Proximal Policy Optimization With Policy Feedback

Yang Gu, Yuhu Cheng, *Member, IEEE*, C. L. Philip Chen, *Fellow, IEEE*, and Xuesong Wang, *Member, IEEE*

***Abstract*—Proximal policy optimization (PPO) is a deep reinforcement learning algorithm based on the actor–critic (AC) architecture. In the classic AC architecture, the Critic (value) network is used to estimate the value function while the Actor (policy) network optimizes the policy according to the estimated value function. The efficiency of the classic AC architecture is limited due that the policy does not directly participate in the value function update. The classic AC architecture will make the value function estimation inaccurate, which will affect the performance of the PPO algorithm. For improvement, we designed a novel AC architecture with policy feedback (AC-PF) by introducing the policy into the update process of the value function and further proposed the PPO with policy feedback (PPO-PF). For the AC-PF architecture, the policy-based expected (PBE) value function and discount reward formulas are designed by drawing inspiration from expected Sarsa. In order to enhance the sensitivity of the value function to the change of policy and to improve the accuracy of PBE value estimation at the early learning stage, we proposed a policy update method based on the clipped discount factor. Moreover, we specifically defined the loss functions of the policy network and value network to ensure that the policy update of PPO-PF satisfies the unbiased estimation of the trust region. Experiments on Atari games and control tasks show that compared to PPO, PPO-PF has faster convergence speed, higher reward, and smaller variance of reward.**

***Index Terms*—Actor–critic (AC), clipped discount factor, policy feedback, proximal policy optimization (PPO), value function.**

I. Introduction

REINFORCEMENT learning (RL), as an important branch of machine learning, aims to solve the policy optimization problem via collecting information from the agent's interaction with the environment. AlphaGo's achievements in the field of Go have opened the door to the development of DRL. The success of OpenAI Five in the field of e-sports has further promoted the development boom. In recent years, a large number of DRL algorithms have been proposed and widely used in video games [1], [2], visual object tracking [3], natural language processing [4], scheduling optimization [5], autonomous driving [6], [7] and other fields. DRL has become one of the hot topics of artificial intelligence research.

DRL algorithms can be divided into two categories: 1) value-based and 2) policy-based DRL algorithms. Value-based DRL algorithms originate from the classic deep Q network (DQN) [8]. The basic idea of DQN is to first design a value network to estimate Q-value, and then improve the current policy based on the learned value function. Based on DQN, a series of improved DRL algorithms is proposed. For example, Schaul *et al.* [9] proposed the prioritized experience replay technique. The samples in the experience buffer are assigned with different priorities that are determined by TD error, which makes the samples that are more favorable to the value network update have a larger probability of being sampled, thus effectively improving the learning efficiency. Double DQN [10] uses different networks to select actions and evaluate actions. The independence between the two networks is strengthened to solve the value function overestimation problem existed in DQN. The approximate policy-based accelerated DQN (APA-DQN) proposed by Wang *et al.* [11] reduced the accumulative error of the action-value function update and is proved to be convergent even with a more aggressive learning rate, thus has a faster learning speed. Noisy DQN [12] enhances the exploration ability of agent by adding noise to network parameters to increase the variation of value network. Distributed DQN [13] converts the estimation of Q-value into the estimation of Q-value distribution. Dueling DQN [14] decomposes the action-value function into a state-value function and advantage function to improve the function approximation effect and, meantime, the construction of the advantage function also provides the direction for implementing actor–critic (AC) architecture. However, value-based DRL algorithms have limitations [15]: unable to solve large-scale action space RL problems and so do continuous-action problems. Because of the greedy action selection, value-based RL algorithms cannot deal with stochastic policy problems.

Differ from value-based DRL, policy-based DRL directly calculates the policy to maximize the discounted reward. The most commonly used policy-based DRL is the policy gradient (PG) algorithm [16]. The stochastic PG (SPG) [17] provides an effective solution for dealing with continuous-action and stochastic policy problems. The deterministic PG (DPG) [18] solves the problem that SPG requires too many

Manuscript received April 22, 2021; accepted July 16, 2021. Date of publication July 29, 2021; date of current version June 16, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61976215, Grant 61772532, and Grant U1813203. This article was recommended by Associate Editor K. G. Vamvoudakis. *(Corresponding author: Xuesong Wang.)*

Yang Gu, Yuhu Cheng, and Xuesong Wang are with the Engineering Research Center of Intelligent Control for Underground Space, Ministry of Education and the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China (e-mail: guyang@cumt.edu.cn; chengyuhu@163.com; wangxuesongcumt@163.com).

C. L. Philip Chen is with the College of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau, China (e-mail: philip.chen@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2021.3098451.

Digital Object Identifier 10.1109/TSMC.2021.3098451

samples to calculate policy gradient, which can reduce the complexity of PG calculation. Natural PG [19] uses natural gradients to achieve global optimization of policy and improve the efficiency of PG algorithms. The trust region policy optimization (TRPO) [20] uses the Kullback–Leibler (KL) divergence between the old and new policy network outputs to construct the hard constraint term, i.e., the trust region constraint. The monotonic optimization of PG is guaranteed such that the estimation of PG is unbiased. The proximal policy optimization (PPO) [21] simplifies the TRPO algorithm by replacing the penalty term with the KL divergence and using a clip-based truncation operation to reduce the update magnitude of the policy. The expected PG (EPG) [22] reduces the variance of gradient estimation without requiring deterministic policies by drawing inspiration from expected Sarsa [15]. Chow *et al.* [23] conducted EPG exploration on bounded RL problems and achieved good experimental results. The essence of EPG is to apply a constraint on the size of policy update by evaluating the policy. The theoretical derivation of this constraint is rigorous, but its programming implementation is more complicated and the update effect of PG is not as good as PPO. Although the PG algorithm has strong versatility and more stable training process, it has its inherent shortcomings: large variance of trajectories [15], low sample utilization, and being prone to getting stuck in local optima.

To compensate for the defects of value-based and policy-based algorithms, researchers have introduced the AC architecture into the field of DRL. As a combination of value-based and policy-based algorithms, AC has the advantages of both algorithms [24], [25]: the value function from Critic makes the policy update with lower trajectory variance; the policy from Actor makes it feasible to handle continuous-action tasks and improve the versatility of algorithm. Lillicrap *et al.* [26] proposed deep DPG (DDPG) by combining DPG with DQN. In DDPG, actions obtained from the policy network and states observed from the environment are taken as the input of DQN network, and the partial derivative of the value function with respect to action is used to calculate PG and update policy network. DDPG can solve the difficulty of estimating SPG in continuous action space by using DPG, which makes more stable and high-quality learning effects can be achieved by DDPG on both discrete-action and continuous-action tasks than that by DQN or DPG alone. However, the Actor network in DDPG directly outputs actions, weakening the exploration ability of agent. To enhance the exploration ability of agent, the agent selects an action according to the action probability distribution obtained by the Actor network. Nachum *et al.* [27] proposed an off-policy trust region method, trust path consistency learning (Trust-PCL), by adding an entropy regularization to the value function. The pathwise consistency [28] is optimized by sampling on-policy and off-policy experiences. Experiments on a number of continuous-action control tasks show that Trust-PCL can maintain optimization stability while improve sample utilization. The soft Q-learning proposed by Haarnoja *et al.* [29] used the energy-based policies for continuous states and actions as the bias, satisfying the soft Bellman equation and enhancing the exploration ability of agent. According to the equivalence between PG and soft Q-learning [30], [31], Haarnoja *et al.* [32] proposed soft AC (SAC) by combining soft Q-learning and DDPG.

However, unlike the above-mentioned off-policy AC algorithms, the popular AC architecture is generally on-policy. This is mainly because TRPO believes that [20]: only the on-policy algorithm can guarantee that the policy estimation is unbiased. But on-policy algorithm has defects of low sample utilization and large sample correlation. The on-policy asynchronous advantage AC (A3C) [33] uses asynchronous exploration to enhance the independence between training samples and expand the sample size, improving the learning accuracy and speed. Ilyas *et al.* [34] pointed out that: 1) TRPO and PPO cannot perform deterministic policy iteration; 2) the sample utilization of TRPO and PPO is low; and 3) if the sample size is not large enough, the trust region constraint for policy update will not be satisfied. Therefore, the on-policy AC algorithm has a gap between the theoretical and actual working effects. In summary, the following observations can be drawn: 1) the update manner of on-policy deep AC algorithms is that the target value network is updated based on sampled data, and the policy network is updated according to the PG obtained by the output of value network and 2) the off-policy deep AC algorithm pays more attention to designing the policy as a bias and further introducing it into the update process of value function to enhance the algorithm stability, which embodies the idea of feedback control. Therefore, we consider introducing the idea of feedback control into on-policy deep AC algorithm, i.e., how to introduce policy more directly into the value function update.

Discount factor is the basis of the RL model, which plays a very important role from the earliest biological experiment to the proposal of RL algorithms. Sutton and Barto [15] defined the discount factor in the range [0, 1] to ensure the convergence of RL algorithms. In the process of DRL development, many scholars have studied the selection of the discount factor. Thomaz and Breazeal [35] thought that the discount factor should be 0.75, while Tenorio-Gonzalez *et al.* [36] recommended it to be 0.9. OpenAI [37] settled the discount factor as 0.99 to deal with complex learning problems and Liu *et al.* [38] suggested that the discount factor should be closer to 1. The larger the discount factor is, the more sensitive the algorithm is to long-term rewards. Knox and Stone [39] believed that when dealing with RL problems of human reward, a discount factor of 0 can yield a better result. In summary, it can be found that there is no uniform standard for setting the value of discount factor. Francois-Lavet *et al.* [40] designed a discount factor that changes over time. Experiments in [40] show that for the PPO algorithm, higher training efficiency can be obtained by gradually reducing the learning rate when the discount factor is 0.99. Yoshida *et al.* [41] designed the state-dependent discount factor that changes according to the state and based on which derived a novel ExQ-learning algorithm. Experiments on the grid world verify that the proposed ExQ-learning performs better than that with a constant discount factor. Obviously, the state-dependent discount factor is not applicable for complex environment due that the division of large-scale or continuous state space is very difficult. It can be known from the analysis of research status of discount factor that constant discount

factor cannot make all RL algorithms yield state-of-the-art performance and that the design of variable discount factor is still remained to be solved.

In order to improve the accuracy of value estimation, based on the on-policy deep AC architecture and PPO algorithm, we proposed a novel PPO with the policy feedback (PPO-PF) algorithm by treating the value of the policy as the discount factor and making it directly participate in the update of value function. The main contributions of this article are outlined as follows.

1) We designed a novel AC architecture with policy feedback (AC-PF). In the AC-PF, the value function estimated by the Critic network is used to perform policy update and, in the meantime, the policy from the Actor network is used to guide the value function update, forming a collaborative update manner of value function and policy.
2) By drawing inspiration from expected Sarsa, we designed a policy-based expected (PBE) value function and discount reward formulas.
3) In view of such a common sense that the discount factor for complex RL problems is generally large, we defined the clipped discount factor and accordingly proposed a policy update method based on clipped discount factor.
4) We specifically defined the loss functions of the policy network and value network to ensure that the policy update of PPO-PF satisfies the unbiased estimation of the trust region.

The structure of this article is organized as follows: Section II introduces related work, including expected Sarsa and PPO. The details of the proposed AC-PF architecture, PBE value function, clipped discount factor, policy update, and pseudocode of PPO-PF are described in Section III. Experimental results of PPO-PF on the Atari and OpenAI Gym platforms are reported in Section IV. Finally, Section V gives the conclusion.

II. Related Work

A. Expected Sarsa

In general, the goal of RL is to find an optimal policy $\pi^*(s, a)$ to maximize the following expected discount reward:

$$R_t = E\left[\sum_{i=0}^{T} \gamma^i r_{t+i}\right] \tag{1}$$

where s is the state, a is the action, r is the immediate reward, T is the time step corresponding to the final state on the trajectory, and γ is the discount factor. Compared with on-policy Sarsa algorithm, we prefer to use off-policy Q-learning to maximize the discount reward. In Q-learning, the action-value is updated according to

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\Big(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big) \tag{2}$$

where α represents the learning rate and $Q(s_t, a_t)$ is the Q-value of state–action pair (s_t, a_t). Q-learning usually assumes that the agent greedily selects actions, i.e., only selects the action with the largest Q-value and the selection probability for other actions is 0, thus ensuring the convergence of Q-learning. Compared to the Sarsa, off-policy Q-learning requires shorter training time and has a larger probability of jumping out of local optima. However, if the agent samples actions according to the probability model of Q-value instead of greedy selection, the estimation error of the Q-value with the off-policy technique will increase. Expected Sarsa takes the sampling probability into account and guides the value function update by estimating an unbiased Q-value [15]

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\Bigg(r_t + \gamma \sum_a \pi(s_{t+1}, a) Q(s_{t+1}, a) - Q(s_t, a_t)\Bigg) \tag{3}$$

where π is the policy used to estimate the expected Q-value. It can be seen that the expected Sarsa update will become Q-learning only when the agent greedily selects actions.

B. Proximal Policy Optimization

TRPO is a policy optimizing algorithm, with guaranteed monotonic improvement [20]. Letting $c_t(\theta) = ([\pi_\theta(a_t|s_t)]/[\pi_{\theta_{\text{old}}}(a_t|s_t)])$ denote the probability ratio, the objective function of TRPO is [20]

$$\begin{cases} L_\pi(\theta) = \max_\theta \quad E[c_t(\theta) A_t] \\ \text{s.t.} \quad E\big[KL\big[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\big]\big] \le \delta \end{cases} \tag{4}$$

where θ is the policy parameter vector, $\pi_{\theta_{\text{old}}}$ represents the old policy, V_t is the state-value function, and A_t is the advantage function. PPO has some of the benefits of TRPO, but PPO is much simpler to implement, more general, and has better sample complexity [21]. The advantage function of TRPO and PPO can be drawn by $A_t = Q_t - V_t = r_t + \gamma V_{t+1} - V_t$, where Q_t is the Q-value of state s_t. V_{t+1} is the state-value function of next state s_{t+1}. $KL[\cdot]$ represents KL divergence and δ is a hyperparameter used to constrain the KL divergence. Since TRPO uses a hard constraint to calculate the policy gradient, it is difficult to choose a single constraint value that performs well across different problems. In contrast, PPO replaces a clip operation with the KL divergence [41]

$$L_\pi^{\text{clip}}(\theta) = E\big[\min\ (c_t(\theta) A_t, \text{clip}(c_t(\theta), 1-\varepsilon, 1+\varepsilon) A_t)\big] \tag{5}$$

where hyperparameter ε is used to clip the value of $c_t(\theta)$. $\text{clip}(c_t(\theta), 1-\varepsilon, 1+\varepsilon)$ removes the incentive for moving $c_t(\theta)$ outside of the interval $[1-\varepsilon, 1+\varepsilon]$.

III. PPO With Policy Feedback

A. AC Architecture With Policy Feedback

In the classic AC architecture, the Critic network realizes the estimation of the value function and the Actor network performs policy improvement given the estimated value function. In other words, the estimated value function generates the baseline to make the policy evaluation more stable [42]. However, the policy does not directly participate in the value function update. Therefore, this kind of on-policy AC architecture will increase the instability of DRL algorithms based on

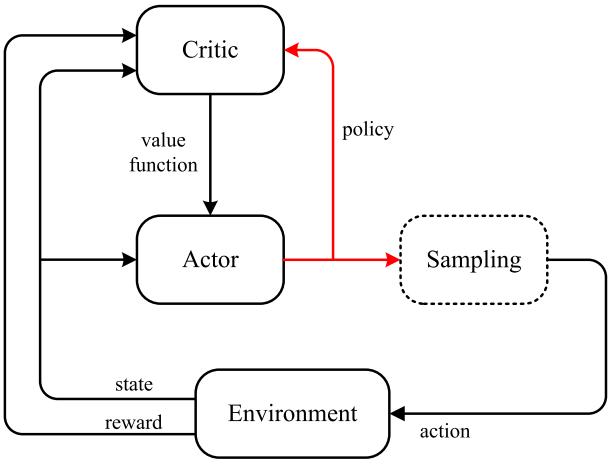


Fig. 1. On-policy AC-PF.

AC to some extent. Aiming at the above problem, we designed an on-policy AC-PF shown in Fig. 1 by introducing the policy into the update of the value function, thus implementing the collaborative update of value function and policy.

In the classic on-policy AC architecture, the output of the Actor network, i.e., policy, only affects the sampled data used for value function update. Therefore, the policy update cannot be realized until the value function was updated with a large amount of sampled data for a period of time. By directly making the policy involved in the value function update, the Critic network can quickly reflect the change of policy. Similarly, the Actor network can quickly perform policy update according to the value function estimated by the Critic network, thereby improving the exploration efficiency of Actor network. In addition, compared to the classic on-policy AC architecture, the connection between the Critic and Actor networks in AC-PF is much closer, so that the solutions that satisfy the value function and policy update loss functions can be found faster, thereby reducing the estimation error of the value function and enhancing the stability of the entire AC architecture. The core problem of constructing AC-PF is how to design the policy-based value function iteration method.

B. Policy-Based Expected Value Function

Compared to other value function update rules, the expected Sarsa exploits knowledge about stochasticity in the behavior policy to perform updates with lower variance. The update rule of one-step expected Sarsa under policy π is [15]

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma \sum_{a} \pi(s_{t+1}, a) Q(s_{t+1}, a). \tag{6}$$

Suppose $a^* = \arg\max_a Q(s_{t+1}, a)$, then the update rule of one-step Q-learning is

$$Q^{O}(s_t, a_t) = r(s_t, a_t) + \gamma Q\big(s_{t+1}, a^*\big). \tag{7}$$

The following observation can be obtained from formulas (6) and (7): if and only if the probability $\pi(s_{t+1}, a^*) = 1$ and probabilities of other actions are 0, $Q^{\pi}(s_t, a_t)$ and $Q^{O}(s_t, a_t)$ will be identical. That is, Q-learning can be regarded as a simplified version of expected Sarsa. The samples required by the expected Sarsa should cover the entire action space. Therefore, the expected Sarsa is not suitable for dealing with complex problems. However, it can be seen from formula (6) that the expected Sarsa can realize the policy feedback operation on both value function and policy updates. Therefore, by drawing inspiration from the expected Sarsa, we introduce the policy into the update process of value function, and get the PBE action-value function

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \pi(s_{t+1}, a_{t+1}) Q(s_{t+1}, a_{t+1}). \tag{8}$$

Based on the assumption of action a^*, we expand formula (6) and get

$$\sum_{a} \pi(s_{t+1}, a) Q(s_{t+1}, a) = \pi\big(s_{t+1}, a^*\big) Q\big(s_{t+1}, a^*\big) + \sum_{a \in A, a \neq a^*} \pi(s_{t+1}, a) Q(s_{t+1}, a) \tag{9}$$

where A represents the action space. Because the agent greedily selects action in the classic Q-learning, the error between expected Sarsa and Q-learning can be written as the following formula [43]:

$$\begin{aligned} \left|Q^{\pi} - Q^{O}\right| &\triangleq \left|Q^{\pi}(s_t, a_t) - Q^{O}(s_t, a_t)\right| \\ &= \left|\gamma \pi\big(s_{t+1}, a^*\big) Q\big(s_{t+1}, a^*\big) + \gamma \sum_{a \in A, a \neq \mu} \times \pi(s_{t+1}, a) Q(s_{t+1}, a) - \gamma Q\big(s_{t+1}, a^*\big)\right| \\ &\leq \gamma \left|\pi\big(s_{t+1}, a^*\big) Q\big(s_{t+1}, a^*\big) - Q\big(s_{t+1}, a^*\big)\right| + \gamma \sum_{a \in A, a \neq a^*} \left|\pi(s_{t+1}, a) Q(s_{t+1}, a)\right|. \end{aligned} \tag{10}$$

With the same reason, the error between the PBE action-value function and the expected Sarsa is

$$\begin{aligned} \left|Q^{\pi} - Q\right| &\triangleq \left|Q^{\pi}(s_t, a_t) - Q(s_t, a_t)\right| \\ &= \left|Q^{\pi} - \gamma \pi(s_{t+1}, a_{t+1}) Q(s_{t+1}, a_{t+1})\right| \\ &\leq \gamma \sum_{a \in A, a \neq a_{t+1}} \left|\pi(s_{t+1}, a) Q(s_{t+1}, a)\right| \end{aligned} \tag{11}$$

According to formulas (10) and (11), the following three theorems will hold.

Theorem 1: If $a_{t+1} = a^*$, $P(\sup|Q^{\pi} - Q| \leq \sup|Q^{\pi} - Q^{O}|) = 1$.

Theorem 2: If $\pi(s_{t+1}, a^*) \leq 0.5$, $P(\sup|Q^{\pi} - Q| \leq \sup|Q^{\pi} - Q^{O}|) = 1$.

Theorem 3: If $\pi(s_{t+1}, a^*) > 0.5$, $P(\sup|Q^{\pi} - Q| \leq \sup|Q^{\pi} - Q^{O}|) > \pi(s_{t+1}, a^*)$.

It can be observed from Theorem 3 that $P(\sup|Q^{\pi} - Q| \leq \sup|Q^{\pi} - Q^{O}|)$ will converge to 1 with a higher speed than $\pi(s_{t+1}, a^*)$. With Theorems 2 and 3, $\sup|Q^{\pi} - Q| \leq \sup|Q^{\pi} - Q^{O}|$ holds for most of the process in value update. That is, the PBE action-value iteration is closer to the expected Sarsa than Q-learning.

The PBE discount reward can be derived by analyzing the PBE action-value function

$$\tilde{R}(s_t) = \sum_{i=t}^{T} \gamma^{i-t} r_i \prod_{j=t}^{i} \pi\left(s_j, a_j\right). \tag{12}$$

Formula (12) can be understood as an expected estimation of discount reward with respect to truncated trajectory and discounted-ergodic properties [44]. Let a policy of specific state–action on trajectory as the probability density, we can get the following expected discount reward for the corresponding trajectory:

$$E_{\tau\sim\pi}\left[\sum_{i=0}^{T} \gamma^i r_i\right] = \int_s \left[\sum_{i=0}^{T} \gamma^i \prod_{j=0}^{i} \pi\left(s_j, a_j\right) r_i\right] ds \tag{13}$$

where τ is the trajectory and $E_{\tau\sim\pi}[\cdot]$ is the expectation on all trajectories generated from policy π. When the state space is discrete, the expected discount reward is

$$E_{\tau\sim\pi}\left[\sum_{i=0}^{T} \gamma^i r_i\right] = \sum_{i=0}^{T} \gamma^i \prod_{j=0}^{i} \pi\left(s_j, a_j\right) r_i. \tag{14}$$

Formula (14) is a special form of formula (12) when $t = 0$. Therefore, the essence of formula (12) is an approximation of the expected discount reward for specific state on the trajectory. The policy is introduced into the process of value function update, thus reducing the estimation error of state-value.

The policy can be seen as the preference of the DRL agent to select actions in the environment. Through PBE value update, the agent can adjust the discount reward according to its own preferences, so that it is more inclined to explore the direction of its own interest in exploration, instead of the traditional exploration methods. The value estimation error with PBE method is small, so PBE will also accelerate the update process of value function.

C. Clipped Discount Factor

The policy will not be optimal at the beginning of agent learning. $\pi(s, a)$ may be very small and the contribution of subsequent immediate rewards to PBE discount reward can be almost ignored. Therefore, in order to enhance the exploration ability of agent at the early learning stage, it is necessary to forcibly increase the probability of selecting action. As mentioned in [40], DRL algorithms will have a better learning effect when the discount factor reaches 0.99. Experiments have shown that better discount factors generally approach 1 in [38]. Based on the above observations, we apply a clip operation to enlarge the effect of policy on value function update. In order to improve the accuracy of PBE value estimation, the policy-based clipped discount factor is defined as

$$\gamma_{\text{clip}}(s, a; \eta) = \text{clip}(\pi(s, a), \eta, 1) \tag{15}$$

where $\eta(0.5 < \eta < 1)$ is the clipped factor. Then, the update rule of the clipped PBE action-value function is

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma_{\text{clip}}(s_{t+1}, a_{t+1}; \eta) Q(s_{t+1}, a_{t+1}). \tag{16}$$

Similarly, the clipped PBE discount reward is

$$R_t^{\pi} = \sum_{i=t}^{T} r_i \prod_{j=t}^{i} \gamma_{\text{clip}}\left(s_j, a_j; \eta\right). \tag{17}$$

According to Sutton and Barto [15], the discount factor should be in the interval $(0, 1)$. Obviously, $\pi(s, a)$ is in the interval

$$0 < \pi(s, a) < 1, \quad \sum_{a\in A} \pi(s, a) = 1. \tag{18}$$

The convergence of Q-learning can be used to prove the convergence of these clipped PBE update rules.

It can be observed from formula (15) that the clipped discount factor will force $\pi(s, a)$ to be large, thus the agent can jump out of local optima. In addition, the clipped discount factor also makes it possible to apply the PBE value update to calculate the PG of PPO.

D. Policy Update of PPO-PF

The application of the advantage function to PG algorithms needs to satisfy Lemma 1 of TRPO [20] or [45, Th. 4.1], and the sufficient condition for these theories is

$$\mu(\tilde{\pi}) = \mu(\pi) + E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t)\right] \tag{19}$$

where $\mu(\cdot)$ denotes the expected discounted reward, and $\tilde{\pi}$ and π are two different policies. Formula (19) implies that any policy update $\pi \to \tilde{\pi}$ is guaranteed to increase the expected discount reward; thus, the bound of policy optimization and trust region theorem can be established. We use $A_t^{\pi} = r_t + \gamma V_{t+1}^{\pi} - V_t^{\pi}$ as the advantage function, where V_t^{π} is the value function that estimates the PBE discount value of s_t, V_{t+1}^{π} is the estimation of PBE discount reward of s_{t+1}, and π_t is the value of $\pi(s_t, a_t)$. Let $Z_t = \prod_{j=t}^{\infty} \gamma_{\text{clip}}(s_j, a_j; \eta)$, with [20, Lemma 1], we have

$$\begin{aligned}
&E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} Z_t A_t^{\pi}\right] \\
&= E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} Z_t\left(r_t + \gamma V_{t+1}^{\pi} - V_t^{\pi}\right)\right] \\
&= E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} Z_t\left(\gamma V_{t+1}^{\pi} - V_t^{\pi}\right)\right] + E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} Z_t r_t\right] \\
&= E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} Z_t\left(\gamma V_{t+1}^{\pi} - V_t^{\pi}\right)\right] + \mu(\tilde{\pi}).
\end{aligned} \tag{20}$$

If $\gamma \to 1$, then $Z_t \le \gamma^t$. Letting $Z_t = \gamma^t$, we can get formula (19). Since $Z_t \ge \eta^t$, we set $Z_t = \eta^t$ and get

$$\begin{aligned}
&E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} \eta^t\left(\gamma V_{t+1}^{\pi} - V_t^{\pi}\right)\right] \\
&= E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} \eta^t\left(\eta V_{t+1}^{\pi} - V_t^{\pi}\right)\right] + E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} \eta^t(\gamma - \eta) V_{t+1}^{\pi}\right] \\
&= -\mu(\pi) + E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} \eta^t(\gamma - \eta) V_{t+1}^{\pi}\right]
\end{aligned}$$

$$= -\mu(\pi) + (\gamma - \eta)E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\eta^t V_{t+1}^{\pi}\right]$$
$$= -\mu(\pi) + (\gamma - \eta)\mu(\pi) \quad (21)$$
$$E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\eta^t\left(\gamma V_{t+1}^{\pi} - V_t^{\pi}\right)\right] + \mu(\pi) - (\gamma - \eta)\mu(\pi)$$
$$= \mu(\tilde{\pi}). \quad (22)$$

If we use PBE discount reward, there will generate a noisy function $\omega(\pi)$ that is in the interval with the length less than $2(\gamma - \eta)\mu(\pi)$. The entropy of the policy will be decreased along with the policy update progresses. Therefore, the probability of selecting action will increase with the policy update, and the interval of $\omega(\pi)$ will decrease to 0. Then, we can get Theorem 4, which is a sufficient condition for the monotonic improvement of policy.

Theorem 4: $\mu(\tilde{\pi}) = \mu(\pi) + E_{\tau\sim\tilde{\pi}}[\sum_{t=0}^{\infty} Z_t A^{\pi}(s_t, a_t)] + \omega(\pi)$, where $\omega(\pi)$ is a noise function converge to 0.

With Theorem 4, the advantage function $A_t^{\pi} = r_t + \gamma V_{t+1}^{\pi} - V_t^{\pi}$ can guarantee the increases of the expected discount reward.

If we use the clipped PBE discount reward instead of classic discount reward to assist policy update, the expectation of advantage function is

$$E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} Z_t A_t^{\pi'}\right] = E_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} Z_t\left(r_t + \pi_t V_{t+1}^{\pi} - V_t^{\pi}\right)\right] \quad (23)$$

where $\sum_{t=0}^{\infty} Z_{t+1} \neq \pi_{t'} \sum_{t=0}^{\infty} Z_t$, t' is a time step that is larger than t and satisfies $t' \neq t + 1$. Therefore, $\mu(\tilde{\pi})$ and $\mu(\pi)$ cannot satisfy Theorem 4, which will drastically increase the error propagation during the training of policy network. As a result, the value function in AC-PF does not converge. In addition, the policy-based clipped discount factor is used to limit the variation of $\omega(\pi)$ and ensure every policy update is effective.

E. PPO-PF

We proposed the PPO-PF algorithm by combing the AC-PF architecture with PPO. The clipped PBE discount reward R_t^{π} shown in formula (17) is used to update the state-value function, which makes the estimation of classic discount reward more accurate. The classic discount reward R_t shown in formula (1) is used to calculate the advantage function formula. Thus, the advantage function can more accurately evaluate the quality of policy, so as to reduce the variance of policy update. That means both the Actor network and Critic network of PPO-PF will update with lower variance. Thus, the stability PPO-PF can be enhanced. The loss functions of the value network and policy network are, respectively, designed as follows:

$$L_V = E\left[\left(R_t^{\pi} - V_t\right)^2\right] \quad (24)$$
$$L_{\pi} = E\left[\min(c_t(\theta)A_t, \text{clip}(c_t(\theta), 1 - \varepsilon, 1 + \varepsilon)A_t)\right]. \quad (25)$$

In summary, the pseudocode for each actor-learner thread of PPO-PF is provided in Algorithm 1.

Algorithm 1 PPO-PF Pseudocode for Each Actor-Learner Thread

```
//Assume global shared parameter vectors θ and θ_v, and global
counter I = 0
//Assume thread-specific parameter vectors θ' and θ'_v
Input: input parameters t_max, I_max, γ, η, ε, and thread
       step counter t = 1
while I ≤ I_max do
    Set gradients: dθ ← 0, dθ_v ← 0. Synchronize
    thread-specific parameter θ' and θ'_v;
    t_start = t, get state s_t;
    while t ≤ t_max or s_t = s_terminal do
        Set gradients: dθ ← 0, dθ_v ← 0. Synchronize
        thread-specific parameter θ' and θ'_v;
        Receive reward r_t and new state s_{t+1}, t ← t + 1,
        I ← I + 1;
    end
    for every time step t do
        Compute clipped PBE discount reward
        R_t^π = Σ_{i=t}^{T} r_i Π_{j=t}^{i} clip(π(s_j, a_j; θ'), η, 1);
        Compute advantage function
        A_t^π = r_t + γ V_{t+1}^π − V_t^π
        Accumulate gradients with respect to θ'
        dθ ← dθ + ∇_{θ'} log[ min(c_t(θ),
                              clip(c_t(θ), 1 − ε, 1 + ε))A_t^π ]
        Compute advantage function
        A_t^π = r_t + γ V_{t+1}^π − V_t^π;
        dθ_v ← dθ_v + ∂(R_t^π − V(s_t))^2 / ∂θ'_v
    end
    Perform asynchronous update of θ and θ_v using dθ
    and dθ_v respectively.
end
```

IV. Experiments

In this section, we used Atari games and control tasks available from the Arcade Learning Environment and OpenAI Gym as our experimental platforms to evaluate PPO-PF against PPO.

A. Atari Games

In 1976, the Atari Company introduced the first home game console: Atari 2600. In recent years, these Atari games have become the most challenging and commonly used DRL testbed. In the experiments of Atari games, both Actor and Critic networks consist of three convolution layers and a fully connected layer. The structures of three convolution layers from input to output are: 1) 32 kernels of size 8×8 and stride 4; 2) 64 kernels of size 4×4 and stride 2; and 3) 64 kernels of size 3×3 and stride 1. The fully connected layer has 512 units. The activation functions used in each layer are: relu, relu, relu, and linear, respectively.

Fig. 2 shows the reward curves on Atari games, where the shaded area is the standard deviation of the average reward and the solid line is the average reward of three independent

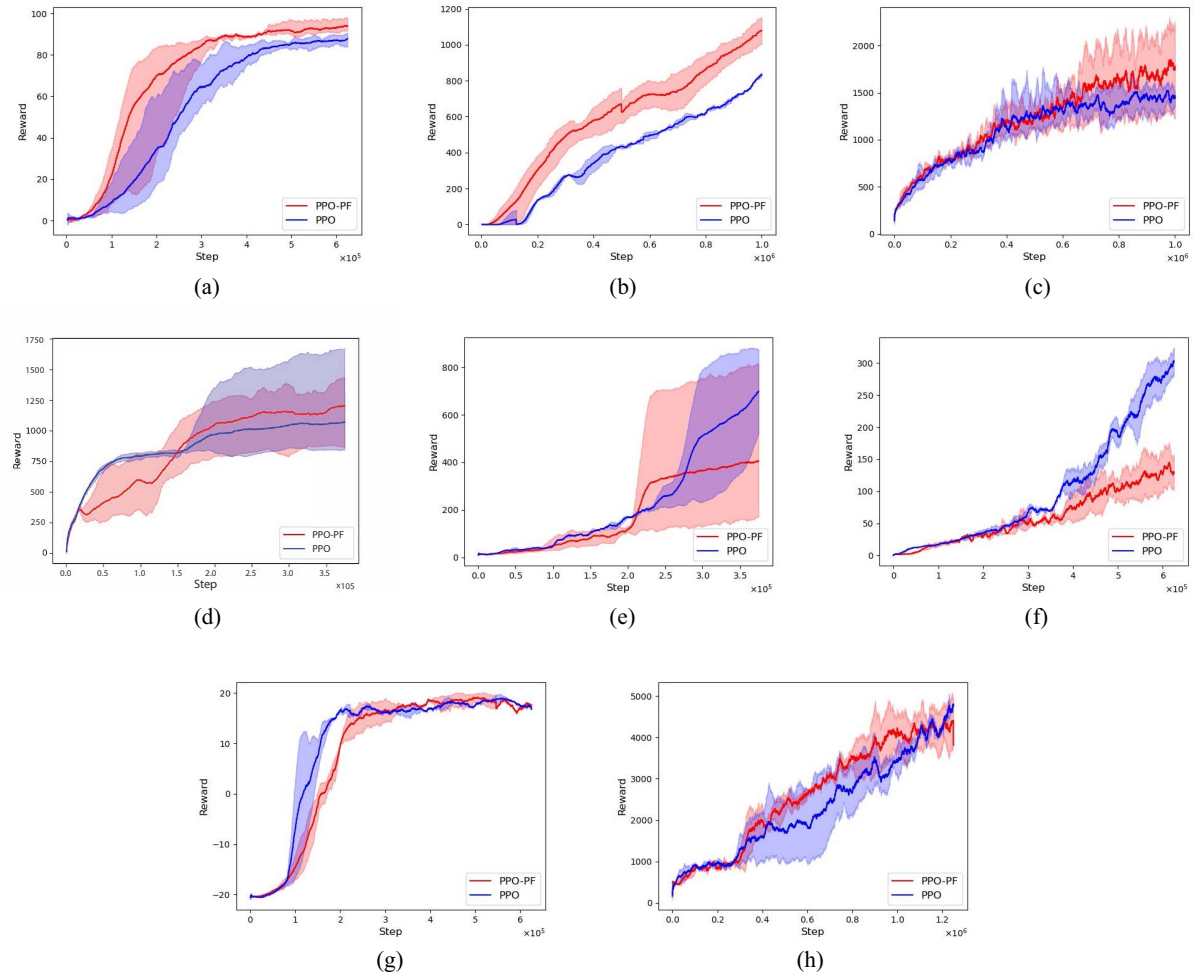


Fig. 2. Reward curves on Atari games. (a) Boxing. (b) Enduro. (c) Alien. (d) Seaquest. (e) Bankheist. (f) Breakout. (g) Pong. (h) Gopher.

TABLE I
MEAN OF AVERAGE REWARDS OVER THE FINAL 100 STEPS AND THE PERFORMANCE IMPROVEMENT OF PPO-PF OVER PPO ON ATARI GAMES

	Boxing	Enduro	Alien	Seaquest	Bankheist	Breakout	Pong	Gopher
PPO	87.15	825.297	1454.72	875.54	**675.57**	**295.44**	17.33	**4688.66**
PPO-PF	**93.86**	**1074.22**	**1779.82**	**1364.18**	405.06	134.23	**17.8**	4345.13
PPO-PF/PPO	**7.40%**	**30.20%**	**22.40%**	**55.80%**	-40%	-54.60%	**2.70%**	-7.30%

experiments. Table I shows the mean of average rewards over the final 100 steps and the improvement of PPO-PF over PPO. It can be observed from Fig. 2(a) and (b) that the average rewards received by PPO-PF are higher than those by PPO on both Boxing and Enduro games during the whole learning process. The goal of playing the Boxing game can be summarized as: getting score by hitting the opponent and avoiding the losing score by avoiding opponent's attacks. The score standards for getting and losing scores are the same. Enduro is a racing video game. In the Enduro game, the driver repeatedly avoids other racers and the rewards obtained by overtaking on different tracks are the same. It can be seen that Boxing and Enduro games have two common points: 1) the task can be repeated periodically and 2) the immediate reward is smooth. When faced with this kind of learning environment, PPO-PF performs much better than PPO. The experimental result on the Alien game shown in Fig. 2(c) indicates that PPO-PF and PPO receive roughly the same average reward before the 6e5th step, after which PPO-PF can find a better policy and receive much higher reward, while PPO is trapped in the local optima. In the Seaquest experiment shown in Fig. 2(d), the variance of the average reward received by PPO is smaller before step 1.5e6 but becomes very large after that, while the fluctuation of variance received by PPO-PF is small throughout the learning process. In addition, PPO-PF gets higher average reward than PPO. Experiments shown in Fig. 2(e) and (f) indicate that PPO-PF performs poorer than PPO on both Bankheist and Breakout games, which is due to the fact that the difference of immediate rewards between the early and late learning stages is relatively large. Take the Breakout game as an example, the farther the brick is located away from the board, the greater the immediate reward, i.e., the immediate reward presents an increasing trend as the learning progresses. But the expected discount reward weakens the change of immediate reward,

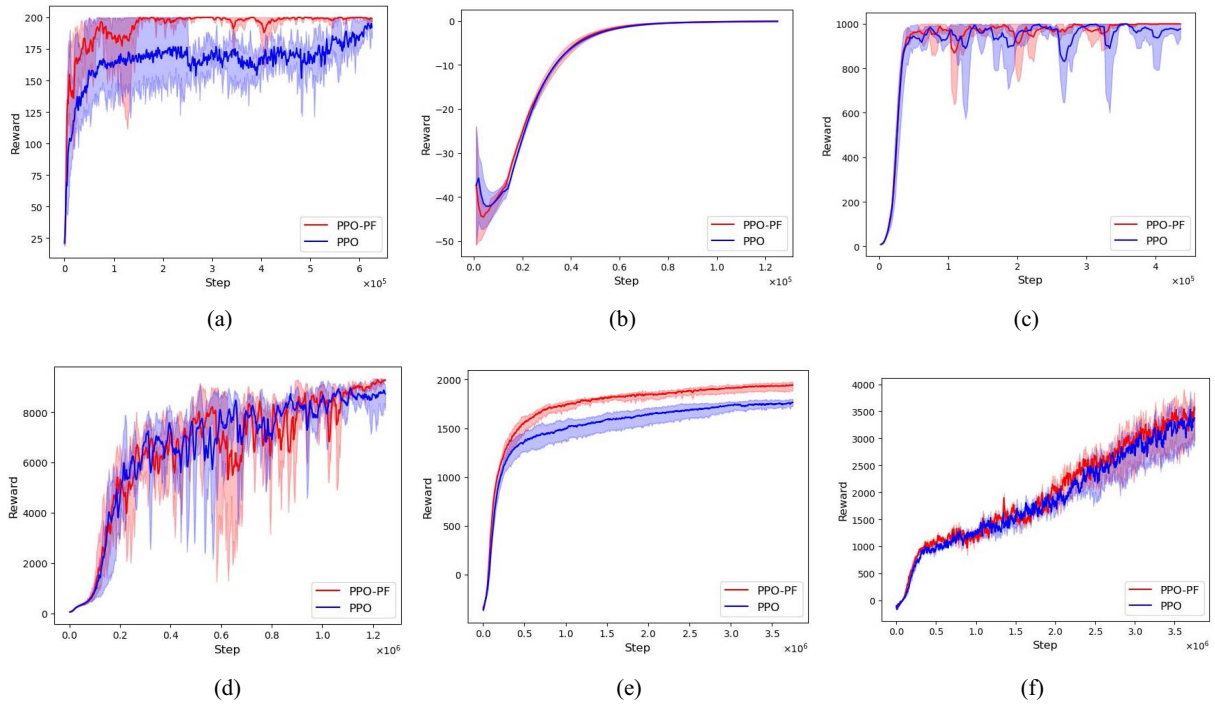


Fig. 3. Reward curves on control tasks. (a) Cartpole-v0. (b) Mountain car. (c) Inverted pendulum. (d) Inverted double pendulum. (e) Half cheetah. (f) Ant.

resulting the value network cannot response to this change and further the agent cannot find the optimal policy. It can be observed from Fig. 2(g) that at the early learning stage of Pong game, the learning speed of PPO is quicker than that of PPO-PF. But it also can be known from Table I that the final average reward received by PPO-PF is larger than that by PPO, showing the performance improvement of PPO-PF over PPO. It can be seen from Gopher experiment shown in Fig. 2(h) and Table I that PPO-PF shows a better learning effect at the early learning stage while receives smaller average reward after the 1.15e5th step than PPO, indicating PPO-PF may fall into a local optima.

B. Control Tasks

In the experiments of control tasks, both Actor and Critic networks adopt a two-layer fully connected neural network. The fully connected layers have 64 units each. We selected six control tasks shown in Table II from the OpenAI Gym platform as our controlled objects. Both the Cartpole-v0 and Mountain car are selected from the classic control toolkits of Gym, while the remaining four control tasks are from the MuJoCo platform. Cartpole-v0 is the inverted pendulum with 200 time steps in one episode, which is a simple one-dimensional (1-D) discrete-action task. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of "+1" is provided for every time step if the pole remains upright. Mountain car is a 1-D continuous-action control task, where a poorly powered car is on a 1-D track and positioned between two "mountains." The goal of Mountain car is to drive up the car to the top of mountain on the right. Only when the car arrives at the destination, the agent receives a reward of 0; otherwise, the reward is "−1." Inverted pendulum and Inverted double pendulum are three-dimensional inverted pendulum controlled objects, whose reward functions are defined as the alive bonus plus distance penalty. The Half cheetah environment is used to simulate how a biped robot controls the movement of two legs, whose goal is to run as fast as possible. It is assumed that the cheetah can run from the beginning and does not need to learn how to walk first. The Ant environment is used to simulate the crawling of a quadruped robot with 111-D states and eight brake controllers. By interacting with the environment, the quadruped robot first learns how to walk smoothly. After completing a smooth walk, the quadruped robot learns how to quickly walk or even run. In both Half cheetah and Ant environments, the reward is a function about the time step, which is proportional to the robot's velocity. It can be observed that all of these control tasks are periodically repeatable and the corresponding immediate rewards are smooth.

Fig. 3 and Table III show experiments of PPO-PF and PPO on control tasks. It can be observed that compared to PPO, PPO-PF has faster convergence speed, higher reward, and smaller variance of reward. Because of the usage of the classic discount reward, PPO is not sensitive to the actions with large selection probability. Therefore, PPO does not change the sampling frequency of these actions for every thread until the value network learned this periodicity from the sampling data of multiple threads. While the value of the policy is high, the corresponding PBE discount reward will also be large. In addition, due to the collaborative update manner of the value function and policy, PPO-PF can quickly realize that the periodicity is beneficial to learning. In summary, PPO is committed to the exploration of state–action space in an attempt to learn the optimal policy from experience, while

TABLE II
DESCRIPTION OF CONTROL TASKS

Task	Number of actions	Range of actions	Number of states	Training goal
Cartpole-v0	1	$\{0, 1\}$	4	The pole stays upright
Mountain car	1	[-3, 3]	2	The car crawled to the top of the hill
Inverted pendulum	1	[-3, 3]	4	The inverted pendulum stays upright
Inverted double pendulum	1	[-3, 3]	11	Inverted double pendulum stays upright
Half cheetah	6	[-1, 1]	17	Biped robot runs
Ant	8	[-1, 1]	111	Quadruped robot runs

TABLE III
MEAN OF AVERAGE REWARDS OVER THE FINAL 100 STEPS AND THE PERFORMANCE IMPROVEMENT OF PPO-PF OVER PPO ON CONTROL TASKS

	Cartpole-v0	Mountain car	Inverted pendulum	Inverted double pendulum	Half cheetah	Ant
PPO	182.59	-2.55	963.5	8646.96	1752.03	3273.22
PPO-PF	**199.33**	**-2.09**	**987.87**	**8847.84**	**1935.75**	**3370.79**
PPO-PF/PPO	**9.20%**	**18%**	**2.50%**	**2.30%**	**10.50%**	**2.98%**

PPO-PF tends to further improve the policy that has obtained. Therefore, PPO-PF can achieve better control effects over PPO on periodically repeatable tasks when the received immediate reward is smooth.

V. CONCLUSION

In this article, we designed a novel AC architecture with policy feedback and based on which proposed PPO-PF. The main characteristics of the proposed PPO-PF are as follows: 1) the policy is directly introduced into the update process of the value function, realizing the collaborative updates of value function and policy; 2) the PBE value function has a smaller error upper bound during the whole value function update process; 3) the policy update method based on the clipped discount factor can enhance the sensitivity of the value function to the change of policy and improve the accuracy of PBE value estimation at the early learning stage; and 4) the classic discount reward is used to define the loss functions of the policy network and value network, ensuring the policy update of PPO-PF satisfies the unbiased estimation of trust region. A collection of benchmark tasks, including games and control tasks selected from the Atari and OpenAI Gym platforms is used to demonstrate the superior performance of PPO-PF. It is worth pointing out that the designed AC-PF architecture is universal, which can be easily combined with the existing AC-based RL algorithms, such as SAC, A3C, and TRPO.

APPENDIX
PROOF OF THE THEOREMS

Theorem 1: If $a_{t+1} = a^*$, $P(\sup|Q^\pi - Q| \le \sup|Q^\pi - Q^O|) = 1$.

Proof: Suppose $a_{t+1} = a^*$, we have

$$\begin{aligned}&\sup\left|Q^\pi - Q^O\right| - \sup|Q^\pi - Q| \\ &\quad = \gamma\left|\pi\left(s_{t+1}, a^*\right)Q\left(s_{t+1}, a^*\right) - Q\left(s_{t+1}, a^*\right)\right| \ge 0. \quad (26)\end{aligned}$$

Thus, Theorem 1 is proved. ■

Theorem 2: If $\pi(s_{t+1}, a^*) \le 0.5$, $P(\sup|Q^\pi - Q| \le \sup|Q^\pi - Q^O|) = 1$.

Proof: Suppose $a_{t+1} \neq a^*$, we have

$$\begin{aligned}&\sup\left|Q^\pi - Q^O\right| - \sup|Q^\pi - Q| \\ &\quad = \gamma\left|\left[1 - \pi\left(s_{t+1}, a^*\right)\right]Q\left(s_{t+1}, a^*\right)\right| \\ &\qquad - \gamma\left|\pi\left(s_{t+1}, a^*\right)Q\left(s_{t+1}, a^*\right)\right| \\ &\qquad + \gamma \sum_{a\in A, a\neq a^*} |\pi(s_{t+1}, a)Q(s_{t+1}, a)| \\ &\qquad - \gamma \sum_{a\in A, a\neq a^*, a\neq a_{t+1}} |\pi(s_{t+1}, a)Q(s_{t+1}, a)| \\ &\quad = \gamma\left|\left[1 - \pi\left(s_{t+1}, a^*\right)\right]Q\left(s_{t+1}, a^*\right)\right| \\ &\qquad - \gamma\left|\pi\left(s_{t+1}, a^*\right)Q\left(s_{t+1}, a^*\right)\right| \\ &\qquad + \gamma|\pi(s_{t+1}, a_{t+1})Q(s_{t+1}, a_{t+1})|. \quad (27)\end{aligned}$$

If $\pi(s_{t+1}, a^*) \le 0.5$, we have

$$\begin{aligned}&\gamma\left|\left[1 - \pi\left(s_{t+1}, a^*\right)\right]Q\left(s_{t+1}, a^*\right)\right| \\ &\qquad - \gamma\left|\pi\left(s_{t+1}, a^*\right)Q\left(s_{t+1}, a^*\right)\right| \ge 0 \quad (28)\end{aligned}$$

$$\sup\left|Q^\pi - Q^O\right| - \sup|Q^\pi - Q| \ge 0. \quad (29)$$

With Theorem 1, we can get

$$P\left(\sup|Q^\pi - Q| \le \sup\left|Q^\pi - Q^O\right|\right) = 1. \quad (30)$$

Thus, Theorem 2 is proved. ■

Theorem 3: If $\pi(s_{t+1}, a^*) > 0.5$, $P(\sup|Q^\pi - Q| \le \sup|Q^\pi - Q^O|) > \pi(s_{t+1}, a^*)$.

Proof: If $a_{t+1} = a^*$, we have

$$\begin{aligned}&P\left(\sup\left|Q^\pi - Q^O\right| \ge \sup|Q^\pi - Q|, a_{t+1} = a^*\right) \\ &\quad = \pi\left(s_{t+1}, a^*\right) \quad (31)\end{aligned}$$

where $P(a_{t+1} = a^*) = \pi(s_{t+1}, a^*)$.

If $a_{t+1} \neq a^*$, we assume that $\sup|Q^\pi - Q| > \sup|Q^\pi - Q^O|$ always exist, which means the following true proposition is established:

$$P\left(\sup\left|Q^\pi - Q^O\right| \ge \sup|Q^\pi - Q|, a_{t+1} \neq a^*\right) = 0. \quad (32)$$

According to formula (27), we have

$$\begin{aligned}&\gamma\left(1 - 2\pi\left(s_{t+1}, a^*\right)\right)\left|Q\left(s_{t+1}, a^*\right)\right| \\ &\qquad + \gamma\pi(s_{t+1}, a_{t+1})|Q(s_{t+1}, a_{t+1})| < 0. \quad (33)\end{aligned}$$

It is easily observed from formula (32) that we can adjust $Q(s_{t+1}, a_{t+1})$ arbitrarily. With $|Q(s_{t+1}, a_{t+1})| = |Q(s_{t+1}, a^*)|$, the following inequality is established:

$$\pi(s_{t+1}, a_{t+1}) < 2\pi\left(s_{t+1}, a^*\right) - 1. \tag{34}$$

However, for example, if the dimension of action space is 2, the solution of formula (34) will be $\pi(s_{t+1}, a^*) > (2/3)$. Obviously, $\pi(s_{t+1}, a^*) > (2/3)$ and the precondition $\pi(s_{t+1}, a^*) > 0.5$ are contradictory. Therefore, the inverse proposition of formula (32) holds, i.e.,

$$P\left(\sup\left|Q^\pi - Q^O\right| \geq \sup\left|Q^\pi - Q\right|, a_{t+1} \neq a^*\right) > 0. \tag{35}$$

According to formulas (31) and (35), we can get

$$P\left(\sup\left|Q^\pi - Q\right| \leq \sup\left|Q^\pi - Q^O\right|\right) > \pi\left(s_{t+1}, a^*\right). \tag{36}$$

Thus, Theorem 3 is proved. ■

REFERENCES

[1] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," 2013. [Online]. Available: arXiv:1312.5602.

[2] H. Hu, S. Song, and G. Huang, "Self-attention-based temporary curiosity in reinforcement learning exploration," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Dec. 18, 2019, doi: 10.1109/TSMC.2019.2957051.

[3] J. Yu, Z. Wu, X. Yang, Y. Yang, and P. Zhang, "Underwater target tracking control of an untethered robotic fish with a camera stabilizer," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Jan. 13, 2020, doi: 10.1109/TSMC.2019.2963246.

[4] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy, "Deep reinforcement learning for sequence-to-sequence models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2469–2489, Jul. 2020.

[5] X. You, X. Li, Y. Xu, H. Feng, and J. Zhao, "Toward packet routing with fully distributed multiagent deep reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Aug. 13, 2020, doi: 10.1109/TSMC.2020.3012832.

[6] S. Wang, D. Jia, and X. Weng, "Deep reinforcement learning for autonomous driving," 2018. [Online]. Available: arXiv: 1811.11329.

[7] X. Xu, L. Zuo, X. Li, L. L. Qian, J .K. Ren, and Z. P. Sun, "A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3884–3897, Oct. 2020.

[8] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[9] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015. [Online]. Available: arXiv: 1511.05952.

[10] V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double *Q*-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[11] X. S. Wang, Y. Gu, Y. H. Cheng, A. P. Liu, and C. L. Philip Chen, "Approximate policy-based accelerated deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 1820–1830, Jun. 2020.

[12] M. Fortunato *et al.*, "Noisy networks for exploration," 2017. [Online]. Available: arXiv:1706.10295.

[13] H. Ong, K. Chavez, and A. Hong, "Distributed deep *Q*-Learning," 2015. [Online]. Available: arXiv: 1508.04186.

[14] Z. Wang, N. Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," 2015. [Online]. Available: arXiv: 1511.06581.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[16] J. Peters, and S. Schaal, "Policy gradient methods for robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 2219–2225.

[17] C. Sammut and G. I. Webb, *Encyclopedia of Machine Learning*. Berlin, Germany: Springer-Verlarg, 2010.

[18] D. Silver, G. Lever, N. Heess, T. Degris, D, Wierstra, and M. Riedmiler, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 605–619.

[19] L. Wang, Q. Cai, Z. Yang, and Z. Wang, "Neural policy gradient methods: Global optimality and rates of convergence," 2019. [Online]. Available: arXiv: 1909.01150.

[20] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: arXiv:1707.06347.

[22] K. Ciosek and S. Whiteson, "Expected policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2868–2875.

[23] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in *Proc. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8092–8101.

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

[25] D. Wang, M. M. Ha, and J. F. Qiao, "Self-learning optimal regulation for discrete-time nonlinear systems under event-driven formulation," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1272–1279, Mar. 2020.

[26] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: arXiv:1509.02971.

[27] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Trust-PCL: An off-policy trust region method for continuous control," 2017. [Online]. Available: arXiv: 1707.01891.

[28] O. Nachum, Y. Chow, and M. Ghavanizadeh, "Path consistency learning in tsallis entropy regularized MDPs," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1578–1594.

[29] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1352–1361.

[30] J. Schulman, X. Chen, and P. Abbeel, "Equivalence between policy gradients and soft *Q*-learning," 2017. [Online]. Available: arXiv: 1704.06440.

[31] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2017, pp. 2776–2786.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor–critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018. [Online]. Available: arXiv: 1801.01290.

[33] H. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement learning through asynchronous advantage actor–critic on a GPU," 2016. [Online]. Available: arXiv: 1611.06256.

[34] A. Ilyas *et al.*, "Are deep policy gradient algorithms truly policy gradient algorithms?" 2018. [Online]. Available: arXiv:1811.02553.

[35] A. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artif. Intell.*, vol. 172, pp. 716–737, Apr. 2008.

[36] A. Tenorio-Gonzalez, E. Morales, and L. Villasenor-Pineda, "Dynamic reward shaping: Training a robot by voice," in *Proc. Adv. Artif. Intell.*, 2010, pp. 483–492.

[37] P. Pilarski, M. Dawson, T. Degris, F. Fahimi, J. Carey, and R. Sutton, "Online human training of a myoelectric prosthesis controller via actor–critic reinforcement learning," in *Proc. IEEE Int. Conf. Rehabil. Robot.*, 2011, pp. 1–7.

[38] X. Liu, Q. Zhou, H. Ren, and C. Sun, "Reinforcement learning for robot navigation in nondeterministic environments," in *Proc. 5th IEEE Int. Conf. Cloud Comput. Intell. Syst.*, 2019, pp. 615–619.

[39] W. Knox and P. Stone, "Reinforcement learning from human reward: Discounting in episodic tasks," in *Proc. IEEE Int. Symp. Robot Human Interact. Commun.*, 2012, pp. 878–885.

[40] V. Francois-Lavet, R. Fonteneau, and D. Ernst, "How to discount deep reinforcement learning: Towards new dynamic strategies," 2015. [Online]. Available: arXiv: 1512.02011.

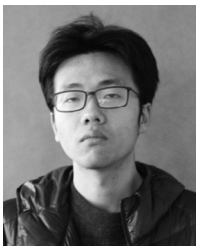
[41] N. Yoshida, E. Uchibe, and K. Doya, "Reinforcement learning with state-dependent discount factor," in *Proc. IEEE 3rd Joint Int. Conf. Develop. Learn. Epigen. Robot.*, 2013, pp. 1–6.

[42] E. Greensmith, P. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," in *Proc. Annu. Neural Inf. Process. Syst. Conf.*, 2004, pp. 1–60.

[43] H. Seijen, H. Hasselt, D. Whiteson, and M. Wiering, "A theoretical and empirical analysis of expected sarsa," in *Proc. IEEE Symp. Adapt. Dyn. Program. Reinforcement Learn.*, 2009, pp. 177–184.

[44] K. Ciosek, and S. Whiteson, "Expected policy gradients for reinforcement learning," 2018. [Online]. Available: arXiv: 1801.03326.

[45] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 267–274.



Yang Gu received the B.S. degree in electrical engineering and automation from the China University of Mining and Technology, Xuzhou, China, in 2016, where he is currently pursuing the Ph.D. degree with the School of Information and Control Engineering.

His main research interest is reinforcement learning.

Yuhu Cheng (Member, IEEE) received the Ph.D. degree in control science and engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2005.

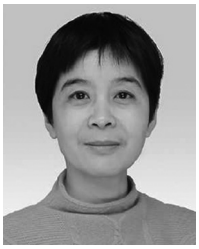
He is currently a Professor with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China. His main research interests include machine learning, transfer learning, and intelligent system.

Prof. Cheng was a recipient of the New Century Excellent Talents in University from the Ministry of Education of China in 2010.

C. L. Philip Chen (Fellow, IEEE) received the M.S. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 1985, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1988.

He is currently the Dean of the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China, and a Chair Professor of the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau, China. His research area is in computational intelligence, systems, and cybernetics.

Dr. Chen was a President of IEEE Systems, Man, and Cybernetics Society from 2012 to 2013. He has been the Editor-in-Chief of IEEE TRANSACTIONS ON CYBERNETICS since 2020 and an Associate Editor of several IEEE TRANSACTIONS. He is also the Chair of TC 9.1 Economic and Business Systems of International Federation of Automatic Control. He is an Accreditation Board of Engineering and Technology Education Program Evaluator for Computer Engineering, Electrical Engineering, and Software Engineering Programs. He is a Fellow of AAAS, the Chinese Association of Automation, and HKIE.



Xuesong Wang (Member, IEEE) received the Ph.D. degree in control science and engineering from the China University of Mining and Technology, Xuzhou, China, in 2002.

She is currently a Professor with the School of Information and Control Engineering, China University of Mining and Technology. Her main research interests include machine learning, bioinformatics, and artificial intelligence.

Prof. Wang was a recipient of the New Century Excellent Talents in University from the Ministry of Education of China in 2008. She is currently an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, and *International Journal of Machine Learning and Cybernetics*.