can then be developed.

Deep RL possess the end-to-end learning characteristic in the sense that it can make a swift decision directly by inputting images to the agent. With this operation, the representation capability of deep learning and the decision-making ability of RL are incorporated and fully utilized. The end-to-end learning mechanism makes the agent much closer to human thinking. Because of the end-to-end learning property, deep RL has been paid considerable attention lately. Integrating deep learning and ADPRL will be conducive to establish agents with higher level intelligence. Note that there are still some unsettled concerns to be handled. Most of the pending issues related to approximating solutions of DP with higher accuracy and less computational cost. With the decade-long trends in deep learning, cloud computing, optimization, Metaverse, as well as other mathematical subjects, we believe that ADPRL has a promising development prospect. ADPRL has enjoyed quite remarkable successes for a wide range of fields including the orbital rendezvous, robot arm, urban wastewater treatment, and energy scheduling.

Article History

Received: 26 April 2022; Revised: 19 August 2022; Accepted: 14 September 2022

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of Go with deep neural networks and tree search, *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] D. Silver, J. L. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of Go without human knowledge, *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[5] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, et al., Human-level performance in 3D multiplayer games with population-based reinforcement learning, *Science*, vol. 364, no. 6443, pp. 859–865, 2019.

[6] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al., Grandmaster level in StarCraft II using multi-agent reinforcement learning, *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[7] H. Bastani, K. Drakopoulos, V. Gupta, I. Vlachogiannis, C. Hadjichristodoulou, P. Lagiou, G. Magiorkinis, D. Paraskevis, and S. Tsiodras, Efficient and targeted COVID-19 border testing via reinforcement learning, *Nature*, vol. 599, no. 7883, pp. 108–113, 2021.

[8] L. C. Garaffa, M. Basso, A. A. Konzen, and E. P. De Freitas, Reinforcement learning for mobile robotics exploration: A survey, *IEEE Trans. Neural Netw Learn. Syst.*, doi: 10.1109/TNNLS.2021.3124466.

[9] P. Leinen, M. Esders, K. T. Schütt, C. Wagner, K. R. Müller, and F. S. Tautz, Autonomous robotic nanofabrication with reinforcement learning, *Sci. Adv.*, vol. 6, no. 36, p. eabb6987, 2020.

[10] W. Zhu, X. Guo, D. Owaki, K. Kutsuzawa, and M. Hayashibe, A

survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots, *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2021.3112718.

[11] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, Deep reinforcement learning for autonomous driving: A survey, *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2022.

[12] S. Aradi, Survey of deep reinforcement learning for motion planning of autonomous vehicles, *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, 2022.

[13] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. De Las casas, et al., Magnetic control of tokamak plasmas through deep reinforcement learning, *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.

[14] R. Hafner and M. Riedmiller, Reinforcement learning in feedback control, *Mach. Learn.*, vol. 84, no. 1, pp. 137–169, 2011.

[15] 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. Autom. Sin.*, vol. 5, no. 1, pp. 36–46, 2018.

[16] D. Wang, M. Ha, and J. Qiao, Data-driven iterative adaptive critic control toward an urban wastewater treatment plant, *IEEE Trans. Ind. Electron.*, vol. 68, no. 8, pp. 7362–7369, 2021.

[17] Y. Zhao, Y. Ma, and S. Hu, USV formation and path-following control via deep reinforcement learning with random braking, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5468–5478, 2021.

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

[19] D. Liu, X. Yang, D. Wang, and Q. Wei, Reinforcement-learning-based robust controller design for continuous-time uncertain nonlinear systems subject to input constraints, *IEEE Trans. Cybern.*, vol. 45, no. 7, pp. 1372–1385, 2015.

[20] S. Xue, B. Luo, and D. Liu, Event-triggered adaptive dynamic programming for unmatched uncertain nonlinear continuous-time systems, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 2939–2951, 2021.

[21] Z. Yan and Y. Xu, Real-time optimal power flow: A *Lagrangian* based deep reinforcement learning approach, *IEEE Trans. Power Syst.*, vol. 35, no. 4, pp. 3270–3273, 2020.

[22] N. Wang, Y. Gao, and X. Zhang, Data-driven performance-prescribed reinforcement learning control of an unmanned surface vehicle, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5456–5467, 2021.

[23] D. Liu, D. Wang, and H. Li, Decentralized stabilization for a class of continuous-time nonlinear interconnected systems using online learning optimal control approach, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 418–428, 2014.

[24] S. Xue, B. Luo, D. Liu, and Y. Gao, Event-triggered ADP for tracking control of partially unknown constrained uncertain systems, *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 9001–9012, 2022.

[25] J. Tromp, Number of legal go positions, http://tromp.github.io/go/legal.html, 2021.

[26] F. Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, J. Zhang, and L. Yang, Where does AlphaGo go: From church-Turing thesis to AlphaGo thesis and beyond, *IEEE/CAA J. Autom. Sin.*, vol. 3, no. 2, pp. 113–120, 2016.

[27] G. Tesauro, Practical issues in temporal difference learning, *Mach. Learn.*, vol. 8, no. 3, pp. 257–277, 1992.

[28] G. Tesauro, TD-gammon, a self-teaching backgammon program, achieves master-level play, *Neural Comput.*, vol. 6, no. 2, pp. 215–219, 1994.

[29] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron, A world championship caliber checkers program, *Artif. Intell.*, vol. 53, nos. 2–3, pp. 273–289, 1992.

[30] M. Buro, From simple features to sophisticated evaluation functions, in *Proc. 1st Int. Conf. Computers and Games*, Tsukuba, Japan, 1998, pp. 126–145.

[31] M. Campbell, A. J. Hoane, and F. H. Hsu, Deep blue, *Artif. Intell.*, vol. 134, no. 1-2, pp. 57–83, 2002.

[32] C. Moyer, How Google's AlphaGo beat a Go world champion, https://www.theatlantic.com/technology/archive/2016/03/the-invisible-opponent/475611/, 2016.

[33] S. Byford, AlphaGo retires from competitive Go after defeating world number one 3–0, https://www.theverge.com/2017/5/27/15704-088/alphago-ke-jie-game-3-result-retires-future, 2017.

[34] S. Shead, Google DeepMind is edging towards a 3-0 victory against world Go champion Ke Jie, https://www.businessinsider.nl/google-deepmind-edges-towards-ke-jie-victory-2017-5/, 2017.

[35] Y. Bengio, A. Courville, and P. Vincent, Representation learning: A review and new perspectives, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[36] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[37] J. Schmidhuber, Deep learning in neural networks: An overview, *Neural Netw.*, vol. 61, pp. 85–117, 2015.

[38] X. Cai and D. C. Wunsch, A parallel computer-Go player, using HDP method, in *Proc. Int. Joint Conf. Neural Networks*, Washington, DC, USA, 2001, pp. 2373–2375.

[39] N. N. Schraudolph, P. Dayan, and T. J. Sejnowski, Temporal difference learning of position evaluation in the game of Go, in *Proc. 6th Int. Conf. Neural Information Processing Systems*, Denver, CO, USA, 1993, pp. 817–824.

[40] D. Silver, R. S. Sutton, and M. Müller, Temporal-difference search in computer Go, *Mach. Learn.*, vol. 87, no. 2, pp. 183–219, 2012.

[41] R. Zaman, D. Prokhorov, and D. C. Wunsch, Adaptive critic design in learning to play game of Go, in *Proc. Int. Conf. Neural Networks*, Houston, TX, USA, 1997, pp. 1–4.

[42] R. Zaman and D. C. Wunsch, TD methods applied to mixture of experts for learning 9×9 Go evaluation function, in *Proc. Int. Joint Conf. Neural Networks*, Washington, DC, USA, 1999, pp. 3734–3739.

[43] R. Coulom, Computing ELO ratings of move patterns in the game of Go, *ICGA J.*, vol. 30, no. 4, pp. 198–208, 2007.

[44] M. Enzenberger, Evaluation in Go by a neural network using soft segmentation, in *Proc. 10th Int. Conf. Advances in Computer Games*, Graz, Austria, 2003, pp. 97–108.

[45] C. Clark and A. Storkey, Training deep convolutional neural networks to play Go, in *Proc. 32nd Int. Conf. Machine Learning*, Lille, France, 2015, pp. 1766–1774.

[46] C. J. Maddison, A. Huang, I. Sutskever, and D. Silver, Move evaluation in Go using deep convolutional neural networks, in *Proc. 3rd Int. Conf. Learning Representations*, San Diego, CA, USA, 2015.

[47] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[48] A. G. Barto, Reinforcement learning and adaptive critic methods, in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D. A. White and D. A. Sofge, Eds. New York, NY, USA: Van Nostrand Reinhold, 1992, pp. 469–491.

[49] F. L. Lewis and D. Vrabie, Reinforcement learning and adaptive dynamic programming for feedback control, *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, 2009.

[50] F. L. Lewis, D. Liu, and G. G. Lendaris, Guest editorial-Special issue on adaptive dynamic programming and reinforcement learning in feedback control, *IEEE Trans. Syst. Man Cybern. B: Cybern*, vol. 38, no. 4, pp. 896–897, 2008.

[51] D. Liu, F. L. Lewis, and Q. Wei, Editorial special issue on adaptive dynamic programming and reinforcement learning, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 11, pp. 3944–3947, 2020.

[52] L. Buşoniu, T. De Bruin, D. Tolić, J. Kober, and I. Palunko, Reinforcement learning for control: Performance, stability, and deep approximators, *Annu. Rev. Control*, vol. 46, pp. 8–28, 2018.

[53] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis,

Optimal and autonomous control using reinforcement learning: A survey, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2042–2062, 2018.
[54] D. Liu, S. Xue, B. Zhao, B. Luo, and Q. Wei, Adaptive dynamic programming for control: A survey and recent advances, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 51, no. 1, pp. 142–160, 2021.
[55] D. Wang, M. Ha, and M. Zhao, The intelligent critic framework for advanced optimal control, *Artif. Intell. Rev.*, vol. 55, no. 1, pp. 1–22, 2022.
[56] R. S. Sutton, Learning to predict by the methods of temporal differences, *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, 1988.
[57] C. J. C. H. Watkins, Learning from delayed rewards, PhD dissertation, Cambridge Univ., Cambridge, UK, 1989.
[58] C. J. C. H. Watkins and P. Dayan, *Q*-learning, *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.
[59] A. Gosavi, Reinforcement learning: A tutorial survey and recent advances, *INFORMS J. Comput.*, vol. 21, no. 2, pp. 178–192, 2009.
[60] L. P. Kaelbling, M. L. Littman, and A. W. Moore, Reinforcement learning: A survey, *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.
[61] G. A. Rummery and M. Niranjan, *On-Line Q-Learning Using Connectionist Systems*, http://mi.eng.cam.ac.uk/reports/svr-ftp/auto-pdf/rummery_tr166.pdf, 1994.
[62] R. S. Sutton, Generalization in reinforcement learning: Successful examples using sparse coarse coding, in *Proc. 8th Int. Conf. Neural Information Processing Systems*, Denver, CO, USA, 1995, pp. 1038–1044.
[63] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1957.
[64] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. 3rd ed. Belmont, MA, USA: Athena Scientific, 2005.
[65] S. E. Dreyfus and A. M. Law, *The Art and Theory of Dynamic Programming*. New York, NY, USA: Academic Press, 1977.
[66] F. L. Lewis and V. L. Syrmos, *Optimal Control*. New York, NY, USA: Wiley, 1995.
[67] M. C. Weinstein and R. J. Zeckhauser, The optimal consumption of depletable natural resources, *Quart. J. Econ.*, vol. 89, no. 3, pp. 371–392, 1975.
[68] S. G. Papachristos, Adaptive dynamic programming in inventory control, PhD dissertation, The University of Manchester, Manchester, UK, 1977.
[69] S. Papachristos, Note-A note on the dynamic inventory problem with unknown demand distribution, *Manage. Sci.*, vol. 23, no. 11, pp. 1248–1251, 1977.
[70] S. Shields, A review of fault detection methods for large systems, *Radio Electron. Eng.*, vol. 46, no. 6, pp. 276–280, 1976.
[71] A. G. Barto, S. J. Bradtke, and S. P. Singh, Learning to act using real-time dynamic programming, *Artif. Intell.*, vol. 72, no. 1-2, pp. 81–138, 1995.
[72] J. J. Murray, C. J. Cox, and R. E. Saeks, The adaptive dynamic programming theorem, in *Stability and Control of Dynamical Systems with Applications*, D. Liu and P. J. Antsaklis, Eds. Boston, MA USA: Birkhäuser, 2003, pp. 379–394.
[73] W. H. Hausman and L. J. Thomas, Inventory control with probabilistic demand and periodic withdrawals, *Manage. Sci.*, vol. 18, no. 5-part-1, pp. 265–275, 1972.
[74] P. J. Werbos, Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research, *IEEE Trans. Syst. Man Cybern.*, vol. 17, no. 1, pp. 7–20, 1987.
[75] P. J. Werbos, Advanced forecasting methods for global crisis warning and models of intelligence, *Gen. Syst.*, vol. 22, pp. 25–38, 1977.
[76] 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, 1990, pp. 67–95.
[77] P. J. Werbos, Consistency of HDP applied to a simple reinforcement learning problem, *Neural Netw.*, vol. 3, no. 2, pp. 179–189, 1990.
[78] P. J. Werbos, Approximate dynamic programming for real-time control and neural modeling, in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D. A. White and D. A. Sofge, Eds. New York, NY, USA: Van Nostrand Reinhold, 1992, pp.493–525.
[79] D. V. Prokhorov, R. A. Santiago, and D. C. Wunsch, Adaptive critic designs: A case study for neurocontrol, *Neural Netw.*, vol. 8, no. 9, pp. 1367–1372, 1995.
[80] D. V. Prokhorov and D. C. Wunsch, Adaptive critic designs, *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, 1997.
[81] A. Al-Tamimi, M. Abu-Khalaf, and F. L. Lewis, Adaptive critic designs for discrete-time zero-sum games with application to H_∞ control, *IEEE Trans. Syst. Man Cybern. B: Cybern.*, vol. 37, no. 1, pp. 240–247, 2007.
[82] S. N. Balakrishnan and V. Biega, Adaptive-critic-based neural networks for aircraft optimal control, *J. Guid. Control Dyn.*, vol. 19, no. 4, pp. 893–898, 1996.
[83] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator, *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 764–773, 2002.
[84] C. Cox, S. Stepniewski, C. Jorgensen, R. Saeks, and C. Lewis, On the design of a neural network autolander, *Int. J. Robust Nonlinear Control*, vol. 9, no. 14, pp. 1071–1096, 1999.
[85] J. Dalton and S. N. Balakrishnan, A neighboring optimal adaptive critic for missile guidance, *Math. Comput. Modell.*, vol. 23, nos. 1–2, pp. 175–188, 1996.
[86] D. Liu, H. Javaherian, O. Kovalenko, and T. Huang, Adaptive critic learning techniques for engine torque and air-fuel ratio control, *IEEE Trans. Syst. Man Cybern. B: Cybern.*, vol. 38, no. 4, pp. 988–993, 2008.
[87] N. V. Kulkarni and K. KrishnaKumar, Intelligent engine control using an adaptive critic, *IEEE Trans. Control Syst. Technol.*, vol. 11, no. 2, pp. 164–173, 2003.
[88] D. Liu, Y. Zhang, and H. Zhang, A self-learning call admission control scheme for CDMA cellular networks, *IEEE Trans. Neural Netw.*, vol. 16, no. 5, pp. 1219–1228, 2005.
[89] J. Si, L. Yang, and D. Liu, Direct neural dynamic programming, in *Handbook of Learning and Approximate Dynamic Programming*, J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, Eds. New York, NY, USA: Wiley, 2004, pp. 125–151.
[90] S. Chakraborty and M. G. Simoes, Neural dynamic programming based online controller with a novel trim approach, *IEE Proc. Control Theory Appl.*, vol. 152, no. 1, pp. 95–104, 2005.
[91] D. Liu and H. Zhang, A neural dynamic programming approach for learning control of failure avoidance problems, *Int. J. Intell. Control Syst.*, vol. 10, no. 1, pp. 21–32, 2005.
[92] D. P. Bertsekas and J. N. Tsitsiklis, Neuro-dynamic programming: An overview, in *Proc. 34th IEEE Conf. Decision and Control*, New Orleans, LA, USA, 1995, pp. 560–564.
[93] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA, USA: Athena Scientific, 1996.
[94] 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. A: Syst. Hum.*, vol. 30, no. 1, pp. 42–51, 2000.
[95] P. Marbach, O. Mihatsch, and J. N. Tsitsiklis, Call admission control and routing in integrated services networks using neuro-dynamic programming, *IEEE J. Select. Areas Commun.*, vol. 18, no. 2, pp. 197–208, 2000.
[96] 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, 2017.
[97] C. Mu, D. Wang, and H. He, Novel iterative neural dynamic programming for data-based approximate optimal control design, *Automatica*, vol. 81, pp. 240–252, 2017.

[98] M. Aoki, On optimal and suboptimal policies in the choice of control forces for final-value systems, *IRE Trans. Autom. Control*, vol. 5, no. 3, pp. 171–178, 1960.

[99] R. Durbeck, An approximation technique for suboptimal control, *IEEE Trans. Autom. Control*, vol. 10, no. 2, pp. 144–149, 1965.

[100] R. J. Leake and R. W. Liu, Construction of suboptimal control sequences, *SIAM J. Control*, vol. 5, no. 1, pp. 54–63, 1967.

[101] F. Y. Wang and G. N. Saridis, Suboptimal control for nonlinear stochastic systems, in *Proc. 31st IEEE Conf. Decision and Control*, Tucson, AZ, USA, 1992, pp. 1856–1861.

[102] G. N. Saridis and F. Y. Wang, Suboptimal control of nonlinear stochastic systems, *Control Theory Adv. Technol.*, vol. 10, no. 4, pp. 847–871, 1994.

[103] P. Werbos, ADP: Goals, opportunities and principles, in *Handbook of Learning and Approximate Dynamic Programming*, J. Si, A. Barto, W. Powell, and D. Wunsch, Eds. New York, NY, USA: Wiley, 2004, pp. 3–44.

[104] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York, NY, USA: Wiley, 2007.

[105] P. J. Werbos, Using ADP to understand and replicate brain intelligence: The next level design, in *Proc. IEEE Int. Symp. Approximate Dynamic Programming and Reinforcement Learning*, Honolulu, HI, USA, 2007, pp. 209–216.

[106] P. J. Werbos, Foreword - ADP: The key direction for future research in intelligent control and understanding brain intelligence, *IEEE Trans. Syst. Man Cybern. B: Cybern.*, vol. 38, no. 4, pp. 898–900, 2008.

[107] X. Bai, D. Zhao, and J. Yi, Coordinated multiple ramps metering based on neuro-fuzzy adaptive dynamic programming, in *Proc. Int. Joint Conf. Neural Networks*, Atlanta, GA, USA, 2009, pp. 241–248.

[108] Y. Zhu, D. Zhao, and H. He, Integration of fuzzy controller with adaptive dynamic programming, in *Proc. 10th World Congress on Intelligent Control and Automation*, Beijing, China, 2012, pp. 310–315.

[109] H. Zhang, J. Zhang, G. H. Yang, and Y. Luo, Leader-based optimal coordination control for the consensus problem of multiagent differential games via fuzzy adaptive dynamic programming, *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 1, pp. 152–163, 2015.

[110] R. E. Saeks, C. J. Cox, K. Mathia, and A. J. Maren, Asymptotic dynamic programming: Preliminary concepts and results, in *Proc. IEEE Int. Conf. Neural Networks*, Houston, TX, USA, 1997, pp. 2273–2278.

[111] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.

[112] J. M. Zurada, *Introduction to Artificial Neural Systems*. St. Paul, MN, USA: West, 1992.

[113] D. Liu, X. Xiong, and Y. Zhang, Action-dependent adaptive critic designs, in *Proc. Int. Joint Conf. Neural Networks*, Washington, DC, USA, 2001, pp. 990–995.

[114] G. G. Lendaris and C. Paintz, Training strategies for critic and action neural networks in dual heuristic programming method, in *Proc. IEEE Int. Conf. Neural Networks*, Houston, TX, USA, 1997, pp. 712–717.

[115] J. Si and Y. T. Wang, Online learning control by association and reinforcement, *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, 2001.

[116] H. He, Z. Ni, and J. Fu, A three-network architecture for on-line learning and optimization based on adaptive dynamic programming, *Neurocomputing*, vol. 78, no. 1, pp. 3–13, 2012.

[117] R. Padhi, N. Unnikrishnan, X. Wang, and S. N. Balakrishnan, A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems, *Neural Netw.*, vol. 19, no. 10, pp. 1648–1660, 2006.

[118] B. Lincoln and A. Rantzer, Relaxing dynamic programming, *IEEE Trans. Autom. Control*, vol. 51, no. 8, pp. 1249–1260, 2006.

[119] A. Rantzer, Relaxed dynamic programming in switching systems, *IEE Proc. Control Theory Appl.*, vol. 153, no. 5, pp. 567–574, 2006.

[120] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof, *IEEE Trans. Syst. Man Cybern. B: Cybern.*, vol. 38, no. 4, pp. 943–949, 2008.

[121] F. Y. Wang, N. Jin, D. Liu, and Q. Wei, Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound, *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 24–36, 2011.

[122] D. Liu, D. Wang, D. Zhao, Q. Wei, and N. Jin, Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming, *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 3, pp. 628–634, 2012.

[123] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming, *Automatica*, vol. 48, no. 8, pp. 1825–1832, 2012.

[124] D. Liu, D. Wang, and X. Yang, An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs, *Inf. Sci.*, vol. 220, pp. 331–342, 2013.

[125] D. Liu and Q. Wei, Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 3, pp. 621–634, 2014.

[126] 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, 2015.

[127] 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, 2014.

[128] Q. Zhao, H. Xu, and S. Jagannathan, Near optimal output feedback control of nonlinear discrete-time systems based on reinforcement neural network learning, *IEEE/CAA J. Autom. Sin.*, vol. 1, no. 4, pp. 372–384, 2014.

[129] X. Zhong, H. He, H. Zhang, and Z. Wang, Optimal control for unknown discrete-time nonlinear Markov jump systems using adaptive dynamic programming, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2141–2155, 2014.

[130] A. Sahoo, H. Xu, and S. Jagannathan, Near optimal event-triggered control of nonlinear discrete-time systems using neurodynamic programming, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1801–1815, 2016.

[131] H. Li and D. Liu, Optimal control for discrete-time affine non-linear systems using general value iteration, *IET Control Theory Appl.*, vol. 6, no. 18, pp. 2725–2736, 2012.

[132] 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, 2015.

[133] D. Liu and Q. Wei, Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems, *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 779–789, 2013.

[134] Q. Wei and D. Liu, Numerical adaptive learning control scheme for discrete-time non-linear systems, *IET Control Theory Appl.*, vol. 7, no. 11, pp. 1472–1486, 2013.

[135] Q. Wei and D. Liu, Stable iterative adaptive dynamic programming algorithm with approximation errors for discrete-time nonlinear systems, *Neural Comput. Appl.*, vol. 24, no. 6, pp. 1355–1367, 2014.

[136] Q. Wei, D. Liu, and X. Yang, Infinite horizon self-learning optimal control of nonaffine discrete-time nonlinear systems, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 4, pp. 866–879, 2015.

[137] Q. Wei, D. Liu, and Y. Xu, Neuro-optimal tracking control for a class of discrete-time nonlinear systems via generalized value iteration adaptive dynamic programming approach, *Soft Comput.*, vol. 20, no. 2, pp. 697–706, 2016.

[138] Q. Wei, F. Y. Wang, D. Liu, and X. Yang, Finite-approximation-

error-based discrete-time iterative adaptive dynamic programming, *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2820–2833, 2014.

[139] P. Yan, D. Wang, H. Li, and D. Liu, Error bound analysis of *Q*-function for discounted optimal control problems with policy iteration, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 47, no. 7, pp. 1207–1216, 2017.

[140] B. Luo, Y. Yang, H. N. Wu, and T. Huang, Balancing value iteration and policy iteration for discrete-time control, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 11, pp. 3948–3958, 2020.

[141] M. Ha, D. Wang, and D. Liu, A novel value iteration scheme with adjustable convergence rate, *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2022.3143527.

[142] Y. Zhu, D. Zhao, and X. Li, Iterative adaptive dynamic programming for solving unknown nonlinear zero-sum game based on online data, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 714–725, 2017.

[143] C. Li, D. Liu, and D. Wang, Data-based optimal control for weakly coupled nonlinear systems using policy iteration, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 48, no. 4, pp. 511–521, 2018.

[144] H. Zhang, Y. Liu, G. Xiao, and H. Jiang, Data-based adaptive dynamic programming for a class of discrete-time systems with multiple delays, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 2, pp. 432–441, 2020.

[145] N. Lin, R. Chi, and B. Huang, Event-triggered model-free adaptive control, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 51, no. 6, pp. 3358–3369, 2021.

[146] B. Luo, Y. Yang, and D. Liu, Policy iteration *Q*-learning for data-based two-player zero-sum game of linear discrete-time systems, *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3630–3640, 2021.

[147] Q. Wei, L. Zhu, R. Song, P. Zhang, D. Liu, and J. Xiao, Model-free adaptive optimal control for unknown nonlinear multiplayer nonzero-sum game, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 879–892, 2022.

[148] M. Farjadnasab and M. Babazadeh, Model-free LQR design by Q-function learning, *Automatica*, vol. 137, p. 110060, 2022.

[149] C. Mu, D. Wang, and H. He, Data-driven finite-horizon approximate optimal control for discrete-time nonlinear systems using iterative HDP approach, *IEEE Trans. Cybern.*, vol. 48, no. 10, pp. 2948–2961, 2018.

[150] S. Al-Dabooni and D. C. Wunsch, An improved *N*-step value gradient learning adaptive dynamic programming algorithm for online learning, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1155–1169, 2020.

[151] B. Luo, D. Liu, H. N. Wu, D. Wang, and F. L. Lewis, Policy gradient adaptive dynamic programming for data-based optimal control, *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3341–3354, 2017.

[152] J. Ye, Y. Bian, B. Luo, M. Hu, B. Xu, and R. Ding, Costate-supplement ADP for model-free optimal control of discrete-time nonlinear systems, *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2022.3172126.

[153] Y. Li, Z. Hou, Y. Feng, and R. Chi, Data-driven approximate value iteration with optimality error bound analysis, *Automatica*, vol. 78, pp. 79–87, 2017.

[154] Y. Li, C. Yang, Z. Hou, Y. Feng, and C. Yin, Data-driven approximate Q-learning stabilization with optimality error bound analysis, *Automatica*, vol. 103, pp. 435–442, 2019.

[155] H. Zhang, K. Zhang, Y. Cai, and J. Han, Adaptive fuzzy fault-tolerant tracking control for partially unknown systems with actuator faults via integral reinforcement learning method, *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 10, pp. 1986–1998, 2019.

[156] Y. Cao, Y. Song, and C. Wen, Practical tracking control of perturbed uncertain nonaffine systems with full state constraints, *Automatica*, vol. 110, p. 08608, 2019.

[157] C. Chen, H. Modares, K. Xie, F. L. Lewis, Y. Wan, and S. Xie, Reinforcement learning-based adaptive optimal exponential tracking control of linear systems with unknown dynamics, *IEEE Trans. Autom. Control*, vol. 64, no. 11, pp. 4423–4438, 2019.

[158] M. Ha, D. Wang, and D. Liu, Data-based nonaffine optimal tracking control using iterative DHP approach, *IFAC-Papers-OnLine*, vol. 53, no. 2, pp. 4246–4251, 2020.

[159] K. Zhang, H. Zhang, Y. Mu, and C. Liu, Decentralized tracking optimization control for partially unknown fuzzy interconnected systems via reinforcement learning method, *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 4, pp. 917–926, 2021.

[160] F. Liu, C. Jiang, and W. Xiao, Multistep prediction-based adaptive dynamic programming sensor scheduling approach for collaborative target tracking in energy harvesting wireless sensor networks, *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 693–704, 2021.

[161] H. Dong, X. Zhao, and B. Luo, Optimal tracking control for uncertain nonlinear systems with prescribed performance via critic-only ADP, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 1, pp. 561–573, 2022.

[162] B. Luo, D. Liu, T. Huang, and J. Liu, Output tracking control based on adaptive dynamic programming with multistep policy evaluation, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 49, no. 10, pp. 2155–2165, 2019.

[163] C. Li, J. Ding, F. L. Lewis, and T. Chai, A novel adaptive dynamic programming based on tracking error for nonlinear discrete-time systems, *Automatica*, vol. 129, p. 109687, 2021.

[164] M. Ha, D. Wang, and D. Liu, Discounted iterative adaptive critic designs with novel stability analysis for tracking control, *IEEE/CAA J. Autom. Sin.*, vol. 9, no. 7, pp. 1262–1272, 2022.

[165] W. Xue, P. Kolaric, J. Fan, B. Lian, T. Chai, and F. L. Lewis, Inverse reinforcement learning in tracking control based on inverse optimal control, *IEEE Trans. Cybern.*, vol. 52, no. 10, pp. 10570–10581, 2022.

[166] W. Zhang, K. Song, X. Rong, and Y. Li, Coarse-to-fine UAV target tracking with deep reinforcement learning, *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 4, pp. 1522–1530, 2019.

[167] Y. Hu, W. Wang, H. Liu, and L. Liu, Reinforcement learning tracking control for robotic manipulator with kernel-based dynamic model, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3570–3578, 2020.

[168] R. Wu, Z. Yao, J. Si, and H. H. Huang, Robotic knee tracking control to mimic the intact human knee profile based on actor-critic reinforcement learning, *IEEE/CAA J. Autom. Sin.*, vol. 9, no. 1, pp. 19–30, 2022.

[169] S. Cao, L. Sun, J. Jiang, and Z. Zuo, Reinforcement learning-based fixed-time trajectory tracking control for uncertain robotic manipulators with input saturation, *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2021.3116713.

[170] N. Wang, Y. Gao, H. Zhao, and C. K. Ahn, Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 3034–3045, 2021.

[171] S. A. A. Rizvi, A. J. Pertzborn, and Z. Lin, Reinforcement learning based optimal tracking control under unmeasurable disturbances with application to HVAC systems, *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2021.3085358.

[172] S. Song, M. Zhu, X. Dai, and D. Gong, Model-free optimal tracking control of nonlinear input-affine discrete-time systems via an iterative deterministic Q-learning algorithm, *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2022.3178746.

[173] M. Lin, B. Zhao, and D. Liu, Policy gradient adaptive critic designs for model-free optimal tracking control with experience replay, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 6, pp. 3692–3703, 2022.

[174] S. Li, P. Durdevic, and Z. Yang, Model-free H_∞ tracking control for de-oiling hydrocyclone systems via off-policy reinforcement learning, *Automatica*, vol. 133, p. 109862, 2021.

[175] W. Sun, X. Wang, and C. Zhang, A model-free control strategy for vehicle lateral stability with adaptive dynamic programming, *IEEE Trans. Ind. Electron.*, vol. 67, no. 12, pp. 10693–10701, 2020.

[176] A. Heydari, Stability analysis of optimal adaptive control under

value iteration using a stabilizing initial policy, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4522–4527, 2018.

[177] A. Heydari, Stability analysis of optimal adaptive control using value iteration with approximation errors, *IEEE Trans. Autom. Control*, vol. 63, no. 9, pp. 3119–3126, 2018.

[178] 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, 2016.

[179] A. Heydari, Theoretical and numerical analysis of approximate dynamic programming with approximation errors, *J. Guid., Control Dyn.*, vol. 39, no. 2, pp. 301–311, 2016.

[180] M. Ha, D. Wang, and D. Liu, Offline and online adaptive critic control designs with stability guarantee through value iteration, *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2021.3107801.

[181] M. Ha, D. Wang, and D. Liu, Generalized value iteration for discounted optimal control with stability analysis, *Syst. Control Lett.*, vol. 147, p. 104847, 2021.

[182] K. G. Vamvoudakis and F. L. Lewis, Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem, *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[183] M. Ha, D. Wang, and D. Liu, Neural-network-based discounted optimal control via an integrated value iteration with accuracy guarantee, *Neural Netw.*, vol. 144, pp. 176–186, 2021.

[184] S. Al-Dabooni and D. C. Wunsch, Online model-free *n*-step HDP with stability analysis, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1255–1269, 2020.

[185] H. Zhang, C. Qin, B. Jiang, and Y. Luo, Online adaptive policy learning algorithm for H_∞ state feedback control of unknown affine nonlinear discrete-time systems, *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2706–2718, 2014.

[186] Y. Sokolov, R. Kozma, L. D. Werbos, and P. J. Werbos, Complete stability analysis of a heuristic approximate dynamic programming control design, *Automatica*, vol. 59, pp. 9–18, 2015.

[187] S. Al-Dabooni and D. Wunsch, The boundedness conditions for model-free HDP(λ), *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 7, pp. 1928–1942, 2019.

[188] J. W. Kim, T. H. Oh, S. H. Son, D. H. Jeong, and J. M. Lee, Convergence analysis of the deep neural networks based globalized dual heuristic programming, *Automatica*, vol. 122, p. 109222, 2020.

[189] R. A. Santiago and P. Werbos, New progress towards truly brain-like intelligent control, in *Proc. World Congress on Neural Networks*, San Diego, CA, 1994, pp. 27–33.

[190] P. J. Werbos, Intelligence in the brain: A theory of how it works and how to build it, *Neural Netw.*, vol. 22, no. 3, pp. 200–212, 2009.