

Personalized Federated Deep Reinforcement Learning-based Trajectory Optimization for Multi-UAV Assisted Edge Computing

Zhengrong Song^{*}, Chuan Ma[†], Ming Ding[‡], Howard H. Yang[§], Yuwen Qian^{*}, Xiangwei Zhou[¶]

^{*}Nanjing University of Science and Technology, Nanjing, China

[†]Zhejiang Lab, Hangzhou, China [§]ZJU-UIUC Institute, Zhejiang University, Haining 314400, China

[‡]Data61, CSIRO, Sydney, Australia [¶]Louisiana State University, Baton Rouge, LA 70803, USA

Email: zhengrongsong@njjust.edu.cn, chuan.ma@zhejianglab.edu.cn

Abstract—In the era of 5G mobile communication, there has been a significant surge in research focused on unmanned aerial vehicles (UAVs) and mobile edge computing technology. UAVs can serve as intelligent servers in edge computing environments, optimizing their flight trajectories to maximize communication system throughput. Deep reinforcement learning (DRL)-based trajectory optimization algorithms may suffer from poor training performance due to intricate terrain features and inadequate training data. To overcome this limitation, some studies have proposed leveraging federated learning (FL) to mitigate the data isolation problem and expedite convergence. Nevertheless, the efficacy of global FL models can be negatively impacted by the high heterogeneity of local data, which could potentially impede the training process and even compromise the performance of local agents. This work proposes a novel solution to address these challenges, namely personalized federated deep reinforcement learning (PF-DRL), for multi-UAV trajectory optimization. PF-DRL aims to develop individualized models for each agent to address the data scarcity issue and mitigate the negative impact of data heterogeneity. Simulation results demonstrate that the proposed algorithm achieves superior training performance with faster convergence rates, and improves service quality compared to other DRL-based approaches.

Index Terms—Personalized federated deep reinforcement learning (PF-DRL), multi-UAV, mobile edge computing, trajectory optimization.

I. INTRODUCTION

The speedy advancement of contemporary communication technology has brought about an intensified demand for computing processing from mobile users. However, conventional cloud computing models fall short of catering to the requirements of massive data processing. In recent times, mobile edge computing has considerably enhanced the efficacy of task processing, providing mobile users with high-quality network services and minimal latency [1]. Nevertheless, certain remote regions present considerable challenges in the deployment

of edge servers. In light of their portability, flexibility, and high mobility, UAV-assisted edge computing has emerged as a burgeoning trend in such scenarios [2].

Trajectory optimization is a vital component in UAV edge computing [3]. As an action-learn-based algorithm, reinforcement learning (RL) shows its potential advantages in addressing the complex environment [4]. In [5], an improved Q-learning-based algorithm was proposed for path planning in an unknown antagonistic environment. Authors in [6] designed a Monte Carlo Tree Search (MCTS)-based path planning scheme, which can plan the flight path of a UAV in a dynamic environment reasonably. However, the aforementioned algorithms can only be applied to discrete and low-dimensional action spaces [7]. In reality, each state of the agent contains numerous action possibilities, and previous work [4-6] may suffer from the curse of high-dimensional action, leading to a slow or even non-convergence rate.

In order to solve the above problems, combining the deep neural network with RL, a deep reinforcement learning (DRL) algorithm is utilized to solve the dimensional disaster of huge states and action spaces [8], [9]. Moreover, a single UAV provides limited services, while multi-UAV can expand the service scope and provide better service quality through mutual cooperation. Thus, authors in [10] proposed a multi-agent deep deterministic policy gradient (MADDPG) based trajectory optimization algorithm to realize the fairness of user service and the server terminal load, as well as minimizing the UAV's energy consumption. In [11], a collaborative multi-agent DRL framework was proposed to obtain the joint strategy of trajectory design, task allocation, and power management. In [12], a potential game method was proposed to solve the service allocation problem of multi-UAV in advance and then optimize the trajectory.

Nevertheless, multi-agent deep reinforcement learning faces challenges like low learning efficiency and slow convergence rates in complex dynamic environments. These issues arise due to the fact that agents interact and learn from each other, and changes for different clients can affect each other's decisions, leading to instability. As a new learning paradigm, federated learning (FL) has become more and more popular

This work is partially supported by the Youth Foundation Project of Zhejiang Lab (No. K2023PD0AA01), partially supported by the Research Initiation Project of Zhejiang Lab (No. 2022PD0AC02), partially supported by the National Natural Science Foundation of China under Grant No. 62002170, partially supported by Zhejiang Lab Open Research Project (No. K2022PD0AB05). Corresponding author: Chuan Ma.

in recent years. Through FL, distributed learning schemes can be efficiently performed among multiple participants at a low communication cost [13], and the combination of FL and RL has become a natural solution to address the multi-UAV trajectory optimization. Recently, the authors in [14] proposed a federated multi-agent deep deterministic policy gradient (F-MADDPG), in which model parameters are shared, thus greatly reducing communication delay and overhead. Although this algorithm has a good performance in MEC system allocation scheduling problems, it is not practical that different clients share a single model, especially when clients locate in heterogeneous situations [15].

To address the aforementioned issues, in this paper, we propose a UAV trajectory optimization algorithm based on personalized federated deep reinforcement learning (PF-DRL). In detail, each UAV no longer uses a single global model. By improving the aggregation process of FL, the global model and the local model are aggregated with a moderate weight to train a personalized model. Such an approach not only amplifies the learning efficiency but also empowers each UAV to make personalized action decisions guided by the local training model, thus ensuring the overall learning performance. Specifically, the main contributions of this paper are summarized as follows.

- We build a multi-UAV-assisted mobile edge computing model for the complex and dynamic environment. While optimizing the flight trajectory of each UAV, the overall energy consumption is reduced as required.
- Since each UAV is trained locally based on the MADDPG algorithm. We, for the first time, propose a UAV trajectory optimization method based on personalized federated multi-agent deep deterministic policy gradient (PF-MADDPG).
- Finally, the proposed method is synthetically simulated. The results show that the learning efficiency and convergence rate of our algorithm is significantly improved without degrading the overall performance.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, the system model is presented. As shown in Fig. 1, the UAV serves as a mobile edge computing server to provide computing migration and data storage services for a group of users on the ground. We assume that there are M randomly distributed ground users in a square area with a side length L^{max} , and the set of users is denoted as $m \in \mathcal{M} = \{1, 2, \dots, M\}$. We define all of the users served by N UAVs where $n \in \mathcal{N} = \{1, 2, \dots, N\}$, which are set to fly in the designated area.

We divide the flight time of the UAV in an episode into T different time intervals, and a single time slot (TS) $t \in \mathcal{T} = \{1, 2, \dots, T\}$. In a specific TS, all users will move to random locations in the area. The positions of users can be expressed as $\mathcal{U}_m(t) = [x_m(t), y_m(t), 0]$, and the locations of

UAVs are denoted as $\mathcal{P}_n(t) = [x_n(t), y_n(t), H]$. We consider that each UAV is flying at a fixed altitude H . In addition, there are obstacles such as buildings and trees in our environment, therefore each UAV should keep a distance from them, as well as other UAVs, during the execution of tasks.

The horizontal distance between the m -th user and the n -th UAV in the t -th TS can be expressed as

$$d_{m,n}(t) = \sqrt{\|x_m(t) - x_n(t)\|^2 + \|y_m(t) - y_n(t)\|^2}. \quad (1)$$

Our communication link between the UAV and users is line-of-sight (LoS). The calculation of channel power gain follows the free space path loss model, which can be expressed as

$$g_{m,n}(t) = \frac{\rho_0}{H^2 + d_{m,n}^2(t)}, \quad (2)$$

where ρ_0 represents the channel's power gain at the reference distance. Then, the uploading data rate from the m -th user to the n -th UAV in the t -th TS can be expressed as

$$R_{m,n}(t) = B_m \log_2 \left(1 + \frac{P_u g_{m,n}(t)}{\sigma^2} \right), \quad (3)$$

where B_m represents the channel bandwidth allocated by the n -th UAV to the m -th user, so that $\sum_{m=1}^{M'} B_m = B$ and M' is number of users served by UAV simultaneously. In order to simplify the model, we assume that the UAV shares the channel bandwidth equally with the serving users. In addition, P_u denotes the transmission power of the m -th user and σ^2 is the power of additive white Gaussian noise.

We also consider the energy consumption generated by the UAV during task execution, mainly including flight energy consumption and computing energy consumption. We assume that the UAV flies at a constant speed $v(t)$, and in one TS, the energy consumed by n -th UAV flight can be defined as

$$E_n^f(t) = \kappa v_n^2(t), \quad (4)$$

where $\kappa = 0.5Mt$, and M is the mass of the UAV [16]. Meanwhile, the computing energy consumption of UAV in one TS can be described as

$$E_n^c(t) = \gamma_c C_n R_n(t) (f_c)^2, \quad (5)$$

where C_n expresses the number of CPU cycles needed for one bit computing, $R_n(t)$ represents the overall achievable sum uploading data rate of n -th UAV in one TS, f_c is the CPU frequency, and γ_c denotes the effective switching capacitance.

The model aims to optimize the trajectory of the UAVs to maximize the quality of service. Thus, the trajectory optimization problem of multi-UAV-assisted edge computing in the dynamic environment can be formulated as

$$\max \frac{\sum_{n=0}^N \sum_{t=0}^T R_n(t)}{NT}, \quad (6a)$$

$$\text{s.t. } \|\mathcal{P}_i(t) - \mathcal{P}_j(t)\| \geq \psi, \quad \forall i, j \in \mathcal{N}, \quad \forall t \in \mathcal{T} \quad (6b)$$

$$\|x_n(t)\| \leq L^{max}, \quad \forall n \in \mathcal{N}, \quad \forall t \in \mathcal{T} \quad (6c)$$

$$\|y_n(t)\| \leq L^{max}, \quad \forall n \in \mathcal{N}, \quad \forall t \in \mathcal{T} \quad (6d)$$

$$\|\mathcal{P}_i(t) - \mathcal{P}_b\| \geq \varphi, \quad \forall n \in \mathcal{N}, \quad \forall t \in \mathcal{T} \quad (6e)$$

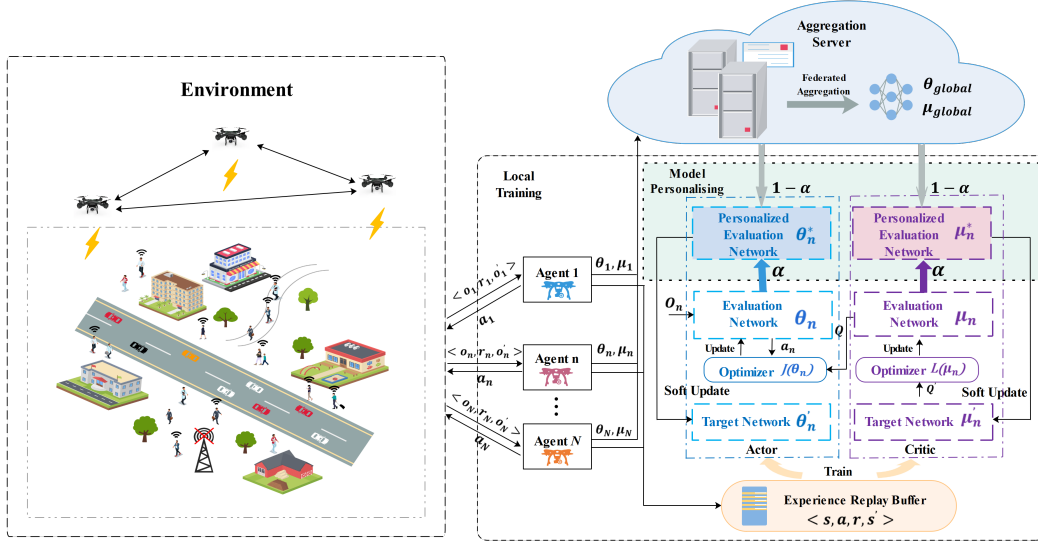


Fig. 1. System model and learning framework of PF-MADDPG

where (6b) represents the minimum distance between UAVs, (6c)-(6d) states UAVs should fly within the specified range, \mathcal{P}_b denotes the position of the obstacles and (6e) explains UAVs should keep a safe distance from obstacles. Our objective is to maximize the real-time information transmission rate between users and UAVs. Note that this problem is non-trivial to solve with traditional optimization methods since it involves continuous state space and action space, and the number of variables is unprecedented. Thus, in this paper, we propose a PF-DRL solution based on the DRL algorithm, which can achieve a fast convergence rate in a complex dynamic environment.

III. PF-MADDPG FOR TRAJECTORY OPTIMIZATION PROBLEM

In this section, we introduce a multi-UAV trajectory optimization scheme based on the PF-MADDPG algorithm. We first use the MADDPG algorithm to train local models. Then, the global model is trained by aggregating local models to improve the convergence rate of the algorithm. Finally, by improving the aggregation process, a personalized training solution is proposed to solve the problem that a single global model may affect the performance of local agents.

A. Problem Transformation

We first formulate the multi-UAV trajectory optimization problem as a multi-agent Markov decision process (MDP). The observation, action and reward function for each agent in t -th TS are defined as follows:

1) *Observation* $o_n(t)$: The uploading data rate is closely related to the distance between the UAV and the user. All UAVs also need to obtain each other's location information.

Thus, each UAV is equipped with a position acquisition device, such as a camera, so that all UAVs can obtain the position information of other UAVs and all users in real time. In summary, the observation space $o_n(t)$ can be expressed as: $o_n(t) = \{\mathcal{P}_n(t), \mathcal{P}_{-n}(t), \mathcal{U}_{m,M}(t)\}$, where $\mathcal{P}_n(t)$ denotes the position of n -th UAV, $\mathcal{P}_{-n}(t)$ denotes the position of other UAVs except n -th UAV, and $\mathcal{U}_{m,M}(t)$ denotes the position of all users.

2) *Action* $a_n(t)$: Each UAV needs to make action decisions in real time according to the observed environment. The action can be written as $a_n(t) = (\Delta x_n(t), \Delta y_n(t), \Delta z_n(t))$ in the TS t , where $\Delta z_n(t) = 0$ since the UAV flies at a fixed altitude.

3) *Reward function* $r_n(t)$: we define the reward function as:

$$r_n(t) = \frac{1}{N} \sum_{n=1}^N \left(\frac{R_n(t)}{R_{max}} - \frac{1}{e_{max}} (E_n^f(t) + E_n^c(t)) \right) - \varepsilon, \quad (7)$$

where R_{max} represents the maximum achievable rate, e_{max} represents the maximum energy consumption in one time slot, and ε denotes the penalty for various collisions and exceeding the boundary.

In the setting of the reward function, on the one hand, we should ensure the maximization of service quality. We should also ensure that the energy consumption of the UAVs is reduced as much as possible, so the energy consumption should have a negative impact on the reward function. Average return value in the random policy is given by $R = \frac{1}{T_0} \sum_{t=1}^{T_0} r_n(t)$, where T_0 denotes the total time step of a training episode.

B. PF-MADDPG based Solution

1) *Local Training*: We use the MADDPG algorithm to train the local model. MADDPG algorithm includes two main components, actor network, and critic network. The actor

Algorithm 1 PF-MADDPG based Trajectory Optimization Algorithm

```

1: Initialize: the position of UAVs  $\mathcal{P}_n$ , users  $\mathcal{U}_m$ ;
2: Initialize: the parameters of each UAV's actor and critic
   evaluation and target networks;
3: Initialize: each UAV's experience replay buffer;
4: for Episode=1,2,..., $N_{max}$  do
5:   Initialize the environment state  $s_0$ ;
6:   for Time step=1,2,..., $T$  do
7:     for each UAV  $n$  do
8:       Obtain observation  $o_n(t)$ ;
9:       Select action  $a_n(t)$  based on policy  $\pi(o_n|\pi_n)$ ;
10:      Take action  $a_n(t)$ , obtain  $r_n(t)$ ;
11:    end for
12:    obtain  $s$  and  $s'$  by mutual communication;
13:    for each UAV  $n$  do
14:      Select  $a^{all}(t)$  and  $r^{all}(t)$ ;
15:      Store sample  $\{s, a^{all}(t), r^{all}(t), s'\}$  into
16:      experience replay buffer;
17:      Randomly select  $k$  samples;
18:      Update critic evaluation network by (9)-(10);
19:      Update actor evaluation network by (11);
20:    end for
21:    Aggregate the model parameters of the actor and
22:    critic evaluation network of each UAV:
23:     $\theta_{global} \leftarrow \sum_n \rho_n \theta_n$ ;
24:     $\mu_{global} \leftarrow \sum_n \rho_n \mu_n$ ;
25:    for each UAV  $n$  do
26:      Model personalized training:
27:       $\theta_n^* \leftarrow \alpha \theta_n + (1 - \alpha) \theta_{global}$ ;
28:       $\mu_n^* \leftarrow \alpha \mu_n + (1 - \alpha) \mu_{global}$ ;
29:      Update parameters of target networks:
30:       $\theta_n' \leftarrow \tau \theta_n^* + (1 - \tau) \theta_n'$ ;
31:       $\mu_n \leftarrow \tau \mu_n^* + (1 - \tau) \mu_n$ ;
32:    end for
33:    Update the current global status and enter  $s'$ ;
34:  end for
35: end for

```

network outputs actions according to the policy π_n , and the critic network evaluates the action by calculating Q-value. In order to improve the stability of the algorithm, two target networks with the experience replay buffer, i.e., DQN, are further inserted.

The global state of the environment can be combined by the observation information of each UAV, expressed as $s(t) = \{o_n(t), \forall n \in \mathcal{N}\}$. Each UAV can acquire each other's observation information by communicating with each other, so all UAVs will acquire the global state s and the next global state s' after one action. Then, the state transition sample $\{s, a_1(t), a_2(t), \dots, a_N(t), r_1(t), r_2(t), \dots, r_N(t), s'\}$ is illustrated and the sample is stored in the experience replay buffer.

Next, by taking k samples from the experience replay buffer, $\{s_i, a_{i,1}(t), \dots, a_{i,N}(t), r_{i,1}(t), \dots, r_{i,N}(t), s'_i\}$, where

$i = 1, 2, \dots, k$, the actor target network outputs the optimal action $a_{i,n}$ under each state s'_i , and the critic target network calculates the target Q-value of the k samples by

$$Q(s_i, a_{i,1}(t), \dots, a_{i,N}(t) | \mu'_n) = r_{i,n} + \gamma Q(s'_i, a'_{i,1}(t), \dots, a'_{i,N}(t) | \mu'_n). \quad (8)$$

The parameters of the critic evaluation network can be updated by minimizing the loss function:

$$L(\mu_n) = \frac{1}{k} \sum_{i=1}^k [Q(s_i, a_{i,1}(t), \dots, a_{i,N}(t) | \mu'_n) - Q(s_i, a_{i,1}(t), \dots, a_{i,N}(t) | \mu_n)]^2; \quad (9)$$

$$\mu_n = \mu_n - \omega \nabla_{\mu_n} L(\mu_n), \quad (10)$$

where ω represents the update step. Then, the parameters of the actor evaluation network can be updated by the policy gradient as:

$$\nabla_{\theta_n} J(\theta_n) = \frac{1}{k} \sum_{i=1}^k \nabla_{\theta_n} Q(\cdot) \nabla_{\theta_n} \pi(o_{i,n} | \theta_n) |_{a_n = \pi(o_{i,n} | \theta_n)}. \quad (11)$$

The parameters of the actor and critic target networks can be softly updated as follows [17]:

$$\theta'_n = \tau \theta_n + (1 - \tau) \theta'_n; \quad (12)$$

$$\mu'_n = \tau \mu_n + (1 - \tau) \mu'_n, \quad (13)$$

where τ represents the update coefficient, generally taking a smaller value.

2) *Federated Aggregation Process*: We set up an aggregation cloud server on the ground. Each agent uploads its local model after a round of training, and the aggregation server collects all the models and obtains the global model through federation averaging, and then distributes the global model to each agent. Thus, by combining the non-independent identically distributed data of agents, the sample utilization is improved.

After the actor and critic evaluation network are trained, the network parameters are updated. Since the target network parameters are obtained from the evaluation network parameters training, each agent only needs to upload the local actor and critic evaluation network parameters. The aggregation server calculates the global parameters after collecting the local parameters of each agent. The global parameters of the actor and critic evaluation network can be calculated as:

$$\theta_{global} = \sum_n \rho_n \theta_n; \quad (14)$$

$$\mu_{global} = \sum_n \rho_n \mu_n, \quad (15)$$

where ρ_n represents the weight of the n -th agent network, and $\sum_n \rho_n = 1$. Each agent updates the parameters of its target network after obtaining the global parameters.

3) *Model Personalization*: FL can effectively improve learning efficiency, however, it is worth noting that a single global model cannot be well generalized to local agents because of the heterogeneity of various data distributions. Ideally, each

agent can use the global model to supplement the problem of fewer local training datasets and alleviate the negative impact of lacking personalization. Thus, the PF-MADDPG algorithm is proposed to solve this problem. In our algorithm, taking the actor evaluation network as an example, the optimization goal of each agent can be described as:

$$\min L(\alpha\theta_n + (1 - \alpha)\theta_{\text{global}}), \quad (16)$$

where α is mixed weight, and $\alpha\theta_n + (1 - \alpha)\theta_{\text{global}}$ is a convex combination of the local model and the global model, namely, the personalized model. So we need to train the personalized model to update the two evaluation networks.

The architecture of the PF-MADDPG algorithm is shown in Fig. 1, and each UAV trains a local model based on its observed environment and uploads the local model to the aggregator. After obtaining the model parameters of the global actor and critic evaluation network, the global and local models are aggregated according to a certain weight ratio α to obtain the personalized model. After receiving the global model, the server will aggregate it with the local model as:

$$\theta_n^* = \alpha\theta_n + (1 - \alpha)\theta_{\text{global}}; \quad (17)$$

$$\mu_n^* = \alpha\mu_n + (1 - \alpha)\mu_{\text{global}}, \quad (18)$$

where θ_n^* and μ_n^* are the personalized actor and critic evaluation model parameters. The size of the mixed weight α will directly affect the performance of the personalized network model, which will be analyzed in the simulation section. Finally, θ_n^* and μ_n^* are used for updating the target network parameters. We provide a pseudo-code of the algorithm in Algorithm 1.

IV. PERFORMANCE EVALUATION

In this section, we will comprehensively evaluate the learning performance of different algorithms. For simulations, we consider a $200 \text{ m} \times 200 \text{ m}$ square area which consists of the random distribution of several dynamic users. Four UAVs serve users in this area, distributed in the four top corners of the square area. A safe distance of 10m shall be kept between UAVs and obstacles. The maximum speed of a UAV is set to 10 m/s, and the maximum speed of users is set to 2 m/s. There are 200 time slots in each episode of training. The average of each ten episodes of training is recorded as the return value of one training episode.

A. Overall Performance Comparison

Fig. 2 shows the convergence of the MADDPG algorithm [10], the F-MADDPG algorithm, and the PF-MADDPG algorithm to optimize the trajectory of 4 UAVs, and ten rounds of experiments are conducted respectively. Four UAVs jointly serve several ground users, and each UAV can simultaneously serve five users, who share the bandwidth of the UAVs. The solid line shows the average results of the ten rounds of experiments, and the shaded part shows the range of fluctuations.

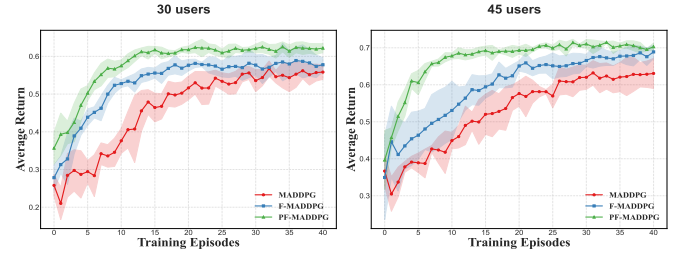


Fig. 2. Performance comparison of different algorithms for 30 users and 45 users scenarios. Compared with the MADDPG algorithm and the F-MADDPG algorithm, the PF-MADDPG algorithm has a faster convergence rate and better average return.

TABLE I
THE PERFORMANCE GAIN

| | | PF-MADDPG | |
|----------|----------|----------------|------------------|
| | | Average Return | Convergence Rate |
| 30 users | MADDPG | 14.5% | 115.4% |
| | F-MADDPG | 10.5% | 38.5% |
| 45 users | MADDPG | 14.3% | 136.4% |
| | F-MADDPG | 7.4% | 81.1% |

The simulation results indicate that the PF-MADDPG algorithm has a faster convergence rate than the MADDPG algorithm and the F-MADDPG algorithm. Conversely, the training process of the MADDPG algorithm and the MADPPG algorithm is exceedingly unstable and subject to considerable fluctuations when compared to the PF-MADDPG algorithm. TABLE 1 provides a comprehensive overview of the performance and convergence rate gain of the PF-MADDPG algorithm across two distinct environments. In general, the PF-MADDPG algorithm showcases superior average return and faster convergence speed, coupled with notable performance improvements.

B. Local Performance Analysis

The simulation analysis above indicates that the PF-MADDPG algorithm significantly improves the overall convergence rate of the system. However, the benefits of personalized federated learning extend beyond just the convergence rate, as it also enhances the local performance of each agent through personalized model training. Thus, in this subsection, we analyze the local performance of a single agent. Fig. 3 illustrates weighted statistics of the local return of each agent, comparing the PF-MADDPG algorithm with the MADDPG algorithm. The simulation analysis shows that the PF-MADDPG algorithm yields a performance gain of up to 13%-18% for each UAV, resulting in substantial improvement in local performance compared to the MADDPG algorithm.

C. Discussion on the Mixed Weight α

In this subsection, we analyze the weight of the global model in the PF-MADDPG algorithm, that is, the aggregate

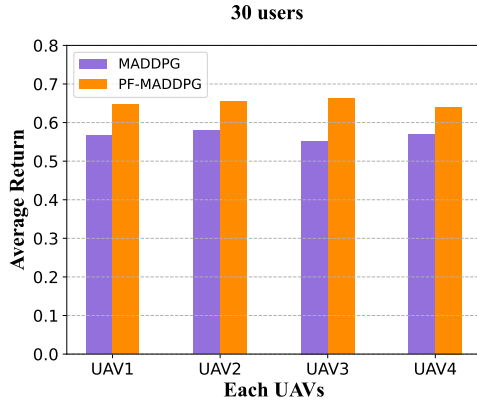


Fig. 3. Comparison of local performance between PF-MADDPG algorithm and MADDPG algorithm in 30 users scenario.

ratio of the local model to the global model. In the experiment, we find that different weight ratios will affect the overall performance of the personalized model. As shown in Fig. 4, we have set several fixed weight ratios. The weight ratios of the private model and the aggregate model are 3:7, 5:5, 7:3, and 9:1 respectively for the simulation experiments, and are compared with the MADDPG algorithm. From the experimental results, it can be concluded that in the UAV trajectory optimization problem, the private model should have more weight, and the optimal ratio should be around 7:3. However, it is difficult to find the best-mixed weight. An optimized α is decided by a combination of several system factors, such as the correlation between a local distribution and global distribution[18], which can be further explored in the future.

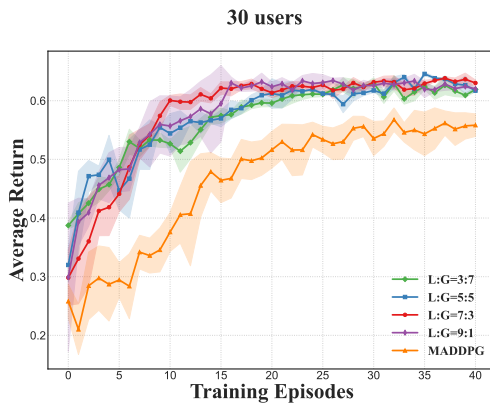


Fig. 4. Performance comparison of personalized federated reinforcement learning algorithms with different mixed weights. The mixture weight ratio shown in the figure is the ratio of the local model to the global model(L: G).

V. CONCLUSION

In this paper, we have proposed a personalized federated deep reinforcement learning algorithm to solve the trajectory optimization problem of multi-UAV in complex dynamic

scenarios. By aggregating local and global models with a certain mixture weight, a well-trained personalized model can be obtained by several rounds of communication. Simulation results show that the proposed scheme can achieve a faster convergence rate and better learning performance, compared with the MADDPG algorithm and the F-MADDPG algorithm.

REFERENCES

- [1] P. Corcoran and S. K. Datta, "Mobile-edge computing and the Internet of Things for consumers: Extending cloud computing and services to the edge of the network," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 73–74, 2016.
- [2] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. Garcia-Rodriguez, and J. Yuan, "Survey on UAV cellular communications: Practical aspects, standardization advancements, regulation, and security challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3417–3442, 2019.
- [3] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.
- [4] M. Guo, T. Long, H. Li, and J. Sun, "Reinforcement-learning-based path planning for UAVs in intensive obstacle environment," in *2021 China Automation Congress (CAC)*, 2021, pp. 6451–6455.
- [5] C. Yan and X. Xiang, "A path planning algorithm for UAV based on improved Q-learning," in *2018 2nd International Conference on Robotics and Automation Sciences (ICRAS)*. IEEE, 2018, pp. 1–5.
- [6] Y. Qian, K. Sheng, C. Ma, J. Li, M. Ding, and M. Hassan, "Path planning for the dynamic UAV-aided wireless systems using Monte Carlo Tree Search," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 6716–6721, 2022.
- [7] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.
- [8] P. Luong, F. Gagnon, L.-N. Tran, and F. Labeau, "Deep reinforcement learning-based resource allocation in cooperative UAV-assisted wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7610–7625, 2021.
- [9] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "Multi-UAV path planning for wireless data harvesting with deep reinforcement learning," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1171–1187, 2021.
- [10] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 73–84, 2020.
- [11] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, 2022.
- [12] A. Gao, Q. Wang, W. Liang, and Z. Ding, "Game combined multi-agent reinforcement learning approach for UAV assisted offloading," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 12 888–12 901, 2021.
- [13] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, "A survey of federated learning for edge computing: research problems and solutions," *High-Confidence Computing*, 2021.
- [14] D. Kwon, J. Jeon, S. Park, J. Kim, and S. Cho, "Multiagent DDPG-based deep learning for smart ocean federated learning IoT networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9895–9903, 2020.
- [15] A. Ziyang Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *arXiv e-prints*, pp. arXiv–2103, 2021.
- [16] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, 2017.
- [17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *Computer ence*, 2015.
- [18] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *arXiv preprint arXiv:2003.13461*, 2020.