Multi-Agent Deep Reinforcement Learning Based UAV Trajectory Optimization for Differentiated Services

Zhaolong Ning, *Senior Member, IEEE*, Yuxuan Yang, Xiaojie Wang, *Senior Member, IEEE*, Qingyang Song, *Senior Member, IEEE*, Lei Guo, and Abbas Jamalipour, *Fellow, IEEE*

***Abstract*—Driven by the increasing computational demand of real-time mobile applications, Unmanned Aerial Vehicle (UAV) assisted Multi-access Edge Computing (MEC) has been envisioned as a promising paradigm for pushing computational resources to network edges and constructing high-throughput line-of-sight links for ground users. Most exsiting studies consider simplified scenarios, such as a single UAV, Service Provider (SP) or service type, and centralized UAV trajectory control. In order to be more in line with real-world cases, we intend to achieve distributed trajectory control of multiple UAVs in UAV-assisted MEC networks with multiple SPs providing differentiated services. Our objective is to minimize the short-term computational costs of ground users and the long-term computational cost of UAVs, simultaneously based on incomplete information. We first solve the formulated problem by reaching the Nash Equilibrium (NE) of the game among SPs based on complete information. We further formulate a Markov game model and propose a Deep Reinforcement Learning (DRL)-based UAV trajectory optimization algorithm, where only local observations of each UAV are required for each SP's flying action execution. Theoretical analysis and performance evaluation demonstrate the convergence, efficiency, scalability, and robustness of our algorithm compared with other representative algorithms.**

***Index Terms*—Multi-access edge computing, UAV-assisted communications, game theory, multi-agent DRL.**

Manuscript received 16 December 2022; revised 17 June 2023; accepted 1 September 2023. Date of publication 5 September 2023; date of current version 4 April 2024. This work was supported in part by the Natural Science Foundation of China under Grants 61971084, 62025105, 62001073, 62221005, and 62272075, in part by the National Natural Science Foundation of Chongqing under Grants cstc2021ycjhbgzxm0039, CSTB2022BSXMJCX0109, and CSTB2022BSXMJCX0110, in part by the Science and Technology Research Program for Chongqing Municipal Education Commission under Grant KJZDM202200601, in part by the Support Program for Overseas Students to Return to China for Entrepreneurship and Innovation under Grants cx2021003 and cx2021053, and in part by the Youth Innovation Group Support Program of ICE Discipline of CQUPT under Grant SCIE-QN-2022-03. Recommended for acceptance by X. Liu. *(Corresponding author: Xiaojie Wang.)*

Zhaolong Ning, Xiaojie Wang, Qingyang Song, and Lei Guo are with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: zhaolongning@gmail.com; xiaojie.kara.wang@ieee.org; songqy@cqupt.edu.cn; guolei@cqupt.edu.cn).

Yuxuan Yang and Abbas Jamalipour are with the School of Electrical and Information Engineering, The University of Sydney, Sydney, NSW 2050, Australia (e-mail: yuxuanyang@mail.dlut.edu.cn; abbas.jamalipour@sydney.edu.au).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2023.3312276, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3312276

I. Introduction

OVER the past decades, cloud computing has become a mature computing paradigm, whose vision is to realize the centralization of computing, communication, storage and network management in cloud facilities [1]. However, the recent surge in the number of smart mobile devices has brought an explosion of mobile data, which in turn is accompanied by the surge in computational demands of latency-sensitive smart mobile applications, such as facial recognition, augmented reality, and smart security networks [2]. Confronted with the increasing demand, a new trend in the computing field has emerged, namely Multi-access Edge Computing (MEC), which pushes functions of the cloud toward the network edge [3]. MEC is considered as a promising technology for supporting applications that consume numerous computing resources to achieve satisfactory performance [4], which is promising to provide users with ultra-low-latency and high-bandwidth network services.

Based on technologies such as network function virtualization, software-defined networking, and information center networking, MEC implements services and functions originally located in the data center at the network edge. With computing, storage, and communication resources deployed at the network edge, heterogeneous user devices can enjoy services from resource-rich infrastructures, such as macro Base Stations (BSs) equipped with MEC servers or Unmanned Aerial Vehicles (UAVs) [5]. It greatly reduces both of the network operational cost and the transmission delay, thus improving the service quality of users.

The MEC network design of multi-user devices requires rational solutions, such as efficient allocation of bandwidths and computational resources for multiple users simultaneously, collaboration between heterogeneous devices and edge network facilities, and efficient deployment of MEC servers. Compared with centralized cloud platforms, computing resources of MEC servers are relatively scarce. Therefore, efficient allocation of various limited resources in multi-user MEC scenarios to achieve system-wide optimization goals, such as minimizing system-wide energy consumption or processing delay, is the hotspot of current research. Without careful configuration of multi-user computational offloading strategies, latency-sensitive or real-time computing service requirements may be limited by scarce computing resources [6].

However, the deployment of edge servers and user computational offloading strategies in MEC networks are coupled with each other, and the condition of the wireless channel between edge servers and user devices changes according to different server deployment, which in turn affects user computational offloading strategies [7]. Therefore, a new MEC architecture needs to be introduced to break through the bottlenecks in terms of resource-allocation-related challenges, such as simultaneous efficient server deployment and user computational offloading.

A. Motivation

With the advantages of flexibility and mobility, UAVs are able to extend the performance of mobile wireless networks by providing ground users with services, such as data collection, fast network access, and edge computing. For high-throughput communication services, with appropriate scheduling, interference-free communications among users can be achieved, so as to efficiently allocate limited communication resources. Also, UAVs as aerial servers can provide edge computing services for ground users [8], namely UAV-assisted MEC.

In the UAV-assisted MEC architecture, low-flying UAVs can be utilized as wireless communication BSs, relays, or servers to improve system performance, and realize efficient computational resource allocation for MEC networks. Unlike ground BSs with fixed locations, UAVs with flexible mobilities, are able to move closer to ground users, thereby establishing high-throughput Line-of-Sight (LoS) links. By deploying MEC-enabled UAVs, we can not only address the issue of user computation offloading and edge service deployment at the same time, but also save the construction cost of communication and computing infrastructures, and provide computing resources for on-ground users. However, the flexible mobility of UAVs also brings challenges to the efficient deployment and dispatch of UAVs, especially in multi-UAV cases. Therefore, realizing the trajectory optimization of UAVs in free space has become an urgent problem to be solved in the UAV-assisted MEC network.

In addition, efficient allocation of various resources is based on the binary service preferences of users, i.e., users need to determine which type of service they request with probability 1. But in real-world scenarios, when choosing services, users are affected by various factors, which make their service choices uncertain, including personal issues (do not have a strong demand for one service, and just randomly choose according to momentary whim) or network conditions (changes in service demand due to changes in time-varying network conditions). Therefore, we can model such uncertainty with a non-binary variable, i.e., the non-binary preferences of users. In other words, user preferences for different service types are generated with time-varying probabilities. In addition, most of the existing studies are based on a single Service Provider (SP) case. According to the 5G Architecture White Paper V4.0 [9] released by 5GPPP Architecture Working Group, the integration of Unmanned Aerial Systems (UAS) composed of UAVs and 5G networks is considered to be crucial to meet the Quality of Service (QoS) defined by 3GPP, GSMA and other requirements. More and more SPs including Huawei [10], Nokia [11] and Qualcomm [12] have proposed their own 5G UAV-assisted wireless network services. Therefore, research on the simultaneous deployment of differentiated edge computing services for multiple users in the same target area by multiple SPs is needed.

Therefore, based on above statements, our research motivations are summarized as follows:

- We intend to realize distributed multi-UAV trajectory control in UAV-assisted MEC networks with multiple SPs and non-binary service preferences of ground users, thereby minimizing the short-term computational cost of ground users and the long-term computational cost of UAVs, simultaneously.
- We intend to provide a feasible multi-agent Deep Reinforcement Learning (DRL) approach for the game of multiple SPs, which is promising to lay a solid foundation for multiple SPs deploying multiple UAVs to provide multiple service types for ground users. However, most existing studies only considered one SP or one UAV, and did not consider the preferences of ground users.
- We intend to achieve decentralized trajectory control of UAVs, where each SP has no knowledge of users' service preferences or the policies of other SPs when executing UAV's flying action, i.e., based merely on the local observations of UAVs, which is in line with reality.

B. Contributions

Most existing UAV-assisted MEC studies considered simplified scenarios. For example, some of them considered a single UAV, SP, or service type, and some of them considered deterministic user offloading. In addition, there are also shortcomings, such as the lack of location exploration and centralized control in trajectory optimization, which will be elaborated in Section II. Based on these shortcomings and our research motivations, in this paper, we study a UAV-assisted MEC network with differentiated services, where multiple SPs exist in the target area, and UAVs are deployed to provide differentiated edge computing services to ground users with time-varying service preferences. Each SP considers the optimization of UAV trajectories over a period of time. In addition, the execution of UAV's flying actions is based merely on local observations without additional system information, i.e., we achieve a distributed solution for multi-UAV trajectory optimization. Our main contributions are summarized as follows:

- We consider a realistic scenario of the UAV-assisted MEC network with multiple SPs and differentiated services, where ground user generates computational tasks with time-varying and non-binary service preferences, and multiple UAVs owned by different SPs fly over the target area serving as edge servers.
- We formulate communication, computing, and flying models for the considered scenario, and propose a short-term computational cost minimization problem for users and a long-term computational cost minimization problem for UAVs.
- We investigate the interaction among SPs based on complete system information. With the existence conditions of Nash Equilibrium (NE), we prove the uniqueness of NE

and solve the computational cost minimization problems of both users and UAVs.
- We formulate the Markov game model, and propose a multi-UAV trajectory optimization algorithm to control the trajectory of UAVs belonging to different SPs based on multi-agent DRL distributively, where only local observations are required for flying action executions. Our objective is to minimize the long-term computational cost of UAVs and the computational cost of users simultaneously based on incomplete information.
- We conduct theoretical analysis and performance evaluation to demonstrate the convergence, efficiency, scalability, and robustness of our proposed algorithm, which can achieve the lowest overall computational cost compared with other representative algorithms.

The rest of this paper is organized as follows. In Section II, we summarize the related work. The system model is illustrated in Section III. Two minimization problems are formulated in Section IV. In Section V, we analyze the interaction of SPs and propose three NE-related theorems. In Section VI, we introduce our proposed solution in detail, and analyze the convergence of our proposed algorithm. In Section VII, we evaluate the performance of our proposed solution. Conclusions are drawn in Section VIII.

II. Related Work

In order to minimize the processing delay and energy consumption of computational tasks generated by mobile devices, one challenge of offloading computation-intensive tasks to edge servers is to achieve efficient computation offloading, which has been studied in previous work [23], [24], [25], [26], [27]. There are generally two types of computation offloading, namely partial computation offloading and binary computation offloading. Partial computation offloading allows users to offload part of the task to the MEC server, and process the rest locally. Binary computation offloading does not allow task splitting, and all computational tasks need to be completely processed on MEC servers or user's local devices. For example, the authors in [24] proposed an efficient partial computation offloading and adaptive task scheduling algorithm to maximize the system-wide profit in vehicular networks by integrating game theory and convex optimization.

In recent years, the development of artificial intelligence, especially DRL, has provided new solutions to the challenges faced by various mobile network architectures, such as DRL-based MEC and Mobile Crowd-Sensing (MCS). In [27], the authors jointly considered server selection, cooperative offloading, and handover in MEC networks, and proposed a DRL-based algorithm to solve non-convex problems to minimize the computational cost. The authors in [28] studied the cold-start problem in MCS, and proposed a novel framework based on DRL to achieve intelligent participant selection, with the objective of minimizing the cold-start effect. An intelligent transportation system was studied in [29], where the authors constructed a blockchain enabled crowdsensing framework, and a satisfied trade-off between blockchain safety and latency was achieved by the DRL-based algorithm.

With the increasing popularity of UAVs, UAV-assisted systems have been widely investigated, especially the UAV-assisted MEC architecture. Previous studies mainly focus on the following two topics, i.e., computational resource allocation [30], [31], [32], [33], and UAV trajectory optimization [13], [14], [34], [35]. For the first one, the authors in [30] studied the interactions among Internet of Things (IoT) devices, UAVs, and edge clouds, and proposed an algorithm based on successive convex approximation, to jointly optimize UAV locations, communication as well as computation resource allocation, and task splitting decision-making. The authors in [31] proposed an efficient resource trading mechanism, which analyzes historical statistics related to future resource supplies, demands, and air-to-ground communication qualities, to maximize the expected utility of MEC servers and UAVs.

Generally, there are two cases for UAV trajectory design in UAV-assisted MEC: 1) Clustering ground users, i.e., dividing several hotspots according to the distribution characteristics of users, and these hotspots are utilized as the flying destinations of UAVs; 2) The trajectory of the UAV is determined by the continuous flying direction angle and distance, i.e., free space trajectory design. For example, in [13], the authors gathered multiple users into independent communities and constructed a 5G-enabled UAV-to-community offloading system, where an inter-community auction mechanism and a dynamic task scheduling algorithm were proposed to achieve the optimal trajectory design for the UAV and maximize the system throughput. The authors in [14] studied an energy-saving UAV-enabled MEC framework combined with non-orthogonal multiple access, and proposed an efficient iterative algorithm to solve two sub-problems alternately, aiming at achieving optimal allocations of bandwidths and computing resources, and UAV trajectory optimization.

However, resource allocation and trajectory optimization in existing studies of UAV-assisted MEC are based on deterministic user offloading, while users tend to generate probabilistic time-varying service preferences in practice. Such unknown user preferences make traditional optimization methods ineffective, such as alternating successive convex approximations. Furthermore, we summarize related work [13], [14], [15], [16], [17], [18], [19], [20], [21], [22] on trajectory optimization of UAV-assisted MEC in Table I. It can be found that the existing work has the following defects: 1) Only a single UAV is considered, which cannot meet the increasing service demand; 2) Based on discrete community division and user clustering, the UAV cannot fully explore potential locations, which makes the trajectory less likely to be further optimized; 3) The trajectory design of the UAV is based on complete system information, while in reality, each UAV flies merely based on local observation; 4) Studies based on DRL integrates the entire system as one agent instead of multiple agents, which cannot be applied to the case with multiple SPs.

In summary, to the best of our knowledge, our work is the first study to realize distributed multi-UAV trajectory control in multi-SP scenarios with probabilistic time-varying service preferences.

TABLE I
COMPARISON OF OUR WORK WITH EXISTING STUDIES

Reference	Single UAV	Multi-UAV	Cluster-based	Free space	Complete info.	Local obs.	Single SP	Multi-SP
[13]	✓		✓		✓		✓	
[14]		✓		✓	✓		✓	
[15]		✓	✓		✓		✓	
[16]	✓			✓	✓		✓	
[17]		✓		✓	✓		✓	
[18]		✓		✓	✓		✓	
[19]		✓		✓	✓		✓	
[20]	✓			✓	✓		✓	
[21]	✓			✓	✓		✓	
[22]		✓		✓	✓		✓	
Our work		✓		✓		✓		✓

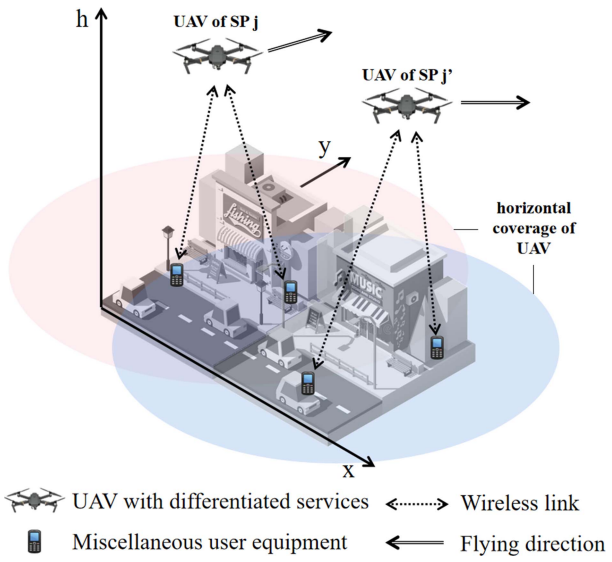


Fig. 1. Illustration of UAV-assisted differentiated services.

III. SYSTEM MODEL

As shown in Fig. 1, we consider a UAV-assisted MEC network with multiple SPs, where each SP deploys its own UAVs according to the commercial strategy to provide differentiated services for ground users. Differentiated services mean that users may request different types of services, which can either be computation-intensive or delay-sensitive applications. UAVs owned by different SPs fly over the target area to provide specific types of services. SPs control their UAVs through efficient strategies to maximize their respective rewards. Furthermore, similar with [36] and [37], we also assume the existence of a network operator, such as BS, which can obtain the system state of the entire network and can send user requirements to different SPs.

For the service requests of users, a service preference variable within $[0, 1]$ is generated for each service type when users generate computational tasks. Users' service preferences are probabilistic rather than binary. The binary preference setting is result-driven, i.e., a posteriori setting. However, when choosing services, even if the user is a rational decision-maker, there are various unknowable factors in the decision-making process that leads to uncertainty in its action executions. In addition, SP tends to pre-plan UAV trajectories to reduce the communication and computational cost and enhance the robustness to dynamic environments, rather than repeatedly re-planning trajectory according to the requirements of users. Therefore, we adopt non-binary preference setting, where users randomly generate corresponding service preferences for each type of service in the range of $[0, 1]$ and summed as 1.

Based on the above analysis, we model the system as follows. We consider an MEC network with N users and M SPs to provide differentiated services. Each SP deploys a UAV in the target area, and each UAV is equipped with computing resources as an edge server to serve ground users in the target area. Specifically, each UAV provides a specific type of service. The sets of users and UAVs are denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$ and $\mathcal{M} = \{1, 2, \ldots, M\}$, respectively. Since the system operates in a time-slot manner, we assume that each user generates a computational task in each time slot $t \in \mathcal{T} = \{0, 1, \ldots, T\}$. These computational tasks can be offloaded to the UAV for processing, or processed locally. At the same time, users generate service preferences for their computational tasks. We define the service preference of user i as follows:

$$l_{ij}(t) \in [0, 1], \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}. \tag{1}$$

Symbol $l_{ij}(t)$ represents the service preference of user i to type j at time slot t. Equation $l_{ij}(t) = 1$ means that user i selects service type j, i.e., the task is completely offloaded to UAV j, and otherwise $l_{ij}(t) = 0$. For $l_{ij}(t) \in (0, 1)$, it indicates that the user has a non-binary preference for service type j. From the perspective of the UAV, $l_{ij}(t) \in (0, 1)$ means that user i offloads its computational task to UAV j at time slot t with a probability.

Thus, the computational task generated by user i at time slot t is defined as follows:

$$\mathcal{I}_i(t) = \{D_i(t), F_i(t), \mathbf{l}_i(t)\}, \forall i \in \mathcal{N}, t \in \mathcal{T}, \tag{2}$$

where $D_i(t)$ represents the data size of computational task $\mathcal{I}_i(t)$ at time slot t, and $F_i(t)$ describes the number of CPU cycles required for computational task $\mathcal{I}_i(t)$ at time t. Vector $\mathbf{l}_i(t) = \{l_{ij}(t), \forall j \in \mathcal{M}\}$ represents the service preference of user i for various types.

Note that we consider the UAV deployment within a period of time. In each time period, the SP deploys a UAV over the target area. At the end of each time period, according to the new users' service preference pattern, UAV's trajectory is replanned. Emerging gas-powered or hybrid-powered UAVs can achieve

a considerable working duration, such as HYBRIX 2.1 [38], whose flight time can be as long as 10 hours and 14 minutes in the latest test flight. Therefore, we do not consider the recharging issue of UAVs in this work. In addition, algorithms related to UAV recharging, such as [39] and [40], can be applied to the system we considered.

A. Communication Model

Based on the above system model, if user $i \in \mathcal{N}$ only needs service type j at time slot t, i.e., choosing to offload the computational task to UAV $j \in \mathcal{M}$ for edge computing, uplink data transmission rate $\hat{r}_{ij}(t)$ is defined as follows:

$$\hat{r}_{ij}(t) = B_w \log_2 \left(1 + \frac{\alpha P_i(t)}{H_j(t)^2 + R_{ij}(t)^2}\right), \tag{3}$$

where B_w is the bandwidth of the wireless channel, and $P_i(t)$ is the transmit power of user i at time slot t. Symbol $H_j(t)$ represents the hovering height of UAV j at time slot t, and $R_{ij}(t)$ represents the horizontal distance between user i and UAV j at time slot t. Variable $\alpha = g_0 G_0/\sigma^2$, where $G_0 \approx 2.2846$ [32]. Symbol g_0 represents the channel gain per unit distance (1 m), and σ^2 represents the background noise power. It should be noted that the inter-user communication interference in this paper is ignored, since many related studies based on Orthogonal Frequency-Division Multiple Access (OFDMA) have achieved this and can be applied to the system we considered [41], [42], [43]. For example, the authors in [41] and [42] divide the total bandwidth into multiple subcarriers based on OFDMA. By defining allocation variables, each subcarrier is dedicated to one user in a time slot, thus ensuring the orthogonality among users on each subcarrier in each time slot. Other technologies such as Orthogonal Multiple Access (OMA) can also be utilized in our system [44]. Via dividing time slots, each sub-slot is assigned to a user, so that there is no mutual interference among users using the same channel for communication and transmission. Therefore, the interference among users can be ignored in our system.

Horizontal distance $R_{ij}(t)$ between user i and UAV j at time t is defined as follows:

$$R_{ij}(t) = \left\| \mathbf{q}_j(t) - \mathbf{p}_i \right\|, \tag{4}$$

where vector $\mathbf{q}_j(t) = (x_j(t), y_j(t))$ is the horizontal location coordinates of UAV j at time slot t, and vector $\mathbf{p}_i$ represents the horizontal location coordinates of user i. We assume that users have low mobility in the considered time slot. Furthermore, we assume that each UAV moves within a rectangular target area [18], i.e., horizontal location coordinates satisfy the following constraints:

$$x_j(t) \in [0, x_{\max}], \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{5}$$

$$y_j(t) \in [0, y_{\max}], \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{6}$$

where $x_{\max}$ and $y_{\max}$ represent the length and width of the target area, respectively.

In addition, our communication model mainly focuses on uplink data transmission and does not consider result backhaul. This is because the communication overhead of result backhaul is negligible compared to that of task offloading.

B. Computing Model

When user $i \in \mathcal{N}$ only needs service type j at time slot t, i.e., choosing to offload computational task $\mathcal{I}(t)$ to UAV $j \in \mathcal{M}$ for edge computing, the transmission energy consumption of user i offloading task $\mathcal{I}(t)$ to UAV j at time slot t can be calculated by:

$$E_{ij}^{\mathrm{Tr}}(t) = P_i T_{ij}^{\mathrm{Tr}}(t), \tag{7}$$

$$T_{ij}^{\mathrm{Tr}}(t) = \frac{D_i(t)}{\hat{r}_{ij}(t)}, \tag{8}$$

where $T_{ij}^{\mathrm{Tr}}(t)$ is the transmission delay. The computational energy consumption caused by UAV j processing computational task $\mathcal{I}(t)$ at time slot t can be calculated by:

$$E_{ij}^{\mathrm{C}}(t) = \eta_j (f_j^C)^{\beta_j} T_{ij}^{\mathrm{C}}(t), \tag{9}$$

$$T_{ij}^{\mathrm{C}}(t) = \frac{F_i(t)}{f_{ij}^C(t)}, \tag{10}$$

where $T_{ij}^{\mathrm{C}}(t)$ is the processing delay of computational task $\mathcal{I}(t)$. Variable η_j is a positive value, indicating an effective switched capacitance, which is used to measure the energy consumption of UAV j per unit time to process computational tasks. Variable β_j is the positive constant, which is usually set to 3 [45]. Symbol $f_{ij}^C(t)$ represents the computing capability obtained by user i from UAV j at time slot t, where $f_{ij}^C(t) = f_j^C/N_j$. Symbol N_j represents the number of users within the service coverage of UAV j, and a detailed explanation is provided in Section IV, and f_j^C represents the computing capability of UAV j.

In real-world scenarios, although each UAV as an edge server is equipped with high computing capability, the total amount of data that can be processed at the same time has an upper limit. We define $\bar{D}_j$ as the threshold data value of UAV j, which is the upper bound of the total amount of data that UAV j can process at the same time. Given the service preferences of users, the constraint of threshold data value is defined as follows:

$$\sum_{i \in \mathcal{N}_j} l_{ij}(t) D_i(t) \le \bar{D}_j, \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{11}$$

where set $\mathcal{N}_j \subseteq \mathcal{N}$ represents N_j users within the service coverage of UAV j.

If a user in the target area is not within the service coverage of any UAV, the user can only choose to process its computational tasks on the local device ($j = 0$). Similarly, user i's computation energy consumption brought by local processing of computational task $\mathcal{I}(t)$ at time slot t is defined as follows:

$$E_i^{\mathrm{L}}(t) = \eta_i (f_i^L)^{\beta_i} T_i^{\mathrm{L}}(t), \tag{12}$$

$$T_i^{\mathrm{L}}(t) = \frac{F_i(t)}{f_i^L}, \tag{13}$$

where $T_i^{\mathrm{L}}(t)$ is the delay caused by the local device to process task $\mathcal{I}(t)$. Similarly, η_i is a positive value, indicating a switched capacitance, which is used to measure the energy consumed

by local device i processing computational tasks per unit time. Variable β_i is also usually set as 3, and f_i^L represents the computing capability of user i's local device.

To sum up, the total energy consumption of processing task $\mathcal{I}(t)$ at time slot t is given by:

$$E_{ij}(t) = \begin{cases} E_{ij}^{\mathrm{Tr}}(t) + E_{ij}^{\mathrm{C}}(t), & \text{if } j \in \mathcal{M}, \\ E_i^{\mathrm{L}}(t), & \text{if } j = 0. \end{cases} \tag{14}$$

Meanwhile, the total delay of processing task $\mathcal{I}(t)$ at time slot t is given by:

$$T_{ij}(t) = \begin{cases} T_{ij}^{\mathrm{Tr}}(t) + T_{ij}^{\mathrm{C}}(t), & \text{if } j \in \mathcal{M}, \\ T_i^{\mathrm{L}}(t), & \text{if } j = 0. \end{cases} \tag{15}$$

In the MEC architecture, users can send computational tasks to UAV-mounted edge servers following a time-slot manner. In each time slot, we use data packets to encapsulate computational tasks, and each time slot is an atomic time unit. Therefore, the processing delay of computational tasks of different service types is measured at the same scale. Furthermore, we do not consider services with high criticalities, such as rescue services. Without loss of generality, similar with [15] and [18], we consider the same delay constraint of different time slots for the generated tasks, i.e.,

$$T_{ij}(t) \leq T_{\max}, \forall i \in \mathcal{N}, j \in \mathcal{M}', t \in \mathcal{T}, \tag{16}$$

where $T_{\max}$ is the upper bound of processing delay. It should be noted that for different types of services, their required time slots (delays) are different. By setting $\mathcal{M}'$ as an extension of set $\mathcal{M}$, local computing is taken into account, i.e., $\mathcal{M}' = \mathcal{M} \cup \{0\}$.

C. Flying Model

Since we consider the horizontal flight of the UAV over the target area, the flying movement of the UAV needs to be modeled. During time slot t, flying horizontal azimuth $\theta_j(t)$ of UAV j and flying distance $d_j(t)$ should satisfy the following constraints, respectively:

$$\theta_j(t) \in [0, 2\pi], \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{17}$$

$$d_j(t) \in [0, d^{\max}], \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{18}$$

where $d^{\max}$ represents the maximum distance that a UAV can fly during time $t \in \mathcal{T}$, and the flying distance can be obtained by:

$$d_j(t) = \|\mathbf{q}_j(t+1) - \mathbf{q}_j(t)\|. \tag{19}$$

In addition, the horizontal location coordinates of UAV j at time t can also be obtained by the horizontal azimuth $\theta_j(t)$ and the flying distance $d_j(t)$ by:

$$\begin{aligned} x_j(t) &= x_j(0) + \sum_{\tau=1}^{t} d_j(\tau)\cos(\theta_j(\tau)), \\ y_j(t) &= y_j(0) + \sum_{\tau=1}^{t} d_j(\tau)\sin(\theta_j(\tau)). \end{aligned} \tag{20}$$

IV. Problem Formulation

Due to the limitation of UAV horizontal service coverage, symbol $R_j^{\max}$ is used to represent the maximum horizontal service radius of UAV j. We define symbolic function $\mathcal{U}_{ij}(x)$ to determine whether a user is within the coverage of UAV:

$$\mathcal{U}_{ij}(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{if } x \leq 0. \end{cases} \tag{21}$$

By setting $x = R_{ij}(t) - R_j^{\max}$, according to equation (21), if user i is within the coverage of UAV j, i.e., $R_{ij}(t) - R_j^{\max} > 0$, then $\mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max}) = 1$, otherwise $\mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max}) = 0$. Based on this, we can leverage equation (22) to determine whether user i should choose local computing:

$$\prod_{j \in \mathcal{M}} \mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max}) = \begin{cases} 1, & \text{if } j = 0, \\ 0, & \text{if } j \in \mathcal{M}. \end{cases} \tag{22}$$

This is because if user i chooses local computing, it means that user i is not within the service coverage of any UAV, i.e., $R_{ij}(t) - R_j^{\max} > 0, \forall j \in \mathcal{M}$, and $\prod_{j \in \mathcal{M}} \mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max}) = 1$. Otherwise, user i is within the service coverage of at least one UAV.

Based on the above-defined symbolic function and the system model in Section III, the computation cost of user i defined from the perspective of energy consumption is as follows:

$$\begin{aligned} C_i(t) = &\sum_{j \in \mathcal{M}} [(1 - \mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max}))l_{ij}(t)E_{ij}^{\mathrm{Tr}}(t)] \\ &+ \prod_{j \in \mathcal{M}} \mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max})E_i^{\mathrm{L}}(t). \end{aligned} \tag{23}$$

It can be observed from equation (23) that the user considers: a) its service preference and transmission energy consumption when edge computing is selected; b) merely the energy consumption caused by local computing. From the perspective of users, total delay $T_{ij}(t)$ of computational task processing can be reformulated as follows:

$$\begin{aligned} T_{ij}(t) = &\sum_{j \in \mathcal{M}} [(1 - \mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max}))l_{ij}(t)(T_{ij}^{\mathrm{Tr}}(t) \\ &+ T_{ij}^{\mathrm{C}}(t))] + \prod_{j \in \mathcal{M}} \mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max})T_i^{\mathrm{L}}(t). \end{aligned} \tag{24}$$

From equation (24), we note that when discussing the total processing delay, in the case of edge computing, not only the transmission delay but also the computing delay should be considered. Then, the computational cost of UAV j from the viewpoint of energy consumption is defined as follows:

$$\begin{aligned} U_j(t) = &\sum_{i \in \mathcal{N}} l_{ij}(t)[(1 - \mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max}))E_{ij}^{\mathrm{Tr}}(t) \\ &+ \prod_{j \in \mathcal{M}} \mathcal{U}_{ij}(R_{ij}(t) - R_j^{\max})E_i^{\mathrm{L}}(t)]. \end{aligned} \tag{25}$$

In equation (25), users in the target area can be divided into three categories. Taking scenarios with two SPs as an example, Fig. 2 shows a schematic diagram of the service coverage of two UAVs:

We take the perspective of UAV 1 as an example. For users within its service coverage, i.e., $R_{i1}(t) < R_1^{\max}$ is satisfied, their computational costs naturally need to be considered in the cost of UAV 1, and need to be multiplied by service preference $l_{i1}(t)$.

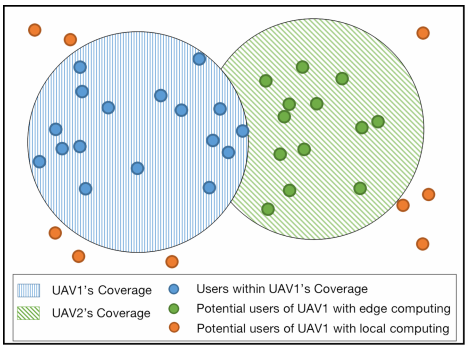


Fig. 2. Schematic diagram of the service coverage.

For users outside its service coverage, they can be divided into two categories, i.e., potential users with edge computing, and potential users with local computing. For potential users with edge computing, since they are already within the service coverage of UAV 2, UAV 1 no longer considers their computational costs. For users with local computing, although they are not within the service coverage of UAV 1 at time slot t, they still have preferences for service type 1, and their computational costs also need to be considered by UAV 1.

From the optimization viewpoint, UAV 1 needs to optimize its flying trajectory to reduce computational costs of users with local computing. Therefore, even if users choose local computing, UAV 1 needs to consider their computational costs with their service preferences.

From the viewpoint of user i, if it is within the service coverage of UAV 1, but not within that of UAV 2, the user only considers the computational cost and preference related to the service provided by UAV 1. If the user is within the service coverage of both UAVs, the corresponding computational costs and preferences of both types of services are considered. If the user is not within the service coverage of any UAV, it can only choose local computing, regardless of service preferences.

Based on the computational costs of the user and the UAV at time slot t in equations (23) and (25), we formulate the computational cost minimization problems of users and UAVs, respectively. First, for all users, the optimization objective is to minimize the computational costs of all users at time slot t, which is defined as follows:

$$\text{P1}: \min_{\mathbf{q}_j(t)} \sum_{i\in\mathcal{N}} C_i(t), \forall j \in \mathcal{M}, t \in \mathcal{T},$$

s.t.

$$\text{C1}: 0 \le x_j(t) \le x_{\max}, \forall j \in \mathcal{M}, t \in \mathcal{T},$$

$$\text{C2}: 0 \le y_j(t) \le y_{\max}, \forall j \in \mathcal{M}, t \in \mathcal{T},$$

$$\text{C3}: \sum_{i\in\mathcal{N}_j} l_{ij}(t) D_i(t) \le \bar{D}_j, \forall j \in \mathcal{M}, t \in \mathcal{T},$$

$$\text{C4}: T_{ij}(t) \le T_{\max}, \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}, \quad (26)$$

where constraints C1 and C2 ensure that UAVs are within the target area. Constraint C3 makes the total amount of data with preferences processed by each UAV at the same time slot not exceed its threshold data value. Constraint C4 guarantees that all computational tasks are processed within the maximum processing delay.

Users as rational decision-makers, inevitably make selfish decisions. They hope that all UAVs fly as close as possible to their locations, but this inevitably leads to an increase in computational costs of other users. To balance the selfish decision-making of users, we intend to minimize the sum of computational costs of all users at time slot t. Differently, the UAV needs to achieve trajectory optimization, and the location hovering at each time slot needs to be considered. That is, the UAV considers the long-term computational cost minimization, which is defined as follows:

$$\text{P2}: \min_{\mathbf{q}_j(t)} \sum_{t\in\mathcal{T}} U_j(t), \forall t \in \mathcal{T},$$

s.t.

$$\text{C1} - \text{C4 in P1},$$

$$\text{C5}: \|\mathbf{q}_j(t+1) - \mathbf{q}_j(t)\| \le d_{\max}, \forall j \in \mathcal{M}, t \in \mathcal{T}, \quad (27)$$

where constraint C5 means that the UAV flying distance during each time slot does not exceed maximum distance $d_{\max}$.

Solving Problems P1 and P2 simultaneously is rather challenging, and this is because:

- Although computational costs considered in the two problems partially overlap, users and UAVs consider different parts of the system-wide computational cost. Users focus on each time slot, while the UAV focuses on the long-term one.
- The existence of unknown information makes problem difficulty in solving. First, the SP does not know each user's service preference in advance, which makes P1 and P2 impossible to solve. In addition, game-theory-based algorithms select the best action based on the obtained utilities by game players, while in our framework, the computational cost of each UAV depends on unknown user preferences. As as result, each UAV needs to iterate with the environment repeatedly to learn its optimal strategy, making traditional game theory methods inefficient. Furthermore, for online learning, each UAV needs to learn its strategy independently without knowing the strategies of other UAVs.
- The nonlinear nature of optimization functions with the existence of unknown parameters makes traditional optimization methods ineffective.

The above reasons complicate the solutions of the formulated two optimization problems. Typically, multi-agent inverse reinforcement learning algorithms are used for online training with unknown rewards. However, this kind of method is only applicable when the reward function can be decoupled from the environment. In our system model, the computational cost of the UAV depends largely on users' preferences, so that the multi-agent inverse reinforcement learning based algorithms are not suitable for solving the formulated problems.

In the next section, we analyze the interaction among SPs, and provide a theoretical basis for proposing a multi-UAV trajectory optimization algorithm based on multi-agent DRL, which is suitable for solving the optimization problems of multiple players with unknown parameters.

V. SP Interaction Analysis

We first analyze the interaction among SPs based on the complete state information of the system, and give the existence condition and uniqueness proof of NE, since Problem P2 can be solved by reaching the NE. Afterward, we prove that when Problem P2 reaches the optimal solution, Problem P1 can also reach its optimal solution.

A. Condition of Nash Equilibriums

When the constraints of Problem P2 are satisfied, the NE condition for multi-SPs is defined as follows:

Theorem 1: Given the case of satisfying constraints of Problem P2, if optimal UAV trajectory $\mathbf{q}_j^*(t)$ satisfies the following equation:

$$\sum_{i\in\mathcal{N}} l_{ij}(t)\left(1-\prod_{j\in\mathcal{M}}\mathcal{U}_{ij}\right)(\mathbf{q}_j^*(t)-\mathbf{p}_i)=0, \tag{28}$$

then the NE can be reached.

Based on user preferences, each SP optimizes its UAV trajectory and minimizes its computational cost. Thus, the core idea of reaching NE is reaching the stationary point of computational cost to UAVs trajectory. Details can be found in Appendix A, available online.

From the condition of NE in equation (28), we note that the NE is affected by users preferences $l_{ij}(t), \forall i \in \mathcal{N}$, UAVs coverage pattern $\mathcal{U}_{ij}, i \in \mathcal{N}, j \in \mathcal{M}$ and location of UAVs $\mathbf{q}_j^*(t)$, $j \in \mathcal{M}$. With the complete state, users preferences and UAVs coverage pattern are known by SPs. Thus, by further analyzing the condition of NE, the uniqueness of NE can be derived in Theorem 2 when the condition in Theorem 1 is satisfied.

Theorem 2: If optimal UAV trajectory $\mathbf{q}_j^*(t)$ satisfies the NE condition in Theorem 1, there is a unique NE for the cost competition among different SPs.

The key to the proof is analyzing the relationship between UAV trajectory and the condition of NE, and details can be found in Appendix B, available online.

So far, the optimal solution of the long-term computational cost minimization problem of UAV, i.e., Problem P2 can be obtained based on Theorems 1 and 2. Then, the optimal solution of the user computational cost minimization problem, i.e., Problem P1 can be obtained according to the following theorem, by studying the solutions of both Problems P1 and P2:

Theorem 3: Optimal solution $\mathbf{q}_j^*(t), \forall j \in \mathcal{M}$ of the UAV long-term computational cost minimization problem P2 is also the optimal solution of the user's computational cost minimization problem P1.

This theorem can be proved by contradiction. Please refer to Appendix C, available online for details. Based on above theorems, we can deduce the following corollary:

Corollary 1: Computational cost minimization problems P1 and P2 can be solved simultaneously by reaching the NE, which satisfies the condition in Theorem 1. Such an NE solution also satisfies the constraints in both Problems P1 and P2.

Please refer to Appendix D, available online for proof details. The above analysis is based on the complete system information, i.e., each SP knows in advance users' preferences and strategies of UAVs owned by other SPs at each time slot. However, it is impossible to realize multi-UAV trajectory optimization by solving the NE of each time slot due to the time-varying nature of the network. When SPs actually start to deploy UAVs, the network state, including user preferences and channel status, has changed. To grasp the long-term network characteristics, a pre-trained model is needed. Furthermore, the SPs may not share their flying strategies, and complete information communication inevitably increases communication delay and overhead. We intend to realize the distributed multi-UAV trajectory design, i.e., each UAV conducts flying actions based on its local observations. Therefore, we formulate a Markov game model for multi-UAV trajectory optimization in the following subsection.

B. Markov Game Formulation

The computational cost minimization problem of different UAVs (or their corresponding SPs) can be modeled as a Markov game, which is an extension of the Markov decision process. UAVs can be regarded as M different learning agents, and the game can be represented by tuple $\langle M, S, O, A, P, \mathbb{R}, \gamma\rangle$. The elements of the tuple are described as follows:

- *State:* The state space of the Markov game can be represented by $S = \{s(t) = (S_1, S_2, S_3)\}, t \in \mathcal{T}$, where S_1 contains location coordinates $\mathbf{p}_i$ of users $\forall i \in \mathcal{N}$ and S_2 contains trajectory information $\mathbf{q}_j(t)$ of all UAVs, $\forall j \in \mathcal{M}$, while S_3 contains the relevant information of the user's computational task $(D_i(t), F_i(t), \mathbf{l}_i(t))$.
- *Observation:* For each UAV, it cannot observe complete network state $s(t)$, but only part of the network state is available, denoted as $O = \{o(t) = \{o_j(t), \forall j \in \mathcal{M}\}, t \in \mathcal{T}\}$, where $o_j(t)$ is the observation of UAV j. The observation of UAV j only contains its current location coordinates, i.e., $o_j(t) = \mathbf{q}_j(t)$. Service preference $l_{ij}(t)$ of users and policies of other UAVs are all unknown to each UAV.
- *Action:* Set $A = \{a(t), t \in \mathcal{T}\}$, $a(t) = \{a_j(t) = (\theta_j(t), d_j(t)), \forall j \in \mathcal{M}\}$ represents the actions taken by all UAVs, where $\theta_j(t)$ and $d_j(t)$ represent the flying horizontal azimuth and the distance of UAV j during time slot t, which need to satisfy equations (17) and (18), respectively.
- *State transition probability:* $P : S \times A \times S \to [0, 1]$ represents the state transition probability distribution. Variable $\rho_0 : S \to [0, 1]$ is the distribution of initial state $s(0)$. The system state transits from $s(t)$ to $s(t + 1)$ by taking action $a(t)$ with probability $P(s(t + 1) \mid s(t), a(t))$.

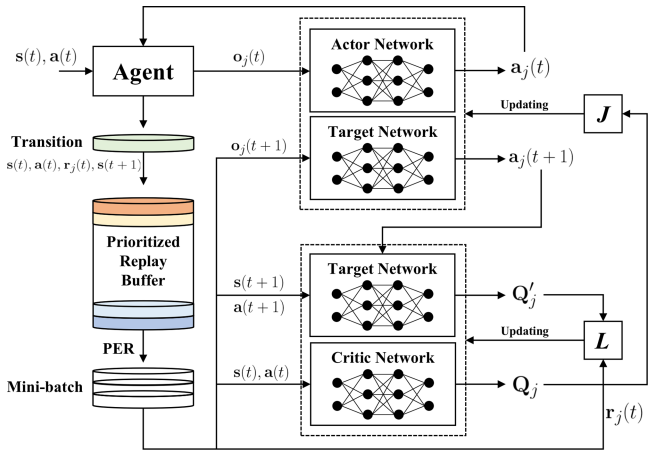


Fig. 3. Neural network structure of each UAV agent.

- *Reward:* $r_j(t) : S \times A \to \mathbb{R}$ indicates the reward that UAV j receives immediately after taking action $a_j(t)$ at state $s(t)$. We define the reward as $r_j(t) = -U_j(t) - p$, where p represents the penalty when constraints are not met. The goal of each UAV agent becomes to maximize its total expected reward $\bar{R}_j(t) = E[\sum_{\tau=0}^{t} \gamma^{\tau} r_j(\tau)]$, where variable $\gamma \in [0, 1]$ is the discount factor.

VI. Multi-UAV Trajectory Optimization

In this section, we propose a Multi-UAV Trajectory Optimization (MUTO) algorithm based on multi-agent DRL. The proposed MUTO-algorithm employs a framework of centralized training and decentralized execution, allowing agents to use additional information to simplify policy training, but not during execution. That is, when the UAV executes flying actions, only local observation is needed, namely a distributed solution of trajectory control. In addition, we adopt Prioritized Experience Replay (PER) technique [46], based on Temporal-Difference (TD), to design importance sampling weight of experience.

A. UAV Agent Design

Fig. 3 shows the network structure of a UAV as a DRL agent, which contains four neural networks, namely actor network $\pi_j(o_j(t))$ and its target network $\pi'_j(\cdot)$, critic network $Q_j(s(t), a(t))$ and its target network $Q'_j(\cdot)$. The actor network executes flying action $a_j(t)$ according to local observation $o_j(t)$ of UAV j, and it can be considered that the actor network contains policy π_j. The critic network calculates the corresponding Q value according to current network state $s(t)$ and action $a(t)$ of all UAVs. According to Deep Q Network (DQN) [47], Q value is defined as follows:

$$Q_j(s(t), a(t)) = E[\bar{R}_j(t) \mid s(t), a(t)]. \tag{29}$$

The interaction between agents and the environment can be organized as follows. First, UAV agent j gets partial observations $o_j(t)$ from the environment, and executes action $a_j(t)$ according to its current policy $\pi_j(o_j(t))$. Next, the environment updates state $s(t)$, and collects actions of all agents $\{a_j(j)\}_{j \in \mathcal{M}}$. Then, the corresponding reward $r_j(s(t), a(t))$ is returned to UAV agent $j \in \mathcal{M}$. Finally, UAV agent j can form transition $(s(t), a(t), r_j(t), s(t+1))$ (namely the experience), and store it in experience replay buffer $\mathcal{B}_j$. During training, a mini-batch of experiences is sampled from the replay buffer, and elements are sent to corresponding networks as the training basis.

In previous DRL algorithms, such as Q-learning [48], SARSA [49] and DDPG [50], the mini-batch uniformly samples experiences from the experience replay buffer. However, to increase the independence of each experience, the experience replay buffer is usually allocated with a large capacity, which leads to the small probability of the experience being selected through random sampling. When the number of experiences in the experience replay buffer grows rapidly, mini-batches obtained through random sampling may have indeterminate effects on learning a satisfied policy. This is because experiences in the mini-batch determine the loss value of the critic network. If the policy implied by the sampled experiences is very common or rare, the loss is very small, resulting in a smooth descent of the gradient, and the update of deep neural network weights becomes difficult [51]. Eventually, the training process slows down or even stops.

Therefore, we introduce the PER technique, referring to giving priority to each experience. Relatively important experiences are given a higher priority and have a higher probability of being sampled. UAVs can reduce their own computational costs by realizing better services to ground users through trajectory design. Specifically, multiple UAVs determine the service pattern of ground users through executing flying actions, and the trajectories of multiple UAVs affect both of the computational costs of ground users and that of UAVs. Thus, a satisfied policy for each UAV agent is hidden in some important experiences in an ambiguous manner, and they need to be sampled with high probabilities.

According to [46], the TD-error is used to design the priority of each experience. Based on our network model, the TD-error of UAV j at time slot t is defined as follows:

$$\delta_j(t) = \left| y_j(t) - Q_j(s(t), a_1(t), \ldots, a_N(t)) \right|, \tag{30}$$

where target value $y_j(t)$ is defined in equation (34), which is also introduced in the critic network in the following sub-section. Based on TD-error, a sampling priority value for each experience can be defined. We consider each mini-batch contains H experiences, and thus the sampling probability of the h-th experience can be obtained by:

$$p_j(h) = \frac{(\delta_j^h + \epsilon)^{\beta}}{\sum_{h'=1}^{H} (\delta_j^{h'} + \epsilon)^{\beta}}, \tag{31}$$

where $\epsilon = 0.001$ is a positive value to avoid not revisiting experiences whose TD-errors are 0. Variable β represents the factor to determine the priority, where $\beta = 0$ means uniform sampling, and usually $\beta = 0.6$ [18]. However, frequently sampling experiences with high TD-errors can lead to divergence and oscillations. In order to solve this problem, we introduce the importance sampling weight [52] to represent the importance of

Algorithm 1: MUTO-Algorithm.

Initialization: For UAV$j \in \mathcal{M}$
1. Randomly initialize actor network $\pi_j(o_j(0)\mid)$ with parameter $\{\theta_{\pi_j}\}$, critic network $Q_j(s(0), a(0) \mid \theta_{q_j})$ with parameter $\{\theta_{Q_j}\}$;
2. Randomly initialize corresponding target networks $\pi'_j(\cdot)$ and $Q'_j(\cdot)$ with parameters $\theta_{\pi'_j} = \theta_{\pi_j}$ and $\theta_{Q'_j} = \theta_{Q_j}$, respectively;
3. Initialize prioritized replay buffer $\mathcal{B}_j$;

for episode$= 1, \ldots, k_{\max}$ **do**
  Initialize environment and receive state $s(0)$;
  for time slot $t = 1, \ldots, T$ **do**
    for UAV $j \in \mathcal{M}$ **do**
      Formulate local observation $o_j(t)$, and execute flying action $a_j(t)$ according to current policy $\pi_j(o_j(t) \mid \theta_{\pi_j}) + \tilde{N}$;
      Collect action $a(t) = (a_j(t))\mid_{j\in\mathcal{M}}$ and reward $(r_j(t))\mid_{j\in\mathcal{M}}$, and transition to the new state $s(t+1)$;
      Constitute the experience $(s(t), a(t),_j (t), s(t+1))$, and store it in replay buffer $\mathcal{B}_j$ with its priority $(\delta_j(t) + \epsilon)$;
      if the number of experiences in replay buffer $\mathcal{B}_j$ is bigger than batch size **then**
        Sample H experiences from replay buffer $\mathcal{B}_j$ by probability $p_j(h)$ to form a mini-batch;
        Calculate the TD-error and importance sampling weight $w_j(h)$ of each experience;
        Update network parameters according to Algorithm 2;
      end
    end
  end
end

the sampled experience, which is defined as follows:

$$w_j(h) = \frac{1}{(|\mathcal{B}_j| \cdot p_j(h))^{\mu}}, \tag{32}$$

where $|\mathcal{B}_j|$ represents the size of the replay buffer. According to [46], μ is set to 0.4.

B. Muto-Algorithm

Based on experiences obtained by mini-batch sampling, each UAV agent starts to train its corresponding DRL model and updates the parameters of four neural networks. Since the designed MUTO-algorithm is based on the actor-critic framework, we first discuss how critic network performs network parameter update, and then that of actor and target networks.

As illustrated in Fig. 3, during the training process, critic network of each UAV agent obtains corresponding Q value $Q_j(s(t), a(t) \mid \theta_{Q_j})$ according to current state-action pair $(s(t), a(t))$, where θ_{Q_j} represents network parameters of the critic network. Meanwhile, the target network of actor network executes flying action $a_j(t+1)$ based on observation $o_j(t+1)$ and policy $\pi'_j(o_j(t+1) \mid \theta_{\pi_j})$, where θ_{π_j} represents parameters of the target network. Then, the target network obtains corresponding Q value $Q'_j(s(t+1), a(t+1) \mid \theta_{Q'_j})$ based on new state-action pair $(s(t+1), a(t+1))$, where $\theta_{Q'_j}$ represents network parameters of the target network. Finally, critic network is updated by minimizing loss function $L(\theta_{Q_j})$, which is defined as follows:

$$L(\theta_{Q_j}) = E[(y_j(t) - Q_j(s(t), a_1(t), \ldots a_N(t) \mid \theta_{Q_j}))^2], \tag{33}$$

$$y_j(t) = r_j(t) + \gamma Q'_j(s(t+1), a_1(t+1), \ldots, a_N(t+1) \mid \theta_{Q'_j}). \tag{34}$$

By mini-batch and PER techniques, loss function in equation (33) can be rewritten as:

$$L(\theta_{Q_j}) = \frac{1}{H}\sum_{h=1}^{H} w(h)(\delta_j^h)^2. \tag{35}$$

After completing the update of critic network, we utilize policy gradient to update parameters of actor network. The gradient is defined as follows:

$$\begin{aligned}\nabla_{\theta_{\pi_j}} J \approx \frac{1}{H}\sum_{h=1}^{H}[&\nabla_{\theta_{\pi_j}} \pi_j(o_j^h \mid \theta_{\pi_j}) \\ &\cdot \nabla_a Q_j(s^h, a_1^h, \ldots, a_N^h \mid \theta_{Q_j})\mid_{a_j^h = \pi_j(o_j^h)}].\end{aligned} \tag{36}$$

For the target network, its parameters are updated by defining update rate τ, and update equations are as follows:

$$\theta_{Q'_j} \leftarrow \tau\theta_{Q_j} + (1-\tau)\theta_{Q'_j}, \tag{37}$$

$$\theta_{\pi'_j} \leftarrow \tau\theta_{\pi_j} + (1-\tau)\theta_{\pi'_j}. \tag{38}$$

Algorithm 1 shows the process of MUTO-algorithm, and the specific process is as follows. First, we complete the initialization of three aspects: 1) randomly generate values for network parameters θ_{π_j} and θ_{Q_j} of actor network and critic network; 2) assign initial values to the corresponding target networks, i.e., $\theta_{\pi'_j} \leftarrow \theta_{\pi_j}$ and $\theta_{Q'_j} \leftarrow \theta_{Qj}$; 3) initialize the prioritized replay buffer $\mathcal{B}_j$ of each UAV agent, especially the priority of each experience.

Then, all UAV agents are trained with the upper limit of training episodes defined by $K_{\max}$. In each episode, the environment is initialized first, including take-off points of UAVs and users' location coordinates. At time slot $t \in \mathcal{T}$, each UAV agent first obtains its local observation $o_j(t) = \mathrm{q}_j$, and executes flying action $a_j(t)$ according to the current policy given by actor network, i.e., determining the flying horizontal azimuth and the flying distance.

After actions of all UAVs are executed, the environment updates the system state according to collected action set $a(t)$, moves to new state $s(t+1)$, sends corresponding rewards $(r_j(t))\mid_{j\in\mathcal{M}}$ to all UAVs, and constitutes experience $\{s(t), a(t), r_j(t), s(t+1)\}$. Then the UAV agent calculates the priority value of the experience, i.e., $\delta_j(t) + \epsilon$, and stores it in

Algorithm 2: Update of Neural Network Parameters.

for UAV agent $j \in \mathcal{M}$ **do**
- Calculate the target value according to equation (34);
- Update network parameter θ_{Q_j} of critic network Q_j, based on the minimization of loss function in equation (35);
- Update network parameter θ_{π_j} of actor network π_j, and conduct gradient descent based on the gradient in equation (36);
- Update parameters $\theta_{\pi'_j}$ and $\theta_{Q'_j}$ of target networks based on equations (37) and (38);

end

prioritized replay buffer $\mathcal{B}_j$ along with the experience. If the number of experiences in replay buffer $\mathcal{B}_j$ is bigger than the batch size, each UAV agent samples H experiences from its buffer based on PER technique. Based on a mini-batch of H experiences, each UAV agent updates its network parameters of four neural networks, and the corresponding process is shown in Algorithm 2.

The training is completed when rewards converge, and each UAV is able to execute flying actions at time slot t based merely on its local observation $o_j(t)$, without any other additional information.

Theorem 4: The overall complexity of the MUTO-algorithm for each UAV agent executing flying actions over the considered time T is $\mathcal{O}((\sum_{z=1}^{Z} n_z \cdot n_{z-1})T)$.

Please refer to Appendix E, available online for details.

C. Convergence Analysis

In this section, we analyze the convergence of the MUTO-algorithm. Inspired by previous convergence proof of multi-agent reinforcement learning [53], we utilize the function approximation of the Q-function to prove the convergence of our MUTO algorithm.

First, we introduce the concept of Nash Q-value [54]. According to Theorem 1 in Section V-A, from the perspective of Q-value, if the NE condition is satisfied, there is a set of Nash Q-value $\mathbf{Q}^*(t) = \{Q_1^*(t), \ldots, Q_M^*(t)\}$ for each time slot t (stage). To prove the convergence of $\mathbf{Q}(t) \triangleq \{Q_1(t), \ldots, Q_M(t)\}$ to Nash Q-value $\mathbf{Q}^*$, we approximate Q-function with a set of K linearly independent functions $\omega_j^k(s, a_j, \mathbf{a}_{-j}) : S \times A^M, j \in M, k \in K$, where $\mathbf{a}_{-j}$ represents the action set of agents other than agent j. Considering parameter vector $\psi_j \in \mathbb{R}^K, j \in M$, the approximation of Q-function can be defined by:

$$Q_j(s, a_j, \mathbf{a}_{-j}) = \sum_{k=1}^{K} \omega_j^k(s, a_j, \mathbf{a}_{-j})\psi_j(k) = \omega_j(s, a_j, \mathbf{a}_{-j})^\top \psi_j. \quad (39)$$

Based on such an approximation, we can derive the following Lemma.

Lemma 1: Q-function $Q_j(s, a_j, \mathbf{a}_{-j})$ is L-Lipchitz continuous w.r.t. action a_j, $\forall j \in M$, if Q-function Q_j is expressed as the linear span of a set of K linearly independent functions $\omega_j^k(s, a_j, \mathbf{a}_{-j}) : S \times A^M, j \in M, k \in K$, with parameter vector $\psi_j \in \mathbb{R}^K, j \in M$.

Lemma 1 illustrates that the variation of the Q-function with respect to action execution is limited, and the proof can be found in Appendix F, available online. Note that nonlinear activation functions can be leveraged to approximate real-world cases, but as long as the nonlinear scale is limited, Lipschitz continuous of Q-function still holds. According to the approximation in equation (39), we define the parameter update of ψ_j as follows:

$$\psi'_j = \psi_j + \alpha\Delta\nabla_{\psi_j} Q_j(s, a_j, \mathbf{a}_{-j}), \quad (40)$$

$$\Delta = r_j + \gamma Q_j(s', a_j, \mathbf{a}_{-j}) - Q_j(s, a_j, \mathbf{a}_{-j}), s, s' \in S, \quad (41)$$

where Δ is the temporal difference and r_j represents the reward function.

Therefore, proving the convergence of $\mathbf{Q}(t) \triangleq \{Q_1(t), \ldots, Q_M(t)\}$ to the Nash Q-value $\mathbf{Q}^*$ is equivalent to proving the convergence of the parameter update mechanism in equations (40) and (41) with the following assumptions:

Assumption 1: Each state-action pair is visited infinitely often, and the reward is bounded by constant L, and agent's policy is greedy in the limit with infinite exploration.

Assumption 2: For stage game $\{Q_j(t), j \in M\}$ at time slot t and in state s in training, the Nash equilibrium is recognized either as the global optimum or a saddle point.

Assumption 2 implies a strong constraint on every single stage game in training. However, as we analyzed in Theorem 1, the zone framed by the corresponding constraints is the destination of the optimal solution, and thus such an NE will fall within the constraints. Then, we introduce Lemma 2 inspired by [53] as follows:

Lemma 2: The random process $\{\Delta_t\}$ defined in $\mathbb{R}$ as

$$\Delta_{t+1}(x) = \Delta_t(x) + \alpha_t(x)(F_t(x) - \Delta_t(x))$$

converges to zero with probability 1 (w.p.1) when

1) $0 \le \alpha_t(x) \le 1, \sum_t \alpha_t(x) = \infty, \sum_t \alpha_t^2(x) < \infty$;
2) $x \in \mathcal{X}$, the set of possible states, and $|\mathcal{X}| < \infty$;
3) $\|\mathbb{E}[F_t(x) \mid \mathcal{F}_t]\|_W \le \gamma\|\Delta_t\|_W + c_t$, were $\gamma \in [0, 1]$ and c_t converges to zero w.p.1;
4) $\text{var}[F_t(x) \mid \mathcal{F}_t] \le L(1 + \|\Delta_t\|_W^2)$ with constant $L > 0$.

Herein, $\mathcal{F}_t$ denotes the filtration of an increasing sequence of $\sigma-$fields including the historical processes; variables α_t, Δ_t, $F_t \in \mathcal{F}_t$ and $\|\cdot\|_W$ is a weighted maximum norm [55].

Please refer to Appendix G, available online for the proof. Based on the above lemmas and assumptions, we finally derive the convergence of the parameter update mechanism in equations (40) and (41), thereby deriving the convergence of our proposed algorithm from Theorem 5 by adopting an ordinary differentiable equation with a globally asymptotically stable equilibrium point.

Theorem 5: Given the set of linearly independent functions $\omega_j^k, k \in K, j \in M$, and Q-function $Q_j(s, a_j, \mathbf{a}_{-j})$ is L-Lipchitz continuous w.r.t. action a_j, $\forall j \in M$, if Assumptions 1 and 2 hold, and the first two conditions of Lemma 2 can be met, the update mechanism in equations (40) and (41) converges w.p.1.

We demonstrate the convergence of the MUTO-algorithm with Theorem 5, and the proof can be found in Appendix H, available online.

VII. Performance Evaluation

In this section, we demonstrate the effectiveness of the proposed algorithm based on extensive simulations, including the simulation settings and numerical results.

A. Settings

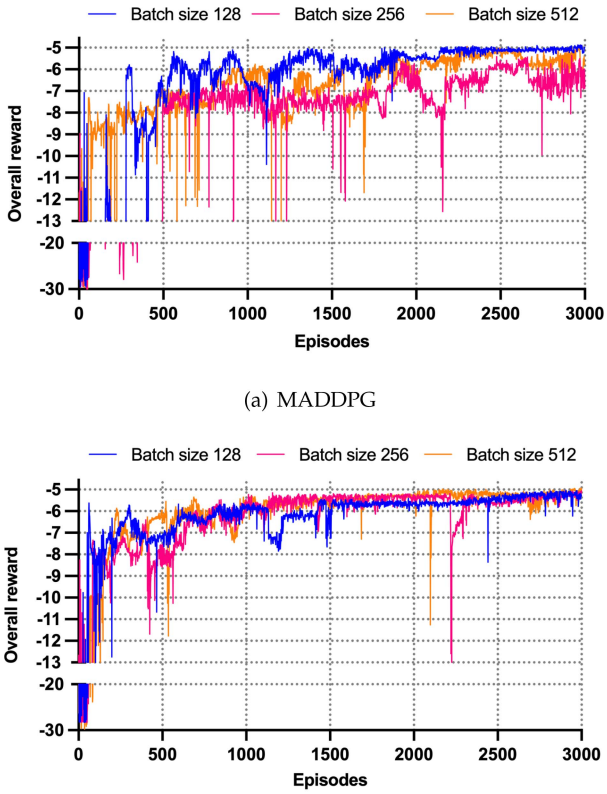
In this section, we conduct simulation experiments for the proposed MUTO-algorithm based on Python 3.9 and Pytorch 1.11.0. For the target area of UAV-assisted MEC network with differentiated services, we consider a square area with the side length of 400 m, i.e., $x_{\max} = y_{\max} = 400$ m. 100 users are randomly distributed over the area, and two UAVs belonging to different SPs act as edge servers. We consider the overall $T = 60$ time slots. The channel bandwidth $B_w = 10$ mHz, and the noise power $\sigma^2 = -90$ dBm. Each user generates computational tasks at each time. We utilize two different datasets, i.e., the electroencephalography (EEG) dataset [56] and YouTube video data [57] to evaluate the MUTO-algorithm.[1] The data size of each EEG task is between $[560, 747]$ KB according to the statistics,[2] and the size of YouTube video data belongs to $[30, 80]$ mB according to the statistics in [57].[3]

In addition, according to [3] and [18], for user i, the number of CPU cycles required for its computational task is $F_i(t) \in \{1000, 1200\}$ megacycles, and the upper limit of processing delay is $T_{\max} = 1$ s. The computing capability of the local device is $f_i^L = 1.0$ GHz, and the transmit power is $P_i(t) = 0.1$ W. For UAV j, its computing capability is $f_j^C = 12$ GHz, and hovering height $H_j(t)$ is 75 m. The maximum flying distance is $d^{\max} = 30$ m, and the data value threshold $\bar{D}_j$ is 800 MB. The horizontal coverage radius $R_j^{\max}$ is 75 m.

For neural networks, actor network and its target network of each UAV agent utilize three fully connected layers, and critic network and its target network utilize four fully connected layers. Both actor and critic networks apply the adaptive moment estimation optimizer for training. For the PER buffer, we set the capacity of replay buffer to 10^5. In addition, we set 4 different pairs of taking off locations to train the MUTO-algorithm, i.e., $\{(50, 200), (350, 200)\}$, $\{(10, 390), (390, 10)\}$, $\{(25, 25), (375, 375)\}$, and $\{(190, 190), (210, 210)\}$.

In order to analyze the performance of the proposed MUTO-algorithm, we compare it with the following four representative algorithms:

- Local computing (Local): Users process their tasks locally.
- Random Flying (RF): Similar to [58], based on the above parameter settings, each UAV randomly selects the flying horizontal azimuth and flying distance within the target area under various constraints.
- Community Flying (CF): Inspired by [13], we divide the target area into 16 communities. Users in each community share the community center as location coordinates. Each UAV chooses the community that can reduce its instantaneous computational cost as the flight destination, thereby achieving trajectory control.
- Multi-Agent Deep Deterministic Policy Gradient (MADDPG): According to [59], each UAV agent trains its policy directly based on the actor-critic algorithm and its local observations without PER technique.



(a) MADDPG

(b) MUTO

Fig. 4. Training performance of MUTO and MADDPG with different batch sizes.

B. Experimental Results

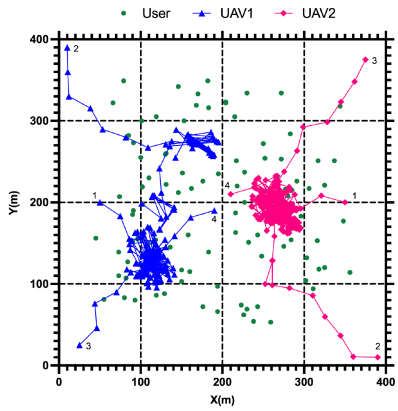
First, we analyze the training performance of our proposed MUTO-algorithm. Fig. 4 shows the training performance comparison of the MUTO-algorithm and MADDPG-algorithm under different batch sizes, i.e., the convergence performance of the UAV overall reward over the training episodes.

As illustrated in Fig. 4(a), the training performance of MADDPG-algorithm under different batch sizes is not stable. Specifically, the overall reward can only converge after 2100 training episodes when the batch size is 128. When the batch

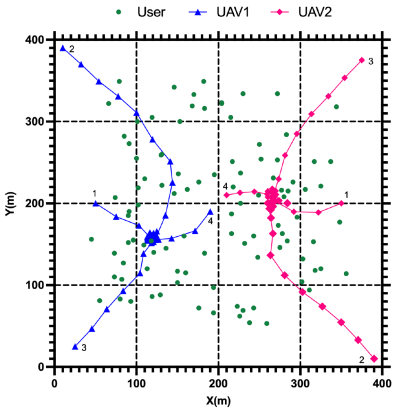
[1]These two datasets correspond to two typical applications of MEC networks, i.e., telemedicine and video services. Meanwhile, these two datasets can represent light and heavy computational intensities, respectively.

[2]https://www.ukbonn.de/en/epileptology/workgroups/lehnertz-workgroup-neurophysics/downloads/

[3]According to [57], the data size of about 90% of the YouTube video files is less than 30 MB and mostly less than 80 MB. We focus on the computational-intensive case, i.e., the data size of the YouTube video files varies between [30, 80] MB, to verify the robustness of our algorithm.



(a) CF



(b) MUTO

Fig. 5. Trajectories controlled by MUTO and CF with different taking-off locations.

size increases to 256 and 512, although the overall reward of UAVs increases with the training episodes, it does not show a clear convergence trend within 3000 episodes; on the contrary, it fluctuates greatly. This is because the reward generated at each training episode includes not only the computational cost, but also huge penalties for not meeting the constraints, such as moving outside borders, and processing computational task timeout.

In contrast, as shown in Fig. 4(b), MUTO-algorithm can quickly converge towards the optimal solution in the early episodes of training due to the utilization of PER technique. During the entire training process, the overall reward of UAVs shows a small fluctuation. In addition, the convergence performance is stable and similar under different batch sizes, i.e., after 1500 training episodes, MUTO-algorithm converges to the optimal solution with the maximum overall reward. To sum up, our proposed MUTO-algorithm has both efficiency and robustness for not only rapidly and stably converging towards the optimal solution over training episodes. In addition, the batch size has limited effects on the convergence speed and result of training our proposed MUTO-algorithm.

Fig. 5 shows the flying trajectories of two UAVs controlled by MUTO-algorithm and CF-algorithm under four pairs of different taking-off locations. As illustrated in Fig. 5(a), UAVs controlled by CF-algorithm are able to find the approximate optimal locations, i.e., the trajectory-aggregated areas in Fig 5(a). Specifically, each UAV finds its own optimal area and repeatedly searches for the optimal location in this area under the control of CF-algorithm. However, due to the short-sighted essential of CF-algorithm, each UAV determines the flying destination merely based on the instantaneous computational cost other than the long-term one. In addition, the exploration space of each UAV is insufficient, since the location of each community is fixed, and the number of communities is limited. This is obviously not as exploratory as free space exploration, which allows each UAV to explore potential locations. Although UAVs controlled by CF-algorithm are able to find the approximate optimal areas, they fly repeatedly around the optimal locations other than hovering over them stably.

Fig. 5(b) shows the flying trajectories of two UAVs controlled by MUTO-algorithm. Specifically, each UAV flies toward its optimal location in a relatively clear trajectory under the control of MUTO-algorithm, regardless of the taking-off location. This shows that our proposed MUTO-algorithm can grasp the long-term characteristics of time-varying user preferences and computational tasks under different taking-off conditions, so as to control each UAV to fly to its optimal location directly. It provides ground users with reliable differentiated services, and thus reduces the overall computational costs of users and UAVs.

Based on EEG data, Fig. 6 compares the performance of different flying control algorithms in terms of two metrics, i.e., the overall cost of users and UAVs. Specifically, Fig. 6(a) shows the variation of computational costs for all users over time slots. RF-algorithm is obviously not stable and computationally costly. MUTO-algorithm greatly reduces the overall computational cost of users. Compared with local computing, the overall cost of users is reduced by more than 50%, and the value of computational cost remains stable after 5 time slots, which implies that UAVs have rapidly reached their optimal locations. Although the CF-algorithm can also quickly reduce and stabilize the overall cost, the fluctuation of the cost value is larger than that of MUTO-algorithm. This is consistent with the trajectory shown in Fig. 5, where MUTO-algorithm guides UAVs to the optimal location within the first ten time slots, while CF-algorithm can only guide them to their optimal areas and control them to hover around the vicinity of the optimal location. Thus, the overall cost caused by CF-algorithm is not as stable as that caused by MUTO, and the cost value of CF-algorithm is 1.4 times greater than that of MUTO.

Fig. 6(b) compares the performance of four algorithms under different taking-off locations, where the ordinate value of each bar is the average value with the standard deviation of users' overall computational cost over 60 time slots. MUTO is the least affected by taking off locations, whose overall cost of users is always around 5, and its standard deviation after stabilization is the smallest, indicating that each UAV is stably hovering over the optimal location. Compared with MUTO, CF-algorithm not only has a greater overall cost but also has less stability, i.e., the overall cost shows a relatively large fluctuation under different taking-off locations. In addition, the standard deviation is greater than

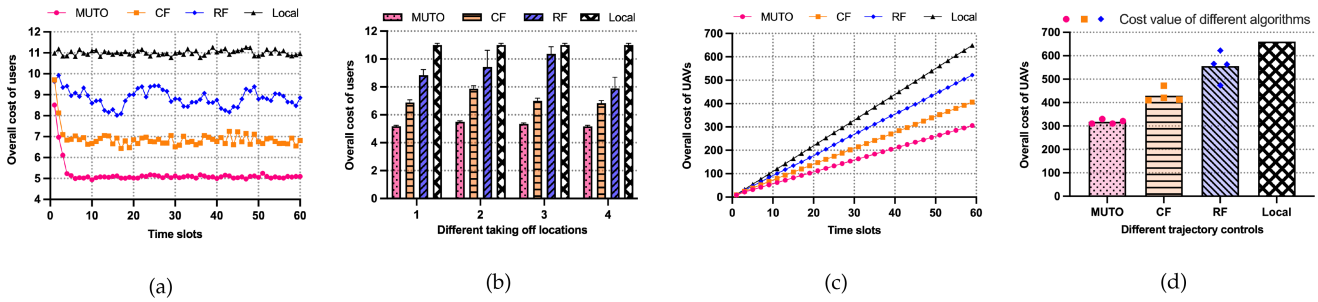


Fig. 6. Performance comparison of different flying control algorithms based on EEG data.

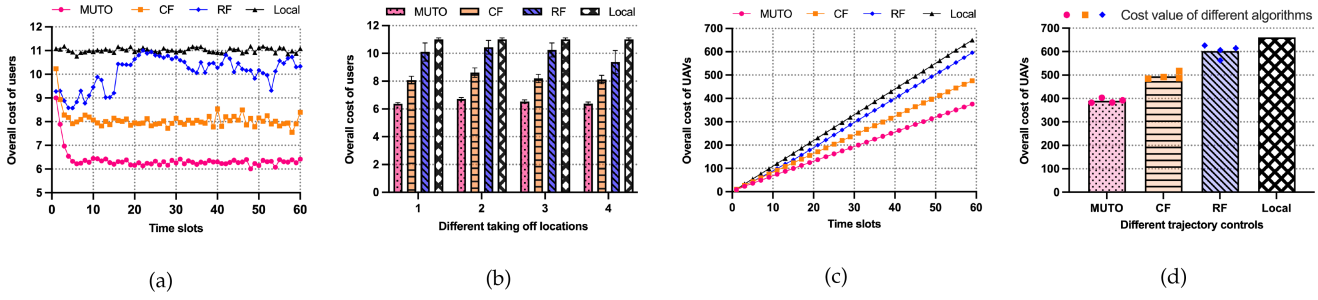


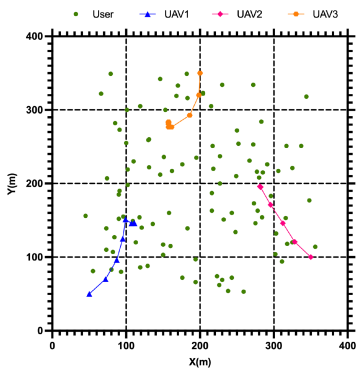
Fig. 7. Performance comparison of different flying control algorithms based on YouTube data.

that of MUTO, which is consistent with the results of Fig. 6(a). Due to randomness, the variation trend of the computational cost under different taking-off locations in RF-algorithm is not consistent with those of MUTO and CF algorithms, and the standard deviation of RF is the largest.

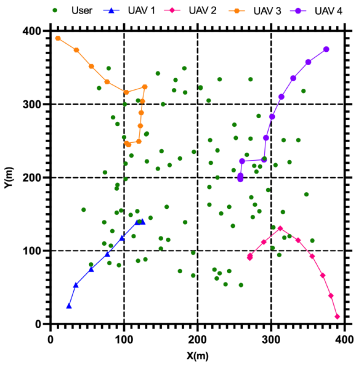
Similarly, Fig. 6(c) and (d) show the overall cost of UAVs over time and under different taking-off locations, respectively. Fig. 6(c) shows that the overall cost of MUTO-algorithm is always the smallest compared with those of other flying control methods. Specifically, the overall cost of MUTO over the entire time is about 310, which is 25%, 41%, and 53% lower than that of CF, RF, and Local algorithms, respectively. Fig. 6(d) shows the scatter plot and the mean value of the overall cost of UAVs in different flying control algorithms under four taking-off locations, where four locations correspond to the four points on each column, and three colors and point shapes correspond to three trajectory control algorithms, i.e., MUTO, CF, and RF. MUTO-algorithm can also obtain the lowest overall cost, and results of different taking-off locations are concentrated around the mean value, which implies the robustness of MUTO. The average overall costs obtained by CF, RF, and Local algorithms increase sequentially, and the scatter plots become more and more dispersed. In addition, from Fig. 6(a), (c), (b), and (d), we can observe that the curves of overall costs of users and UAVs under different algorithms have similar trends, which confirm Theorem 3.

As shown in Fig. 7, the overall cost of UAVs are evaluated based on YouTube video data. Although all flying control algorithms have greater overall computational costs than those based on EEG data, MUTO-algorithm still has the optimal performance. Specifically, Fig. 7(a) shows that MUTO-algorithm reduces the overall cost of users by more than 40% compared to local computing. The cost fluctuation of MUTO after stabilization is slightly larger than that based on EEG data. However, this is mainly caused by the largely increased data size of computational tasks, rather than the unstable flying control of MUTO. This can be verified in Fig. 7(b), where the overall costs of users based on MUTO-algorithm under different taking-off locations are around 6.5, and the standard deviation is slightly greater than that based on EEG data. It means that MUTO-algorithm is still robust against different initial conditions, i.e., taking off locations, but the variation range of the data size is greatly increased, which results in a larger fluctuation of the computational cost over time.

Fig. 7(c) and (d), and Fig. 6(c) and (d) show similar performance of the overall cumulative computational cost of UAVs. Although the costs are greater than that based on EEG data, MUTO-algorithm can still reduce the overall cumulative computational cost of UAVs by 21%, 37% and 42% compared to CF, RF, and Local algorithms, respectively, as shown in Fig. 7(c). Meanwhile, the performance of MUTO-algorithm under different taking-off locations is still the best. As illustrated in Fig. 7(d), MUTO-algorithm not only brings the lowest mean value of the overall cost but also ensures its stability, i.e., the scatter plot of MUTO is the most concentrated. In summary, the simulation results based on EEG data and YouTube video data demonstrate the advantages of MUTO-algorithm. That is, it brings the least computational cost, robust against the taking off locations, and can adapt to different types of data. In addition, there is no need to train on more taking-off locations. It is reasonable, since the taking-off locations of UAVs are relatively fixed in real-world scenarios.

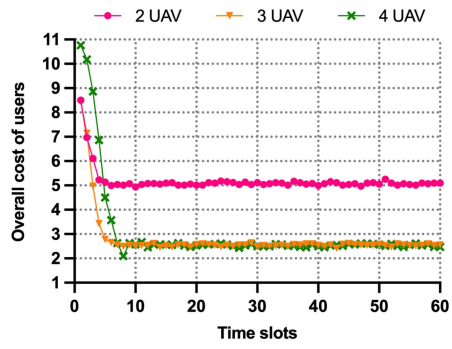


(a) 3 SPs

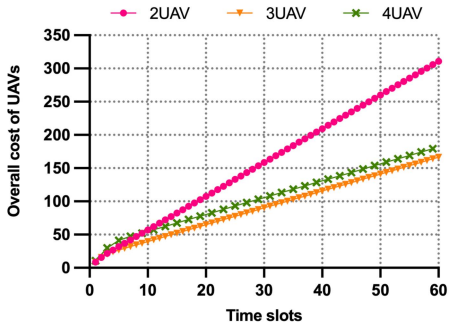


(b) 4 SPs

Fig. 8. Trajectories of UAVs with expanded SPs based on MUTO algorithm.



(a)



(b)

Fig. 9. Performance Comparison with expanded SPs based on MUTO algorithm.

We also expand the number of SPs in the target area, namely, the number of UAVs, and analyze their flying trajectories and computational costs. Fig. 8 shows cases with 3 and 4 SPs, each of which controls its UAV based on MUTO-algorithm and executes flying actions with merely a local observation, i.e., its current location coordinates. Specifically, for the case with 3 SPs in Fig. 8(a), taking off locations are (50, 50), (350, 100) and (200, 350). For the case with 4 SPs in Fig. 8(b), taking off locations are (25, 25), (390, 10), (10, 390) and (375, 375). It can be discovered from Fig. 8(a) and (b), MUTO-algorithm is able to guide each UAV to its optimal location rapidly and stably in both cases. This demonstrates that MUTO-algorithm has strong scalability for the scenarios with more SPs.

Fig. 9 compares overall costs of users and UAVs under different numbers of SPs. Compared to the case with two UAVs, deploying more UAVs can further reduce the computational cost of all users and the cumulative computational cost of UAVs. However, increasing the number of UAVs cannot further reduce the overall computational costs of users and UAVs, since 3 UAVs almost cover all ground users. In addition, as shown in Fig. 9(a), the overall cost of users with 4 SPs decreases slower than that with 3 SPs. Such an initial slow decrease also causes the increase in the overall cost of UAVs controlled by 4 SPs, as shown in Fig. 9(b). This is mainly caused by the difference in taking-off locations, rather than the defect of MUTO flying control. It can be confirmed from Fig. 8, where the case with 4 SPs takes more time for MUTO-algorithm to guide UAVs to their optimal locations, since their taking-off locations are far away compared to the case with 3 SPs.

VIII. Conclusion

Considering differentiated services provided by UAVs from different SPs and time-varying service preferences of users, we first analyzed the interaction among SPs, derived NE conditions with the uniqueness theorem, and proved that solutions to the two proposed minimization problems can be reached simultaneously with complete system information. Then we proposed a multi-UAV trajectory optimization algorithm based on DRL, which is based on partial information, and it can execute flying actions distributively. Specifically, each SP is modeled as an agent, and we trained neural networks of SPs with the PER technique. Finally, the flying action execution of multiple UAVs based on local observations is realized, minimizing the computational cost of users and the long-term one of UAVs.

Although our work addressed the multi-SP UAV trajectory optimization with differentiated services, there are still some challenges in UAV-assisted MEC networks. For example, how to build cooperative computation offloading among different edge computing devices, and how to realize 3D trajectory optimization of UAVs. In addition to integrating aerial facilities such as UAVs, how to integrate low-earth orbit satellites into the aerial-ground computing architecture and achieve ubiquitous Internet access has become a new hotspot for building future MEC, which will be investigated in our future work.

REFERENCES

[1] Z. Ning, X. Kong, F. Xia, W. Hou, and X. Wang, "Green and sustainable cloud of things: Enabling collaborative edge computing," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 72–78, Jan. 2019.

[2] M. Othman, S. A. Madani, S. U. Khan, and A. U. R. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tut.*, vol. 16, no. 1, pp. 393–413, First Quarter 2014.

[3] Z. Ning et al., "Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 2628–2644, May 2023, doi: 10.1109/TMC.2021.3129785.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tut.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.

[5] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-Based IoT platform: A crowd surveillance use case," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 128–134, Feb. 2017.

[6] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM*, 2013, pp. 1285–1293.

[7] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 160–168, 2019.

[8] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tut.*, vol. 21, no. 3, pp. 2334–2360, Third Quarter 2019.

[9] 5G PPP Architecture Working Group, "View on 5G architecture, version 4.0," 2021. [Online]. Available: https://zenodo.org/record/5155657

[10] Huawei News, "Huawei and dronetech elevate partnership to facilitate sustainable farming in austria," 2022. [Online]. Available: https://www.huawei.com/en/news/2022/10/5g-smart-farming-austria-dronetech

[11] J. O'Halloran, "Nokia claims first ce-certified 5G automated drone-in-a-box service," 2023. [Online]. Available: https://www.computerweekly.com/news/366536372/Nokia-claims-first-CE-certified-5G-automated-drone-in-a-box-service

[12] S. Evans, "Qualcomm's 5G drone showcased at hannover messe 2023," 2023. [Online]. Available: https://www.iotworldtoday.com/connectivity/qualcomm-s-5g-drone-showcased-at-hannover-messe-2023

[13] Z. Ning et al., "5G-enabled UAV-to-community offloading: Joint trajectory design and task scheduling," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3306–3320, Nov. 2021.

[14] X. Zhang, J. Zhang, J. Xiong, L. Zhou, and J. Wei, "Energy-efficient multi-UAV-enabled multiaccess edge computing incorporating NOMA," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5613–5627, Jun. 2020.

[15] B. Dai, J. Niu, T. Ren, Z. Hu, and M. Atiquzzaman, "Towards energy-efficient scheduling of UAV and base station hybrid enabled mobile edge computing," *IEEE Trans. Veh. Technol*, vol. 71, no. 1, pp. 915–930, Jan. 2022.

[16] H. Mei, K. Yang, Q. Liu, and K. Wang, "Joint trajectory-resource optimization in UAV-enabled edge-cloud system with virtualized mobile clone," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5906–5921, Jul. 2020.

[17] J. Zhang et al., "Computation-efficient offloading and trajectory scheduling for multi-UAV assisted mobile edge computing," *IEEE Trans. Veh. Technol*, vol. 69, no. 2, pp. 2114–2125, Feb. 2020.

[18] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and A. Nallanathan, "Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3536–3550, Oct. 2022.

[19] T. Ren et al., "Enabling efficient scheduling in large-scale UAV-assisted mobile-edge computing via hierarchical reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7095–7109, May 2022.

[20] X. Hu, K.-K. Wong, and Y. Zhang, "Wireless-powered edge computing with cooperative UAV: Task, time scheduling and trajectory design," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 8083–8098, Dec. 2020.

[21] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.

[22] C. Zhan, H. Hu, Z. Liu, Z. Wang, and S. Mao, "Multi-UAV-enabled mobile-edge computing for time-constrained IoT applications," *IEEE Internet Things J.*, vol. 8, no. 20, pp. 15 553–15 567, Oct. 2021.

[23] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1405–1418, Jun. 2020.

[24] Z. Ning et al., "Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 4, pp. 1319–1333, Apr. 2022.

[25] X. Wang, Z. Ning, L. Guo, S. Guo, X. Gao, and G. Wang, "Mean-field learning for edge computing in mobile blockchain networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5978–5994, Oct. 2023, doi: 10.1109/TMC.2022.3186699.

[26] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2745–2762, Sep. 2021.

[27] T. M. Ho and K.-K. Nguyen, "Joint server selection, cooperative offloading and handover in multi-access edge computing wireless network: A deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 21, no. 7, pp. 2421–2435, Jul. 2022, doi: 10.1109/TMC.2020.3043736.

[28] Y. Hu, J. Wang, B. Wu, and S. Helal, "RL-recruiter+: Mobility-predictability-aware participant selection learning for from-scratch mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4555–4568, Dec. 2022.

[29] Z. Ning et al., "Blockchain-enabled intelligent transportation systems: A distributed crowdsensing framework," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4201–4217, Dec. 2022.

[30] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3147–3159, Apr. 2020.

[31] M. Liwang, Z. Gao, and X. Wang, "Let's trade in the future! A futures-enabled fast resource trading mechanism in edge computing-assisted UAV networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3252–3270, Nov. 2021.

[32] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in UAV-enabled mobile edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4576–4589, Sep. 2019.

[33] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 175–190, Jan. 2023, doi: 10.1109/TMC.2021.3069911.

[34] F. Cheng et al., "UAV trajectory optimization for data offloading at the edge of multiple cells," *IEEE Trans. Veh. Technol*, vol. 67, no. 7, pp. 6732–6736, Jul. 2018.

[35] Z. Ning, H. Chen, X. Wang, S. Wang, and L. Guo, "Blockchain-enabled electrical fault inspection and secure transmission in 5G smart grids," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 1, pp. 82–96, Jan. 2022.

[36] T. Kimura and M. Ogura, "Distributed collaborative 3D-deployment of UAV base stations for on-demand coverage," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1748–1757.

[37] X. Wang, Z. Ning, S. Guo, M. Wen, L. Guo, and V. Poor, "Dynamic UAV deployment for differentiated services: A multi-agent imitation learning based approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2131–2146, Apr. 2023.

[38] Quaternium, Hybrix long-endurance drone: The leader of hybrid multirotors, 2019. [Online]. Available: https://www.quaternium.com/uav/hybrix-drone

[39] K. Wang, X. Zhang, L. Duan, and J. Tie, "Multi-UAV cooperative trajectory for servicing dynamic demands and charging battery," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1599–1614, Mar. 2023.

[40] X. Li, H. Yao, J. Wang, C. Jiang, and F. R. Yu, "An energy-efficient UAV recharging and reshuffling strategy for seamless coverage," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[41] Z. Wei et al., "Sum-rate maximization for IRS-assisted UAV OFDMA communication systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2530–2550, Apr. 2021.

[42] Q. Wu and R. Zhang, "Common throughput maximization in UAV-enabled OFDMA systems with delay consideration," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6614–6627, Dec. 2018.

[43] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898–6908, Aug. 2020.

[44] J. Ji, K. Zhu, C. Yi, and D. Niyato, "Energy consumption minimization in UAV-assisted mobile-edge computing systems: Joint resource allocation and trajectory design," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8570–8584, May 2021.

[45] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6252–6265, Jul. 2020.
[46] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.
[47] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[48] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.
[49] J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work?–A literature review of empirical studies on gamification," in *Proc. IEEE 47th Hawaii Int. Conf. Syst. Sci.*, 2014, pp. 3025–3034.
[50] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[51] C. H. Liu, Z. Dai, Y. Zhao, J. Crowcroft, D. Wu, and K. K. Leung, "Distributed and energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 20, no. 1, pp. 130–146, Jan. 2021.
[52] A. R. Mahmood, H. P. Van Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3014–3022.
[53] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 5571–5580.
[54] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, no. Nov, pp. 1039–1069, 2003.
[55] D. P. Bertsekas, "Weighted sup-norm contractions in dynamic programming: A review and some new applications," Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. LIDS-P-2884, 2012, Art. no. 5.
[56] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state," *Phys. Rev. E*, vol. 64, no. 6, 2001, Art. no. 061907.
[57] X. Che, B. Ip, and L. Lin, "A survey of current youtube video characteristics," *IEEE MultiMedia*, vol. 22, no. 2, pp. 56–63, Second Quarter 2015.
[58] X. Yuan et al., "Capacity analysis of UAV communications: Cases of random trajectories," *IEEE Trans. Veh. Technol*, vol. 67, no. 8, pp. 7564–7576, Aug. 2018.
[59] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

Zhaolong Ning (Senior Member, IEEE) received the PhD degree from Northeastern University, Shenyang, China in 2014. He was a research fellow with Kyushu University from 2013 to 2014, Japan. Currently, he is a full professor with the College of Communication and Information Engineering, the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include mobile edge computing, 6G networks, machine learning, and resource management. He has published more than 150 scientific papers in international journals and conferences. He serves as an associate editor or guest editor of several journals, such as *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Social Computational Systems*, *The Computer Journal* and so on. He is a Highly Cited researcher (Web of Science) since 2020.

Yuxuan Yang received the MSc degree in software engineering from Dalian University of Technology, Dalian, China in 2022. He is currently working towards the PhD degree with the School of Electrical and Information Engineering, The University of Sydney, Sydney, Australia. His research interests include mobile edge computing, ubiquitous mobile networking, and resource allocation.

Xiaojie Wang (Senior Member, IEEE) received the PhD degree from Dalian University of Technology, Dalian, China, in 2019. After that, she was a postdoctor with the Hong Kong Polytechnic University, Hung Hom, Hong Kong. Currently, she is a full professor with the College of Communication and Information Engineering, the Chongqing University of Posts and Telecommunications, Chongqing, China. Her research interests are wireless networks, mobile edge computing and machine learning. She has published more than 70 scientific papers in international journals and conferences, such as *IEEE Transactions on Mobile Computing*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Parallel and Distributed Systems* and *IEEE Communications Surveys and Tutorials*.

Qingyang Song (Senior Member, IEEE) received the PhD degree in telecommunications engineering from the University of Sydney, Sydney, Australia, in 2007. From 2007 to 2018, she was with Northeastern University, Shenyang, China. She joined Chongqing University of Posts and Telecommunications in 2018, where she is currently a professor. She has authored more than 100 papers in major journals and international conferences. Her current research interests include cooperative resource management, edge computing, vehicular ad hoc network, mobile caching, and simultaneous wireless information and power transfer. She serves on the editorial boards of two journals, including area editor for *IEEE Transactions on Vehicular Technology* and technical editor for *Digital Communications and Networks*.

Lei Guo received the PhD degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He is currently a full professor with Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored or coauthored more than 200 technical papers in international journals and conferences. He is an editor for several international journals. His current research interests include communication networks, optical communications, and wireless communications.

Abbas Jamalipour (Fellow, IEEE) received the PhD degree in electrical engineering from Nagoya University, Nagoya, Japan in 1996. He holds the positions of professor of Ubiquitous Mobile Networking with the University of Sydney and since January 2022, the editor-in-chief of the *IEEE Transactions on Vehicular Technology*. He has authored nine technical books, eleven book chapters, more than 550 technical papers, and five patents, all in the area of wireless communications and networking. He is a recipient of the number of prestigious awards, such as the 2019 IEEE ComSoc Distinguished Technical Achievement Award in Green Communications, the 2016 IEEE ComSoc Distinguished Technical Achievement Award in Communications Switching and Routing, the 2010 IEEE ComSoc Harold Sobol Award, the 2006 IEEE ComSoc Best Tutorial Paper Award, as well as over 15 Best Paper Awards. He was the president of the IEEE Vehicular Technology Society (2020–2021). Previously, he held the positions of the executive vice-president and the editor-in-chief of VTS Mobile World and has been an elected member of the Board of Governors of the IEEE Vehicular Technology Society since 2014. He was an editor-in-chief *IEEE Wireless Communications*, the vice president-Conferences, and a member of Board of Governors of the IEEE Communications Society. He sits on the Editorial Board of the IEEE ACCESS and several other journals and is a member of Advisory Board of *IEEE Internet of Things Journal*. He has been the general chair or technical program chair for several prestigious conferences, including IEEE ICC, GLOBECOM, WCNC, and PIMRC. He is a fellow of the Institute of Electrical, Information, and Communication Engineers (IEICE), and the Institution of Engineers Australia, an ACM professional member, and an IEEE distinguished speaker.