Safe Adaptive Policy Transfer Reinforcement Learning for Distributed Multiagent Control

Du, Bin; Xie, Wei; Li, Yang; Yang, Qisong; Zhang, Weidong; Negenborn, Rudy R.; Pang, Yusong; Chen, Hongtian

DOI
10.1109/TNNLS.2023.3326867

Publication date
2025

Document Version
Final published version

Published in
IEEE Transactions on Neural Networks and Learning Systems

Safe Adaptive Policy Transfer Reinforcement Learning for Distributed Multiagent Control

Bin Du, *Member, IEEE*, Wei Xie, *Member, IEEE*, Yang Li, Qisong Yang, Weidong Zhang, *Senior Member, IEEE*, Rudy R. Negenborn, Yusong Pang, and Hongtian Chen, *Member, IEEE*

***Abstract*— Multiagent reinforcement learning (RL) training is usually difficult and time-consuming due to mutual interference among agents. Safety concerns make an already difficult training process even harder. This study proposes a safe adaptive policy transfer RL approach for multiagent cooperative control. Specifically, a pioneer and follower off-policy policy transfer learning (PFOPT) method is presented to help follower agents acquire knowledge and experience from a single well-trained pioneer agent. Notably, the designed approach can transfer both the policy representation and sample experience provided by the pioneer policy in the off-policy learning. More importantly, the proposed method can adaptively adjust the learning weight of prior experience and exploration according to the Wasserstein distance between the policy probability distributions of the pioneer and the follower. Case studies show that the distributed agents trained by the proposed method can complete a collaborative task and acquire the maximum rewards while minimizing the violation of constraints. Moreover, the proposed method can also achieve satisfactory performance in terms of learning speed and success rate.**

***Index Terms*— Adaptive weight update, distributed multiagent system, policy transfer, safe reinforcement learning (RL), transfer learning.**

I. Introduction

In the past few years, reinforcement learning (RL) has shown its effectiveness [1], [2], [3]. Similar in principle to data-driven adaptive optimal control methods, RL enables agents to maximize their cumulative rewards by interacting with the real or virtual environment. Owing to the inherent characteristics of data-driven techniques, the majority of model-free deep RL methods necessitate a significant amount of time and computational resources for agent training. Training difficulty tends to increase geometrically as the number of agents increases. Transfer learning helps multiagent systems to improve training efficiency, thanks to a better baseline. Specifically, the multiagent training efficiency can be fully improved if it exploits the pretrained agent policy neural network. In this process, agents learn individual skills first and then learn how to work together as a team. This brief aims to study how to utilize the transfer learning method to make the multiagent RL training process more efficient.

Transfer learning has achieved remarkable success in computer vision, natural language processing, and RL control. In transfer learning, a well-trained policy (similar to the control law) can be treated as an expert policy that can be deployed on an untrained agent [4]. Policy transfer [5] is derived from policy distillation (PD) [6], which is one of the most common methods of transfer learning. Policy transfer can be used to train a new policy and apply it to a similar mission. The expert policy can be created by pretraining or deriving from demonstration data of human [7]. The new policy is always learned by regularizing the divergence of action distributions between the expert policy and the new exploration. This technique makes the training progress effortless and enormously improves data efficiency. Policy transfer can also be used to adjust the exploration and training strategy for multiagent systems [8]. To improve the data efficiency, an off-policy sample repainted algorithm was proposed in [9] by using samples collected from the expert policy. The repainted policy is updated with the previous off-policy collected samples. In [10], a multiagent lateral transfer method was proposed to ease training in multiagent RL. It utilized features for knowledge transfer and incorporated pretrained policy networks to boost training efficiency. In [11], a transductive learning algorithm using the cellular learning automata was introduced to address the negative transfers issue. The algorithm employs two learning automata estimators and integrates customized decision criteria, namely merit and attitude parameters, to constrain negative transfers within the cellular learning automata.

The safety concerns are crucial for multiagent systems that are always associated with state constraints [12], [13], [14]. To solve state constraint problems, constrained RL is employed in the safety-critical learning [15], [16]. During this process, the safety-cost signal is employed in the constrained RL. Constrained RL problems are generally converted into unconstrained ones by adding a Lagrangian multiplier [17]. This multiplier balances both the reward and the safety-cost signal, simultaneously. Accordingly, a PID Lagrange multiplier method was proposed in [18], which utilizes derivatives of the constraint function where the traditional Lagrange multiplier is substituted with the control coefficient of integral, proportional, and derivative items. The Lagrangian approach can also solve the max–min optimization problem, which employs penalty coefficients to impose restrictions [19]. One common focus of previous studies is to learn a safe policy by incorporating constraints into the policy training [17], [20], [21]. Agents learn policies that satisfy state constraints before deploying them to applications. The literature [22] addressed the challenge of safely transferring existing policies to new tasks in RL, especially in situations where safety is paramount. Policy optimization algorithms for constrained RL have been developed

Manuscript received 18 December 2022; revised 8 July 2023 and 6 September 2023; accepted 18 October 2023. Date of publication 2 November 2023; date of current version 8 January 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2022ZD0119903, in part by the National Natural Science Foundation of China under Grant U2141234, in part by the Shanghai Science and Technology Program under Grant 19510745200, in part by the China Scholarship Council under Grant 202106230194, and in part by the SURF Cooperative through the Dutch National e-Infrastructure under Grant EINF-2851. *(Corresponding authors: Weidong Zhang; Wei Xie.)*

Bin Du is with the Ocean Institute, Northwestern Polytechnical University, Taicang 215400, China, and also with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: bin.du123@outlook.com).

Wei Xie, Weidong Zhang, and Hongtian Chen are with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: weixie@sjtu.edu.cn; wdzhang@sjtu.edu.cn; hongtian.chen@sjtu.edu.cn).

Yang Li is with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China (e-mail: lyxc56@gmail.com).

Qisong Yang is with the Xi'an Institute of High-Tech, Xi'an 710071, China (e-mail: q.yang@tutanota.com).

Rudy R. Negenborn and Yusong Pang are with the Department of Maritime and Transport Technology, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: r.r.negenborn@tudelft.nl; Y.Pang@tudelft.nl).

Digital Object Identifier 10.1109/TNNLS.2023.3326867

in [23], [24], and [25], which maximizes the cumulative reward under safety constraints.

Although the above research has made significant efforts on policy transfer and constrained RL, the following problems have not been fully studied.

1) Training multiple agents to cooperate with each other is much more difficult than training a single agent, which often requires more computing resources and training time. Constraints also limit the training efficiency of multiagent RL. A promising topic is how to utilize the prior experience of a single agent to help multiple agents complete tasks when computing power is limited.
2) Ongoing exploration and cooperation will be inhibited if agents follow the pioneer experience blindly. A study of how to balance the tradeoff between pioneer experience and ongoing exploration is relatively crucial during different stages of training.

Therefore, this brief proposes a safe adaptive policy transfer RL approach for distributed multiagent control. To accelerate the training process in multiagent RL, the pioneer and follower off-policy policy transfer learning (PFOPT) method has been developed. The proposed PFOPT expands the teacher–student architecture to encompass multiagent scenarios. It can leverage both on-policy and off-policy methods to update the critic value network, utilizing samples collected from pioneer policies. Besides, the learning weights of PFOPT are adaptively adjusted based on the Wasserstein distance. By employing this adaptive weight update strategy, PFOPT effectively balances pioneer experience with ongoing exploration. The main contributions of this study are summarized as follows.

1) A distributed off-policy actor–critic architecture is proposed for multiagent cooperation control, and safety constraints are taken into consideration. It offers significant advantages by allowing agents to build upon existing knowledge and experiences, enabling them to learn more efficiently and complete training faster.
2) A PFOPT method has been developed. It not only transfers a pretrained pioneer policy representation from the on-policy learning to the off-policy learning, but also utilizes a generalized advantage function to transfer useful knowledge deriving from the pioneer policy. This transfer process allows for the efficient utilization of previously acquired knowledge, reducing the need for extensive retraining and accelerating the learning process.
3) An adaptive weight update strategy is proposed for policy updating, which effectively balances the integration of pioneer experience and ongoing exploration. This equilibrium ensures that follower agents can leverage the valuable knowledge and experiences of the pioneer agent while retaining the flexibility to explore.

The rest of the brief is organized as follows. Section II introduces preliminaries of multiagent RL, policy transfer, and Wasserstein distance. The PFOPT method is proposed in Section III. In Section IV, a case study is shown to verify PFOPT. Finally, the conclusion is drawn in Section V.

II. Preliminaries

In this section, the background of multiagent RL, policy transfer, and Wasserstein distance are introduced.

A. Multiagent RL

We frame multiagent RL as a constrained Markov game $(\boldsymbol{N}, \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R}, \boldsymbol{P}, \boldsymbol{C})$, where $\boldsymbol{N}$ is the number of agents, $\boldsymbol{S}$ is the state space, $\boldsymbol{A}$ is the action space of each agent, $\boldsymbol{R} : \boldsymbol{S} \times \boldsymbol{A}^n \times \boldsymbol{S} \to \mathbb{R}$ is the collective reward, $\boldsymbol{P}$ is the transition function that maps the current state s_t and the joint action a_t to a distribution over the next state s_{t+1}, and $\boldsymbol{C}_j : \boldsymbol{S} \times \boldsymbol{A}^n \times \boldsymbol{S} \to \mathbb{R}$ is the space of auxiliary cost functions. At each time step t, each agent's target policy π_i, $i \in 1, \ldots, n$, selects an action $a_{i,t} \in \boldsymbol{A}$. A Markov decision process involves state s, joint action $a = (a_1, \ldots, a_n)$, transition probability function $p(s' \mid s, \boldsymbol{a})$, and reward function $r_i(s_i, a_i)$. At each time step, each agent executes an action according to its policy π_i that maps the state to an action probability. The action value function is expressed as $Q_i(s_i, a_i) = \mathbb{E}_{a_i \sim \pi(\cdot|s_i)}[\sum_{t=0}^{\infty} \gamma^t r(s_{i,t}, a_{i,t}) \mid s_{i,0} = s, a_{i,0} = a]$. The distributional safety critic is introduced to obtain a safety action. By using policy π, the expectation of long-term cumulative costs is defined as $Q_{C,i}(s_i, a_i) = \mathbb{E}_{a_i \sim \pi(\cdot|s_i)}[\sum_{t=0}^{\infty} \gamma^t c(s_{i,t}, a_{i,t}) \mid s_{i,0} = s, a_{i,0} = a]$ and the cost state value function is defined as $V_{C,i}(s) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t c(s_{i,t}, a_{i,t}) \mid s_{i,0} = s]$. The Bellman function is given as $\mathcal{T}^{\pi} Q_{C,i}(s_i, a_i) = c(s_i, a_i) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V_{C,i}(s_{t+1})]$, where $\mathcal{T}^{\pi}$ is a distributional Bellman operator [26], [27], $s_{t+1} \sim p(\cdot \mid s_t, a_t)$, and $a_{t+1} \sim \pi(\cdot \mid s^{t+1})$. The critic and actor functions in deep RL are parameterized by neural networks, which are updated using a policy gradient approach.

B. Policy Transfer

Referring to the survey in [28], the basic concept of transfer learning is introduced in this section. $\mathcal{D}_s$ is defined as the source domain that contains prior knowledge κ_s, and $\mathcal{D}_t$ is the target domain that has target knowledge κ_t. Knowledge can be transferred between $\mathcal{D}_s$ and $\mathcal{D}_t$. The objective of transfer learning is to generate an optimal policy π^* for the agent, by leveraging expert knowledge κ_s from $\mathcal{D}_s$ and target knowledge κ_t from $\mathcal{D}_t$ such that

$$\pi^* = \arg\max_{\pi \in \hat{\Pi}} \mathbb{E}_{s \sim d, a \sim \pi}\left[Q^{\pi}_{\mathcal{D}}(s, a)\right] \tag{1}$$

where $\pi \in \hat{\Pi}(\kappa_s \sim \mathcal{D}_s, \kappa_t \sim \mathcal{D}_t) : \boldsymbol{S} \to \boldsymbol{A}$ is a function mapping from the states to actions for the target domain $\mathcal{D}_t$, and $Q^{\pi}_{\mathcal{D}}(s, a)$ is an action-value function. The agent finally learns information from both κ_t and κ_s. By using the information from κ_s, the agent learns better in the target domain $\mathcal{D}_t$ than the ones without utilizing κ_s.

C. Wasserstein Distance

The Wasserstein distance is essentially a distance metric between two probability distributions. From a physical perspective, the Wasserstein distance can be illustrated as the minimum energy cost of transforming and moving blocks from the shape of one probability distribution to that of another distribution. It has two advantages: 1) it is a symmetric distance, whereas Kullback–Leibler (KL) and Jensen-Shannon (JS) divergence only measures similarity between two distributions based on the information content or loss and 2) the Wasserstein distance takes into account the basic geometry of the space that defines the distribution. In practice, there is a distribution shift between two distributions, and the Wasserstein distance is an appropriate measure to leverage the deviation between the two. In this brief, Wasserstein distance is employed to measure the deviation between two different probability distributions. The standard Wasserstein distance is described as

$$W_p(P, Q) = \left(\inf_{J \in \mathcal{J}(P,Q)} \int \|x - y\|^p dJ(x, y)\right)^{1/p} \tag{2}$$

where J is a joint probability and $\mathcal{J}(P, Q)$ is a coupling distribution that combines a set of individual probability distributions P and Q. In the first dimension, it can be expressed as

$$W_p(P, Q) = \left(\int_0^1 \left|F^{-1}(z) - G^{-1}(z)\right|^p\right)^{1/p} \tag{3}$$

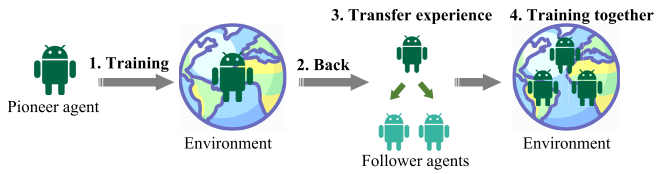

Fig. 1. Schematic of the pioneer and follower policy transfer.

where F^{-1} and G^{-1} are the quantile functions. In the case of $p = 1$, a change of variables leads to the formula that

$$W_p(P, Q) = \int_{\mathbb{R}} |F(x) - G(x)| dx. \tag{4}$$

In this brief, the learning weights of PFOPT are updated with Wasserstein distance.

III. Pioneer and Follower Off-Policy Policy Transfer Method

In this section, PFOPT is proposed to accelerate the RL training process and assist in training agents to work cooperatively. It transfers a pretrained pioneer policy from on-policy learning to off-policy learning and utilizes a generalized advantage function to transfer useful knowledge deriving from the pioneer policy. An adaptive weight update strategy is presented to balance the tradeoff between the pioneer experience and ongoing exploration.

A. Pioneer and Follower Transfer Learning

Fig. 1 illustrates the concept of the pioneer and follower policy transfer. Initially, a few part of the agents are first trained to explore the task environment. The selection of the pioneer agent is based on its ability to achieve high rewards and a high success rate, ensuring the reliability of its policies for subsequent transfer learning steps. As a result, this pioneer agent contributes to positive transfer effects for the follower agents. Then, the source knowledge learned by the pioneer is transferred to the follower agents and trained together. For one agent, PFOPT utilizes four neural networks including a policy network π_θ, three value networks for reward $Q_i^\phi(s, a)$, cost $Q_{C,i}^\varphi(s, a)$, and policy transfer $\hat{Q}_i^\psi(s, a)$. θ, ϕ, φ, and ψ represent the networks parameters. The policy network generates the mean and standard deviation of the conditional Gaussian probability, and then the agent takes a continuous action a in the state s according to the policy. The value networks take observation s and action a as inputs and return the value $Q_i^\phi(s, a)$, $Q_{C,i}^\varphi(s, a)$, and $\hat{Q}_i^\psi(s, a)$. To improve the stability of the optimization, a network with the same structure is set as the target network. The agent periodically updates the target network parameters to that of the latest corresponding value network.

To improve data efficiency, off-policy learning is introduced in PFOPT. A replay buffer $\hat{\mathcal{D}}$ is formed by collecting training samples generated by the pioneer policy π_p, and then the policy transfer value function $\hat{Q}_i^\psi(s_{i,t}, a_{i,t})$ is computed by the neural network with parameter ψ from samples $\hat{\mathcal{D}}$. The pioneer-related advantage function $\hat{A}_{i,t}(s_{i,t}, a_{i,t})$ is calculated via

$$\begin{aligned}\hat{A}_{i,t}(s_{i,t}, a_{i,t}) &= (1-\eta)\left[\delta_{i,t}^Q\left(\frac{1}{1-\eta}\right) + \gamma\delta_{i,t+1}^Q\left(\frac{\eta}{1-\eta}\right)\right. \\ &\quad \left. + \gamma^2\delta_{i,t+2}^Q\left(\frac{\eta^2}{1-\eta}\right) + \cdots + \gamma^{T-t}\delta_{i,T}^Q\left(\frac{\eta^{T-t}}{1-\eta}\right)\right] \\ &= \sum_{l=0}^{T-t}\delta_{i,t+l}^Q(\gamma\eta)^l \end{aligned} \tag{5}$$

where T is the number of steps and $\eta \in [0, 1]$ is the advantage estimate parameter. The variable $\delta_{i,t}^Q$ is defined as $\delta_{i,t}^Q = r_{i,t} + \gamma\hat{Q}_i^\psi(s_{i,t}, a_{i,t}) - \hat{Q}_i^\psi(s_{i,t}, a_{i,t})$. The samples are obtained from different distributions and employed to update the critic state value network. As a part of the global loss function, the pioneer loss function is

$$\begin{aligned}\hat{L} = &\mathop{\mathbb{E}}_{a_i\sim\pi_{\theta_i}(\cdot|s_i)}\left[\frac{\pi_{\theta_i}(a_i \mid s_i)}{\pi_p(a_i \mid s_i)} \cdot \hat{A}_{i,t}(s_{i,t}, a_{i,t})\right] \\ &+ \mathop{\mathbb{E}}_{a_i\sim\pi_{\theta_i}(\cdot|s_i)}\left[\exp\left(\mathrm{KL}[\pi_{\theta_i}(\cdot \mid s_i), \pi_p(\cdot \mid s_i)]\right)\right] \end{aligned} \tag{6}$$

where π_{θ_i} represents a policy of follower agent i and π_p stands for the policy of the pioneer agent. An agent policy π_{θ_i} (follower policy) acquires experience and knowledge from a pioneer agent. Apart from the pioneer loss function, the global loss function contains the critic loss L_Q, policy improvement loss L_π, and cost loss L_C. Both L_Q and L_π are employed to update the Q function. The policy evaluation approach updates the policy π by reducing regularized TD error. A TD target value function y_i is defined as follows:

$$\begin{aligned}y_i(r_{i,t}, s_{i,t+1}) = r_{i,t} + \gamma \mathop{\mathbb{E}}_{a_{i,t+1}\sim\pi_{\theta_i}}\left[Q_i^\phi(s_{i,t+1}, a_{i,t+1})\right. \\ \left. - \alpha_{\pi,i}\log\pi_{\theta_i}(a_{i,t+1} \mid s_{i,t+1})\right] \end{aligned} \tag{7}$$

where γ is the discount factor, and $\alpha_{\pi,i}$ is the entropy loss weight. $\alpha_{\pi,i}\log\pi_{\theta_i}(a_{i,t+1} \mid s_{i,t+1})$ is the weighted policy entropy for the bounded output of the actor when in state s_i. The entropy term is set in (7) to enrich the exploration. The optimal policy is natural to seek a policy with some entropy. It has a positive effect on enlarging the exploration space. Initially, the policy network selects an unbounded action randomly from a Gaussian distribution. PFOPT calculates the entropy of a given policy based on the unbounded probability distribution during training. By applying the $\mathrm{Tan}h(x)$ function and scaling operations to the unbounded action, the policy network generates bounded actions for the agent. When $s_{i,t}$ indicates a terminal state, the target value $y_{i,t}$ equals the reward value $r_{i,t}$. If not, the target value function equals the sum of $r_{i,t}$, the minimum discounted future reward, and the weighted entropy. The parameters of the critic are updated through the following equation:

$$L_Q(\phi_i) = \frac{1}{M}\sum_{i=1}^{M}\mathop{\mathbb{E}}_{\tau_i\sim\mathcal{D}}\left[y_i(r_t, s_{t+1}) - Q_i^\phi(s_{i,t}, a_{i,t})\right]^2 \tag{8}$$

where M is the number of the mini-batch, $\tau_i = (s_t, a_t, r_t, c_t, s_{t+1}, \ldots)_i$ is the trajectory of state and actions from agent i, and $\mathcal{D}$ is the replay buffer. The objective of policy improvement, which should be minimized, is expressed as follows:

$$\begin{aligned}L_\pi = &\frac{1}{M}\sum_{i=1}^{M}\mathop{\mathbb{E}}_{\substack{s_{i,t}\sim p\\ a_{i,t}\sim\pi_{\theta_i}}}\left[\alpha_{\pi,i}\log\pi_{\theta_i}(a_{i,t} \mid s_{i,t})\right] \\ &- \frac{1}{M}\sum_{i=1}^{M}\mathop{\mathbb{E}}_{\substack{s_{i,t}\sim p\\ a_{i,t}\sim\pi_{\theta_i}}}\left[Q_i^\phi(s_{i,t}, a_{i,t})\right]. \end{aligned} \tag{9}$$

Deriving from the soft-actor–critic algorithm [29], the entropy weight is also updated by minimizing the following equation:

$$L(\alpha_{\pi,i}) = \frac{1}{M}\sum_{i=1}^{M}\left(-\alpha_{\pi,i}\log\pi_{\theta_i}(a_{i,t} \mid s_{i,t}) - \alpha_{\pi,i}\mathcal{H}\right) \tag{10}$$

where $\mathcal{H}$ is the target entropy. The cost state value function can be expressed as $V_{C,i}(s_{i,t+1}) = \sum_{a_t\sim\pi_\theta(\cdot|s_t)}\pi(a_{i,t} \mid s_{i,t})Q_{C,i}^\varphi(s_{i,t}, a_{i,t})$. The cost advantage function can be obtained as $A_C(s_i, a_i) = Q_{C,i}^\varphi(s_i, a_i) - V_{C,i}(s_i)$. The loss function of cost [30] can be written as

$$L_C = \mathop{\mathbb{E}}_{\substack{s_i\sim p\\ a_i\sim\pi}}\left[A_C(s_i, a_i)\right]. \tag{11}$$

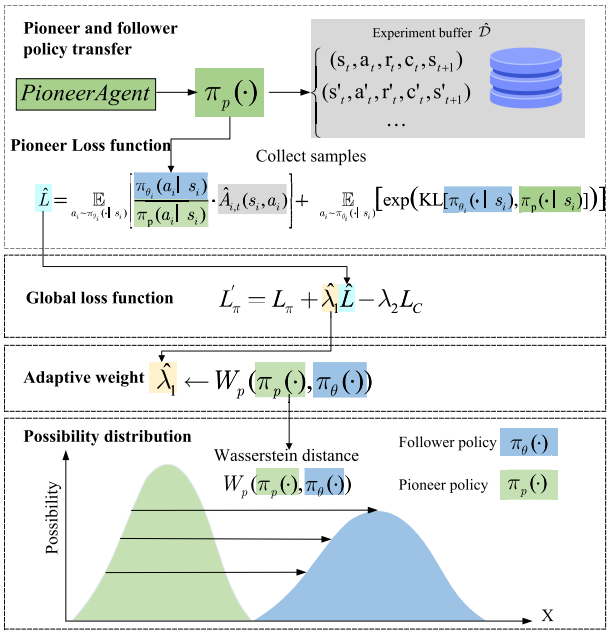

Fig. 2. Illustration of policy updates in PFOPT.

Eventually, the actor parameters are updated by minimizing the following global loss function:

$$L'_\pi = L_\pi + \hat{\lambda}_1 \hat{L} - \lambda_2 L_C \tag{12}$$

where $\hat{\lambda}_1$ represents a weight that leverages the pioneer policy and agent exploration which is automatically adjusted by the Wasserstein distance between the two policies. The Wasserstein distance is different from the KL divergence. The KL divergence only describes the similarity of two distributions, and the Wasserstein distance can measure the positional relationship between two distributions. λ_2 maintains a balance between exploration and compliance with constraints, ensuring that the agent explores the state space while avoiding actions that violate safety constraints. The local loss functions, L_π, $\hat{L}$, and L_C, are derived from (9), (6), and (11), respectively.

B. Adaptive Weight Update Strategy Based on Wasserstein Distance

The Wasserstein distance between the pioneer policy $\pi_p(\cdot)$ and agent policy $\pi_{\theta_i}(\cdot)$ is defined as

$$W_p(\pi_p(\cdot), \pi_{\theta_i}(\cdot)) = \int_{\mathbb{R}} \left| F_{\pi_p(\cdot)} - F_{\pi_{\theta_i}(\cdot)} \right| dx \tag{13}$$

where F is the cumulative distribution function. The weight $\hat{\lambda}_1$ in (12) is updated by

$$\hat{\lambda}_1 = 2/\pi \lambda_1 \arctan\left(W_p(\pi_p, \pi_{\theta_i})^2\right) \tag{14}$$

where λ_1 is a scale constant, and its effect will be considered in the case study. The weight of $\hat{\lambda}_1$ rises if Wasserstein distance increases so that follower policy can be closer to pioneer policy at the initial stage. As training progresses, the policy of follower agents gets close to the pioneer policy. The knowledge from the pioneer agent becomes less valuable, thus $\hat{\lambda}_1$ is decreased to improve the exploration. After conducting multiple simulations, when λ_1 is set to 0.001, the estimates of $\hat{\lambda}_1$ are distributed between 0.0002 and 0.0005. Therefore, PFOPT can efficiently balance the tradeoff using the previous pioneer experience and exploration. Fig. 2 shows the policy update process, and the pseudocode of the proposed method is shown in Algorithm 1.

Algorithm 1 PFOPT Algorithm

Initialize: Agent policy network π_{θ_i}; Pioneer policy π_p; Value network of reward $Q_i^\phi(s,a)$; Value network of policy transfer $\hat{Q}_i^\psi(s,a)$; Value network of costs $Q_{C,i}^\varphi(s,a)$; Discount rates γ; Generalized advantage estimate parameter η; Learning rates $\alpha_{1,i}$, $\alpha_{2,i}$, $\alpha_{3,i}$ and $\alpha_{\pi,i}$; Weights λ_1, λ_2.

for each episode **do**

  for each agent i **do**

    Collect samples $\mathcal{D} = \{(s, a, s', r)\}$ using $\pi_{\theta_i}(\cdot)$;

    Collect samples $\hat{\mathcal{D}} = \{(\hat{s}, \hat{a}, \hat{s}', \hat{r})\}$ using $\pi_p(\cdot)$;

    Compute advantage estimates A_C from $\mathcal{D}$ and $\hat{A}_{i,t}$ from $\hat{\mathcal{D}}$;

    Get local loss functions: L_π in (9), $\hat{L}$ in (6), and L_C in (11);

    Compute global actor loss function:

$$L'_\pi = L_\pi + \hat{\lambda}_1 \hat{L} - \lambda_2 L_C.$$

    Update value networks: $\phi_i \leftarrow \phi_i - \alpha_{1,i} \nabla_\phi L_Q(\phi_i)$, $\varphi_i \leftarrow \varphi_i - \alpha_{2,i} \nabla_\varphi L_C$, $\psi_i \leftarrow \psi_i - \alpha_{3,i} \nabla_\psi \hat{A}(\psi_i)$;

    Update actor policy network $\theta_i \leftarrow \theta_i - \alpha_{\pi,i} \nabla_\theta L'_\pi$;

    Compute the Wasserstein distance $W_p(\pi_p, \pi_\theta)$;

    Update the weight $\hat{\lambda}_1$ via (13) and update temperature parameter $\alpha_{\pi,i} \leftarrow \alpha_{\pi,i} - \nabla_{\alpha_{\pi,i}} L(\alpha_{\pi,i})$.

  end for

end for

IV. Case Study

In this section, a simple simulation environment is developed to validate PFOPT in the constraint multiagent RL control. The simulation environment is called a "push-out game," in which agents are controlled to push the target circle out of the ring. In this study, a single-agent policy is pretrained as a pioneer policy and then it is employed to train follower agents using PFOPT. Additionally, the state-of-the-art methods are employed for comparison with PFOPT. Moreover, to explain the effect of core parameters on PFOPT, λ_1 and λ_2 parameters are selected for sensitivity analysis.

A. Simulation Settings

As shown in Fig. 3, a simulation environment is designed, in which agents cooperate to achieve a mission. The homogeneous agents are trained to push a target object out of a ring together in the 2-D surface. Agents highlighted in red, green, magenta, and yellow are represented by smaller circles with a 1-m radius, respectively. Agents utilize collision forces to move a target blue circle outside the black ring. The radius of the target blue circle and black ring are 2 and 8 m, respectively. The agents start randomly in -10–10 rectangular areas. The target starts in the (0, 0) position. The mass of the target and agent are 2 and 1 kg, respectively. Additionally, the agents are encouraged to satisfy the safety concern: the target circle should not be pushed out of red borders. The boundary line is two vertical lines with $x = \pm 6$. All environmental components have mass and follow Newton's rules. Interaction forces between components and environmental constraints are considered as springs and mass dampers. A neural network is trained as a policy generator to output the force in RL control. The hyperparameters are listed in Table I. In each simulation, ten random seeds and 1000 training episodes are set in the simulation. The agents' departure locations are random. The reward function is defined as $R = \sum_{i=1}^{N}(r_i)/N$, where $r_i = r_{\text{global}} + r_{\text{local},i}$, $r_{\text{global}} = 0.001\, d_{\text{target}}$, and $r_{\text{local},i} = -0.005\, d_{i,\text{target}} - 0.008\, u_i^2$. d_{target} represents the distance between the center of the ring and the target. Reward r_i is determined by r_{global} and $r_{\text{local},i}$. r_{global} is a team reward received by all of the agents as the target moves toward the ring's boundary. $d_{i,\text{target}}$ is the distances

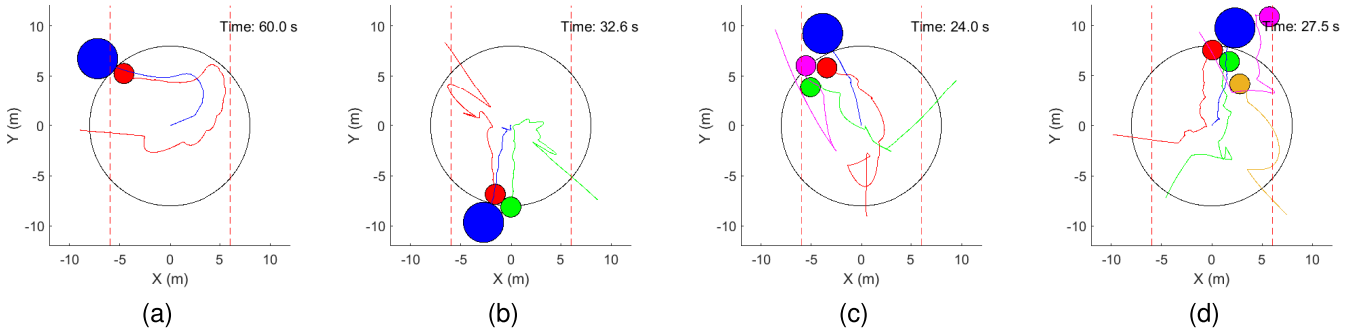

Fig. 3. Trajectories of a different number of agents. (a)–(d) One, two, three, and four agents, respectively.

TABLE I
HYPERPARAMETERS IN ALGORITHMS

Hyper-parameter	MACPO	MAPPO-L	MATRPO-L	PFOPT
No. of hidden layers	3	3	3	3
No. of hidden nodes	512	512	512	512
Activation	relu	relu	relu	relu
Initial log std	-0.5	-0.5	-1	-0.5
Discount for reward, cost	0.99	0.99	0.99	0.99
Batch size	1024	1024	1024	1024
Minibatch size	N/A	64	N/A	64
Maximum episode	1e3	1e3	1e3	1e3
Learning rate for policy	N/A	1e-4	N/A	1e-4
Learning rate for reward, cost value network	1e-4	1e-4	1e-4	1e-4
L2-regularization for value net	1e-5	1e-5	1e-5	1e-5
coefficient λ_1	N/A	N/A	N/A	0.001
coefficient λ_2	N/A	N/A	N/A	0.1

between the agent i and the target. The agents apply external forces that result in motion. u_i is the control variable of the agent i. The range of u_i is set to be from range -1 to 1. The cost function is also defined as $c(x) = 4 - \|x_{\text{target}}\|$, where x_{target} is the x coordinate of the target.

B. Performance of PFOPT in Different Number of Agents

Fig. 3 shows the performance of agents with well-trained policy. Training a single agent as the pioneer to perform the task alone is an essential step before training a multiagent system in PFOPT. Accordingly, the policy of a single agent is only used as the cornerstone, even if its performance is unsatisfactory. Following single-agent experience, multiagents are trained based on PFOPT. In Fig. 3, a well-trained agent pushes the blue target out of the ring after multiple touches and bounces. Generally, efficiency can be improved by adding agents, however, too many agents can be counterproductive due to mutual influence. It is generally accepted that the more agents there are, the more difficult it is to train them to cooperate. Table II shows the performance decreases with the number of agents trained by PFOPT. The well-trained agents are tested with 1000 randomly generated tasks. In Table II, the definition of "constraint violation rate" is the ratio of operations that violate constraints. Increasing the number of agents leads to a decrease in both task success rate and training speed, while constraint violations and task time also increase.

TABLE II
PERFORMANCE OF DIFFERENT NUMBERS OF AGENTS TRAINED BY PFOPT

Index	2 Agents	3 Agents	4 Agents
Success rate	96.1%	86.2%	61.4%
Average task time (s)	22.43	40.46	39.59
Constraint violation rate	27.6%	40.8%	45.2%

TABLE III
PERFORMANCE COMPARISON OF AGENTS TRAINED BY TRPO-L, CPO, PPO-L, AND PFOPT

Index	MACPO	MAPPO-L	MATRPO-L	PFOPT
Success rate	54.3%	42.7%	26.4%	96.1%
Average task time (s)	57.56	58.84	59.73	22.43
Constraint violation rate	24.1%	21.3%	8.4%	27.6%

C. Comparison of PFOPT and PD During Training

To demonstrate that PFOPT significantly improves the training speed, a comparison has been made between the proposed method and a state-of-the-art transfer learning method known as PD [5]. In Fig. 4, the blue line represents the reward/cost curve of PFOPT, and the yellow line stands for the reward/cost curve of PD. It illustrates that PFOPT can significantly accelerate the convergence of the reward curve in comparison with PD. Furthermore, the terminal reward of PFOPT is higher than that of PD. There is a difference in average costs between PFOPT and PD during training, but the terminal costs are almost identical. Fig. 4(a) and (e) shows the single-agent training process whose policy is used as a pioneer policy in transfer learning. Fig. 4(b)–(d) and (f)–(h) displays the multiagent training process. It can also be observed from the case study results that PFOPT shows better performance compared to PD.

D. Performance Comparison of PFOPT With Other Multiagent RL Methods

To evaluate the performance of PFOPT, it is compared with the state-of-the-art methods, such as multiagent policy transfer framework (MAPTF) [31] and multiagent deep deterministic policy gradient (MADDPG) [32]. Due to the absence of safety constraints in the aforementioned methods, we add the cost evaluation in their loss functions. Besides, we also extend the classic single safe RL methods including proximal policy optimization (CPO) [17], proximal policy optimization-Lagrangian (PPO-L) [33], and trust region policy optimization-Lagrangian (TRPO-L) [34] to the multiagent RL. These methods are recalled as MACPO, MAPPO-L,

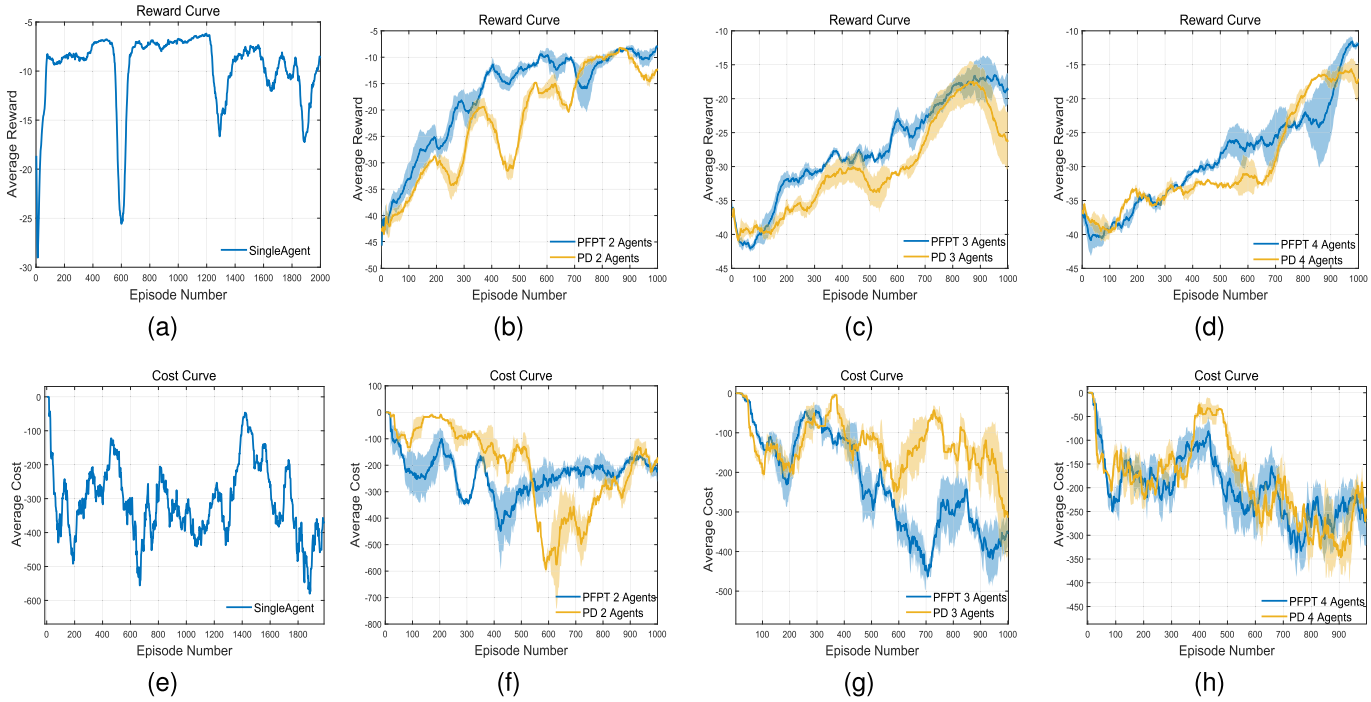

Fig. 4. Performance of PFOPT with different numbers of agents compared to PD. (a)–(d) Reward curves. (e)–(h) Display cost curves.

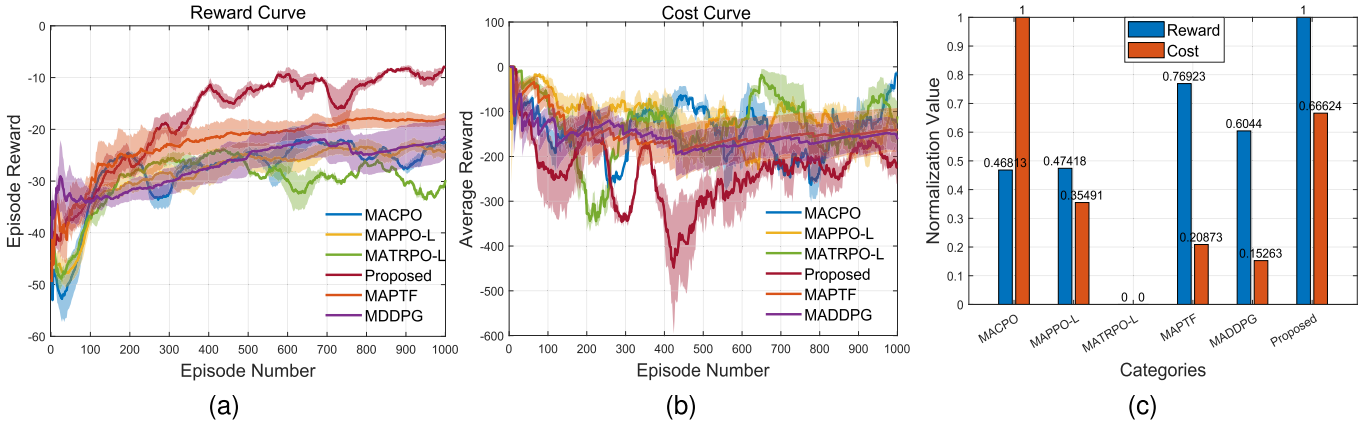

Fig. 5. Comparison of results for MACPO, MAPPO-L, MATRPO-L, MAPTF, MADDPG, and proposed PFOPT via (a) reward curves, (b) cost curves, and (c) normalized average rewards and costs.

and MATRPO-L, respectively. The hyperparameters of MACPO, MAPPO-L, MATRPO-L, and PFOPT are listed in Table I.

Fig. 5 illustrates the training process of two agents using the above four methods. Fig. 5(a) demonstrates that PFOPT not only largely reduces the training time, but also improves the terminal reward compared to MACPO, MAPPO-L, MATRPO-L, MAPTF, and MADDPG. The PFOPT demonstrates exceptional performance among all the investigated approaches, signifying its proficiency in efficiently transferring valuable information to follower agents. This effectiveness can be attributed to PFOPT's adaptive mechanism, which empowers each agent to selectively assimilate relational knowledge from pioneer agents. However, PFOPT shows no substantial difference in constraint costs from MACPO, MAPPO-L, MATRPO-L, MAPTF, or MADDPG in the simulation [see Fig. 5(b)]. Fig. 5(c) indicates that PFOPT has the highest average reward, and its cost is only lower than that of the MACPO. As shown in Table III, PFOPT-trained agents have the highest success rates and the shortest average time to complete tasks.

E. Hyperparameters Sensitivity Analysis

In this section, the sensitivities of two essential hyperparameters λ_1 and λ_2 are investigated. PFOPT is tested with five different values of λ_1 and λ_2. λ_1 directly affects weight $\hat{\lambda}_1$ which is used to adjust the progress of transfer learning. Increasing λ_1 will result in a more pioneer-like behavior, and vice versa. In Fig. 6(a) and (b), the performance of PFOPT is highly sensitive to λ_1. When λ_1 is less than 0.05, convergence of the reward curves is slowing down. Instead of imitating the pioneer agent, the follower agent can formulate its own policy. The hyperparameter λ_2 can be interpreted as a factor that adjusts the strength of the safety constraint. As a result of its increase, rewards will be reduced and costs will fluctuate. Simulation results in Fig. 6(c) and (d) show that PFOPT is not sensitive to the value of λ_2. The rewards and costs show insignificant changes when the value of λ_2 is varied between 0.05 and 0.3. Both the rewards and costs exhibit significant fluctuations as the value of λ_2 increases from 1 to 5.

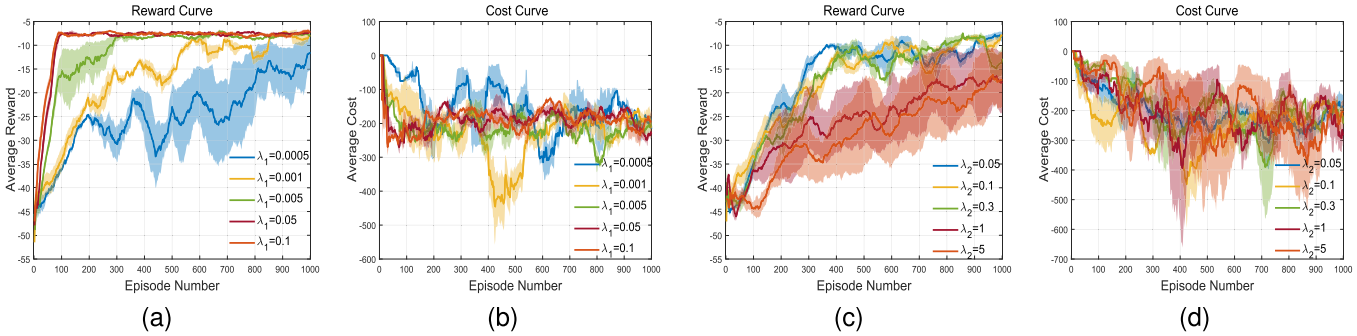

Fig. 6. Performance under different choices of λ_1 and λ_2 in the proposed PFOPT method. (a) λ_1 reward. (b) λ_1 cost. (c) λ_2 reward. (d) λ_2 cost.

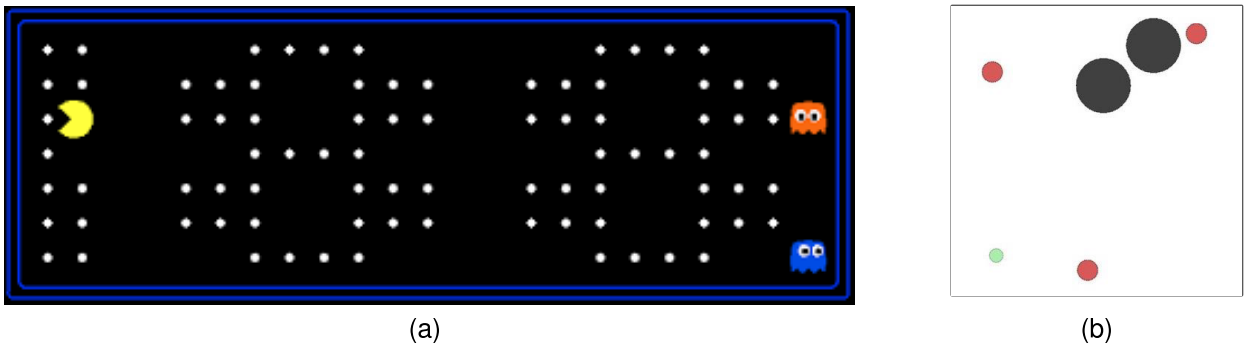

Fig. 7. Supplement simulation scenarios include (a) pac-man and (b) multiparticle environment.

F. Supplement Additional Simulation Scenarios

Pac-man and multiagent particle environment games are chosen as testbeds in this section. Pac-Man is a discrete cooperative-competitive game. This brief focuses on the OpenClassic scenario [see Fig. 7(a)], which entails two ghost players and one pac-man player. The objective of the pac-man player entails consuming as many pills as feasible while evading the relentless pursuit of the ghost players. Conversely, the ghost players are determined to capture the pac-man player. The aim of the algorithm is to govern the actions of the ghost players while considering the pac-man player as the adversary. The termination of the game occurs when a ghost captures the pac-man player or when the episode surpasses 100 steps. Multiagent particle environment [32] is a simulation environment that encompasses a particle world featuring continuous observation and action spaces. Within the environment [Fig. 7(b)], the simple tag scenario is chosen in this brief. The scenario consists of two types of entities: good agents (green) and adversaries (red). Good agents are faster and receive a negative reward (-10) for being hit by adversaries. Adversaries are slower and receive a positive reward ($+10$) for hitting good agents. Additionally, there are obstacles (large black circles) that obstruct the entities' movement. By default, there is one good agent, six adversaries, and two obstacles.

Fig. 8(a) depicts the average rewards attained in the OpenClassic scenario. It is evident that the performance of PFOPT surpasses that of MAPTF and attains an average discount reward of approximately 0.17, exhibiting a certain level of variance. Conversely, MAPTF achieves an average discount reward of approximately -0.1. It suggests that PFOPT effectively facilitates knowledge transfer among ghost players, resulting in enhanced overall performance. Fig. 8(b) illustrates the average rewards curve within the multiagent particle environment. In this simulation, the final reward is determined as the cumulative sum of individual agents' rewards. It is evident that PFOPT outperforms MAPTF in terms of average rewards. This superior performance can be attributed to PFOPT's ability to effectively strike a balance between leveraging the pioneer agent's valuable experience and engaging in new explorations. Additionally, PFOPT employs the Wasserstein distance metric to dynamically adapt

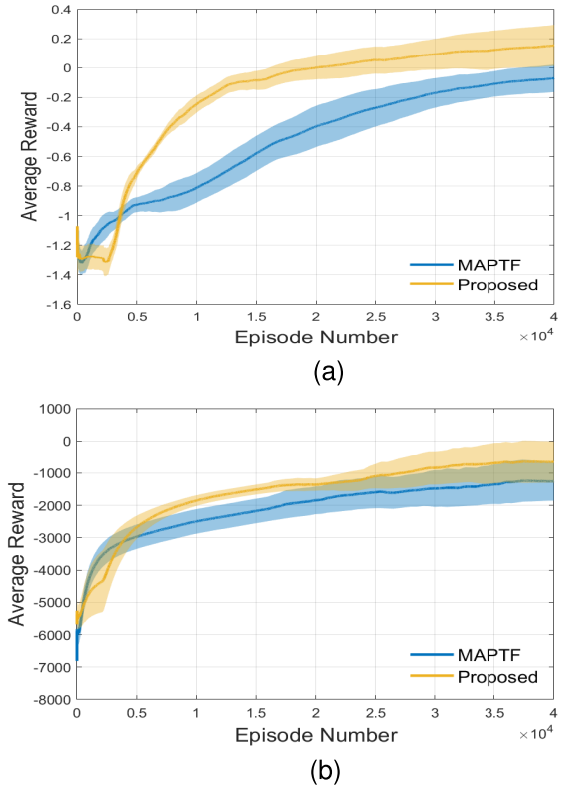

Fig. 8. Reward curves between MAPTF and proposed method in pac-man and multiparticle environment. (a) Pac-man. (b) Multiagent particle environment.

the transfer learning process, thereby fostering a more efficient and effective dissemination of knowledge among the pioneer agent and the follower agents.

V. Conclusion

In this brief, a multiagent RL method has been proposed to solve a distributed multiagent cooperative control problem. Specifically, a distributed off-policy actor–critic architecture has been proposed for training agents to collaboratively perform tasks. To accelerate the training process, a pioneer and follower off-policy transfer learning method has been developed, in which a well-trained policy is adaptively transferred from the pioneer agent to the follower agent. There are two main advantages of the proposed PFOPT method. First, unlike traditional RL methods wherein all agents are trained simultaneously, in the PFOPT method, a pioneer agent is utilized to execute tasks independently, accumulating foundational experience. Subsequently, the learned policy and knowledge are transferred to other agents, enabling the follower agents to swiftly assimilate the experience of their predecessors. Second, to balance the learning weights between pioneer experience and new exploration during training, an adaptive weight update strategy based on the Wasserstein distance between the pioneer policy and the agent's policy has been proposed. The effectiveness of the proposed PFOPT method has been verified by case studies. In the future, the constraint transfer framework will be investigated to quantify the safety requirements in safety-critical RL systems.

References

[1] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[2] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[3] W. Bai, T. Li, Y. Long, and C. L. P. Chen, "Event-triggered multigradient recursive reinforcement learning tracking control for multiagent systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 366–379, Jan. 2023.

[4] J. Ramon, K. Driessens, and T. Croonenborghs, "Transfer learning in reinforcement learning problems through partial policy recycling," in *Proc. 18th Eur. Conf. Mach. Learn. (ECML)*. Berlin, Germany: Springer, 2007, pp. 699–707.

[5] A. A. Rusu et al., "Policy distillation," 2015, *arXiv:1511.06295*.

[6] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[7] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7391–7403, Oct. 2023.

[8] H. Hu, G. Huang, X. Li, and S. Song, "Meta-reinforcement learning with dynamic adaptiveness distillation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 3, pp. 1454–1464, Mar. 2023.

[9] Y. Tao, S. Genc, J. Chung, T. Sun, and S. Mallya, "REPAINT: Knowledge transfer in deep reinforcement learning," in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 10141–10152.

[10] H. Shi, J. Li, J. Mao, and K.-S. Hwang, "Lateral transfer learning for multiagent reinforcement learning," *IEEE Trans. Cybern.*, vol. 53, no. 3, pp. 1699–1711, Mar. 2023.

[11] S. A. H. Minoofam, A. Bastanfard, and M. R. Keyvanpour, "TRCLA: A transfer learning approach to reduce negative transfer for cellular learning automata," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2480–2489, May 2023.

[12] N. T. Kokolakis and K. G. Vamvoudakis, "Safety-aware pursuit-evasion games in unknown environments using Gaussian processes and finite-time convergent reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 10, 2022, doi: 10.1109/TNNLS.2022.3203977.

[13] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han, and Y. Zhao, "Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5435–5444, Dec. 2021.

[14] Y. Yang, K. G. Vamvoudakis, H. Modares, Y. Yin, and D. C. Wunsch, "Safe intermittent reinforcement learning with static and dynamic event generators," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5441–5455, Dec. 2020.

[15] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained Markov decision processes," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 9797–9806.

[16] X. Yang and Q. Wei, "Adaptive critic learning for constrained optimal event-triggered control with discounted cost," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 91–104, Jan. 2021.

[17] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 22–31.

[18] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by PID Lagrangian methods," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 9133–9143.

[19] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. New York, NY, USA: Academic Press, 2014.

[20] Z. Liu et al., "Constrained variational policy optimization for safe reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 13644–13668.

[21] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. J. Spaan, "Safety-constrained reinforcement learning with a distributional safety critic," *Mach. Learn.*, vol. 112, pp. 859–887, Jun. 2022.

[22] Z. Feng, B. Zhang, J. Bi, and H. Soh, "Safety-constrained policy transfer with successor features," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2023, pp. 7219–7225.

[23] S. Lu, K. Zhang, T. Chen, T. Basar, and L. Horesh, "Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 10, 2021, pp. 8767–8775.

[24] S. Huang et al., "A constrained multi-objective reinforcement learning framework," in *Proc. 5th Conf. Robot Learn.*, 2022, pp. 883–893.

[25] T. Zhang, S. Guo, T. Tan, X. Hu, and F. Chen, "Adjacency constraint for efficient hierarchical reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4152–4166, Apr. 2023.

[26] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, Jul. 1966.

[27] Z. Song, R. E. Parr, and L. Carin, "Revisiting the softmax Bellman operator: New benefits and new perspective," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 5916–5925.

[28] H. Chen, H. Luo, B. Huang, B. Jiang, and O. Kaynak, "Transfer learning-motivated intelligent fault diagnosis designs: A survey, insights, and perspectives," *TechRxiv*, pp. 1–18, Oct. 2022.

[29] T. Haarnoja et al., "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

[30] Z. Peng, Q. Li, C. Liu, and B. Zhou, "Safe driving via expert guided policy optimization," in *Proc. 5th Conf. Robot Learn.*, 2022, pp. 1554–1563.

[31] T. Yang et al., "An efficient transfer learning framework for multiagent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 17037–17048.

[32] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.

[33] A. Ray, J. Achiam, and D. Amodei, "Benchmarking safe exploration in deep reinforcement learning," 2019, *arXiv:1910.01708*.

[34] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, "Lyapunov-based safe policy optimization for continuous control," 2019, *arXiv:1901.10031*.