Delay and Load Fairness Optimization with Queuing Model in Multi-UAV Assisted MEC: A Deep Reinforcement Learning Approach

Qiang Tang, Bao Li, Halvin H. Yang, Yan Li, Shiming He, Kun Yang, *Fellow, IEEE*

***Abstract*—Unmanned aerial vehicle (UAV) can alleviate the computational burden on edge devices through assisted computing. However, with the increase in the number of Internet of Things Devices (IoTDs), it is essential to establish a task queue on the UAV to schedule computing tasks from IoTDs. In addition, the load fairness of UAVs should be optimized to fully utilize the computing resources. Therefore, a multi-UAV-assisted mobile edge computing (MEC) network framework based on the queuing model is proposed, which aims at optimizing the average delay of all user devices and the load fairness of UAVs. Firstly, we prove that the arrangement of tasks with different computing delays on the UAV queue can affect the user's average delay, so a short-job-first (SJF) queuing model is proposed to minimize the average delay of users. On this basis, a joint optimization problem related to the UAV's three-dimensional trajectory and user connection scheduling is formulated. A SJF based low-complexity connection scheduling algorithm is proposed and combined in a deep reinforcement learning (DRL) to solve this NP-hard problem. To evaluate the performance of the proposed algorithm, we compare it with deep deterministic policy gradient (DDPG), particle swarm optimization (PSO), random moving (RM), and local computing (LC). Simulation results show that our algorithm effectively reduces user average delay and enhances UAV load fairness. Finally, SJF is compared with the traditional first-come-first-served (FCFS) queuing model on different algorithms. The results indicate that the average delay of SJF is significantly lower than that of FCFS.**

***Index Terms*—Multi-UAV, Mobile edge computing (MEC), Internet of Things devices (IoTDs), Short-job-first (SJF), Deep reinforcement learning (DRL).**

I. INTRODUCTION

The rapid increase in the number of Internet of Things devices (IoTD) brings challenges to current communication networks [1]. At the same time, with the development of Artifical Intelligence (AI), Internet of Things (IoT) and Virtual Reality (VR), the requirements for network bandwidth and communication latency are further increased [2]. In the cloud computing model, the communication delay between the cloud and users is high and can't meet the real-time requirements [3]. Fortunately, MEC can solve this problem by sinking computing resources from the cloud to the edge network [4] [5]. Especially with the evolution of UAV technology in recent years, the UAV-assisted MEC communication architecture has become a research hotspot in the Internet of Things, mainly due to its mobility, line-of-sight (LoS) communication channel, ability to carry certain computing resources and flexible deployment characteristics [6] [7].

At present, there are still some challenges in deploying multi-UAV-assisted mobile edge computing in large-scale user scenarios [8]. It includes issues such as collaboration between multi-UAVs, three-dimensional trajectory planning, and scheduling of concurrent user tasks. Therefore, this paper will study these problems.

The main contributions of this paper are summarized as follows.

- In order to cope with the high concurrency of user tasks and effectively reduce the average delay, we propose a short-job-first (SJF) queuing model to manage large-scale user tasks. It has been proven that the average delay of SJF queuing model is the lowest.
- Based on the SJF queuing model, we formulate a joint optimization problem related to UAV three-dimensional trajectory and user connection scheduling to reduce the average user delay and enhance UAV load fairness. To handle user tasks sequentially, the UAV's working time slot is dynamically divided into two sub-slots, which are used to receive user tasks and compute tasks respectively.
- A DRL algorithm is proposed to solve the NP-hard problem of large-scale user scenarios in multi-UAV networks. To increase the agent's exploration for obtaining better solutions, we adopt soft actor critic (SAC) algorithm to optimize the UAV trajectory instead of the traditional deep deterministic policy gradient (DDPG) algorithm. According to the characteristics of SJF, a low-complexity connection scheduling algorithm is further designed.
- Finally, the performance of the proposed algorithm is verified through numerical simulation and compared with some other advanced algorithms. Experimental results show that the algorithm proposed in this paper has better performance in optimizing user average delay and UAV load fairness.

The rest of the paper is organized as follows. Section II introduces the related works. Section III describes the construction of the system model. In Section IV, we propose

Qiang Tang, Bao Li, Yan Li and Shiming He are with the School of Computer Science and Communication Engineering, Changsha University of Science and Technology, Changsha, Hunan, 410114, China. (emails: tangqiang@csust.edu.cn; 22208051673@stu.csust.edu.cn; liyan@csust.edu.cn; smh_cs@csust.edu.cn)

Halvin H. Yang is with the Department of Electronic and Electrical Engineering, University College London, WC1E 7JE London, U.K. (email: uceehhy@ucl.ac.uk)

Kun Yang is with the School of Computer Science and Electronic Engineering, University of Essex, Essex CO4 3SQ, U.K. (email: kunyang@essex.ac.uk)

Corresponding author: Halvin H. Yang.

a low-complexity scheduling algorithm and combine it with the DRL algorithm to solve the formulated problem. The simulation results of the proposed algorithm are given in Section V. Finally, Section VI summarizes the paper and gives prospects for future work.

II. Related Works

In recent years, some scholars have conducted research on UAV-assisted MEC [9]. Pang *et al.* [10] studied the application of non-orthogonal multiple access (NOMA) in UAV network. In order to reduce energy consumption in UAV networks, Song *et al.* [9] jointly optimized UAV trajectory, user connection scheduling, and resource allocation to minimize system energy consumption through block coordinate descent (BCD) and successive convex approximation (SCA). Ning *et al.* [11] proposed a UAV-assisted computing framework based on 5G network. The authors optimized UAV trajectory and user task offloading through auction algorithm and dynamic programming algorithm to maximize network throughput. For the purpose of reducing communication interference in UAV network, Pang *et al.* [12] proposed an interference mitigation scheme based on intelligent reflecting surfaces (IRSs) and provided important design insights for IRS-aided UAV communication in cellular network. Yuan *et al.* [13] considered using UAV and ground sensor nodes (SNs) to build a wireless power transmission (WPT) network, and optimized flight trajectory of UAV to maximize energy collection of ground SNs. To manage large-scale user tasks, Chen *et al.* [14] introduced a first-come-first-served (FCFS) queue into UAV network to adapt user task concurrency scenarios, and optimized user task offloading through game theory to minimize latency.

Above research mainly focuses on the use of single UAV to assist communication. However, in large-scale user scenarios, we need to consider a communication network composed of multiple UAVs. Since it is very difficult to solve NP-hard problems in multi-UAV networks through heuristics and traditional convex optimization methods, DRL is proposed by some scholars to address these issues [15] [16].

Koushik *et al.* [17] determined the optimal link selection between UAVs based on a deep Q-network (DQN) to improve throughput efficiency of network. Since DQN has an overestimation problem in some scenarios, Li *et al.* [18] proposed using Double DQN (DDQN) to optimize UAV trajectory and device connection scheduling to minimize system energy consumption. Unfortunately, DQN and DDQN are not applicable when solving problems in continuous spaces due to the high dimensionality. In order to address the optimization problem in continuous space, Seid *et al.* [19] proposed using the deep deterministic policy gradient (DDPG) algorithm to optimize resource allocation of UAV clusters, connection scheduling, and transmission power of IoTDs, thereby minimizing delay and energy consumption in the UAV network. Liu *et al.* [20] proposed an improved DRL algorithm based on DDPG to optimize the communication coverage of UAVs, aiming to improve the fairness of user communication and reduce UAVs' energy consumption. To enhance the solving effect of single-agent algorithms in high-dimensional space, the multi-agent deep deterministic policy gradient (MADDPG) algorithm is proposed in [21] to optimize resource allocation and computational offloading of UAV swarms.

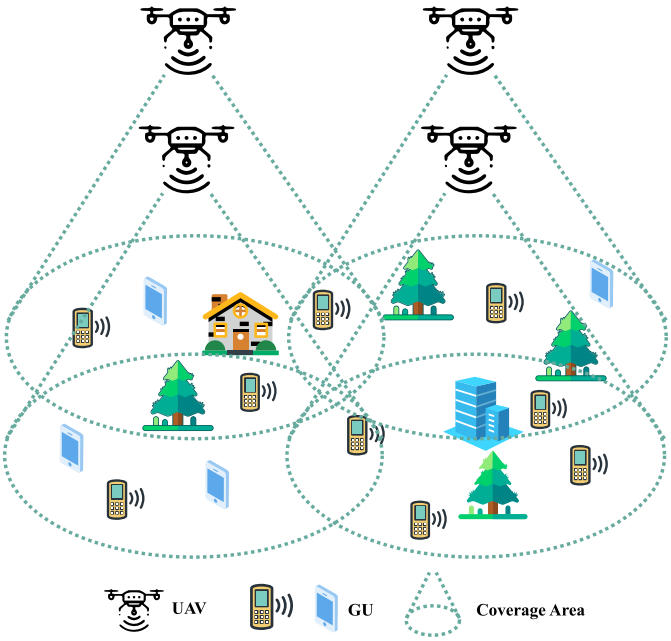


Fig. 1. System Model.

The aforementioned works based on DRL mainly focus on solving optimization problems by maximizing the expected cumulative reward of the agent. However, they do not consider increasing the agent's exploration of the environment to expedite DRL convergence and achieve better solutions. Therefore, this paper will use SAC algorithm instead of the traditional DDPG algorithm. To the best of our knowledge, current research rarely consider establishing task queue in UAV networks to handle user task concurrency scenarios. Some existing research works mainly manage user tasks through a FCFS queue [14] [22] [23]. However, the FCFS queue will lead to longer waiting delay, so SJF queue model is proposed by us to further reduce user's delay.

III. System Model

As shown in Fig. 1, a multi-UAV assisted MEC network is proposed. In this scenario, M UAVs act as aerial servers to assist N ground users (GUs) in task offloading. Each GU randomly generates a computing task in each time slot, which can be processed locally on the GU or offloaded to UAVs for auxiliary computing. UAVs fly in three-dimensional space and can communicate with GUs through LoS channel. The tasks of GUs are transmitted to UAVs through orthogonal frequency division multiple access (OFDMA).We set each UAV to carry a tiny single-core server and process offloaded tasks from GUs by serial computing. To adapt to high-concurrency user task scenarios, the SJF queue is established on each UAV to manage and schedule user tasks. Specifically, when user tasks are offloaded to UAV, UAV will queue up these tasks according to computing delay of each task and process these tasks sequentially. Because compared with traditional FCFS queuing model [14], SJF can make waiting delay of tasks

shorter. Considering GU's maximum tolerance delay, we set a maximum delay T^{max} for each task, and T^{max} is the size of a time slot. Therefore all tasks will be required to be completed within their generated time slot. We assume that a control center is deployed at the edge base station, which can gather real-time information from GUs and UAVs, run our proposed algorithm, and then send scheduling instructions to GUs and moving instructions to UAVs. Since the size of the information data sent by the GU to the control center is very small compared to the size of the task data transmitted to the UAV, we will ignore the existence of the control center in the subsequent model construction and only consider the communication framework composed of the GUs and the UAVs [24] [25].

A. UAVs' Motion Model

It is assumed that each UAV has sufficient energy to support wireless communication, auxiliary computation and three-dimensional flight during a period of $\mathcal{T}$, where $\mathcal{T} = \{1, 2, \ldots, T\}$. There are a total of M UAVs with the set denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$ and N GUs with the set defined as $\mathcal{N} = \{1, 2, \ldots, N\}$. In each time slot $t \in \mathcal{T}$, the motion of the m-th UAV can be described by horizontal direction $\theta_m[t]$, vertical direction $\phi_m[t]$ and flying velocity $v_m[t]$ [26]. Each time slot has a variation, $\Delta\theta_m[t]$, $\Delta\phi_m[t]$ and $\Delta v_m[t]$ respectively. Therefore, the following iterative formulas are proposed:

$$\begin{cases} \theta_m[t] = \theta_m[t-1] + \Delta\theta_m[t], m \in \mathcal{M}, t \in \mathcal{T}, \\ \phi_m[t] = \phi_m[t-1] + \Delta\phi_m[t], m \in \mathcal{M}, t \in \mathcal{T}, \\ v_m[t] = v_m[t-1] + \Delta v_m[t], m \in \mathcal{M}, t \in \mathcal{T}. \end{cases} \quad (1)$$

We assume that in the t-th time slot, the m-th UAV can fly in a vertical direction as:

$$0 \le \phi_m[t] \le \pi, \quad m \in \mathcal{M}, t \in \mathcal{T}. \quad (2)$$

At the same time, the horizontal direction is defined as:

$$0 \le \theta_m[t] \le 2\pi, \quad m \in \mathcal{M}, t \in \mathcal{T}. \quad (3)$$

Each UAV has a maximum flying velocity v^{max} due to the maximum power limit. Thus, it has:

$$0 \le v_m[t] \le v^{max}, \quad m \in \mathcal{M}, t \in \mathcal{T}. \quad (4)$$

In Cartesian coordinate system [27], the coordinate of the m-th UAV at time slot t could be updated by following incremental formulas:

$$\begin{cases} x_m[t] = x_m[t-1] + v_m[t]\sin\phi_m[t]\cos\theta_m[t]\delta, \\ y_m[t] = y_m[t-1] + v_m[t]\sin\phi_m[t]\sin\theta_m[t]\delta, \\ z_m[t] = z_m[t-1] + v_m[t]\cos\phi_m[t]\delta, \end{cases} \quad (5)$$

where δ is the size of a time slot. The above coordinates are expressed as $\boldsymbol{p}_m[t] = [x_m[t], y_m[t], z_m[t]]$. since the UAV flying area has range restriction, the following coordinate constraint is obtained:

$$[0, 0, z^{min}] \le \boldsymbol{p}_m[t] \le [x^{max}, y^{max}, z^{max}], \quad (6)$$

where x^{max}, $y^{max}$and z^{max} are the length, width of flying area and the maximum flying height of UAV respectively, and z^{min} is minimum flying height.

In order to avoid collision of UAVs in the process of flying, a minimum distance d^{min} should be maintained between each UAV. Thus, it has:

$$\|\boldsymbol{p}_m[t] - \boldsymbol{p}_j[t]\| \ge d^{min}, \quad \forall m, j \in \mathcal{M}, t \in \mathcal{T}, m \ne j. \quad (7)$$

B. Service Assignment

Lemma 1. *For tasks in computing queue, the average delay of SJF is lower than other queuing processing algorithms.*

Proof. Without loss of generality, we assume that there are n tasks in a computing queue. Each task in the queue is represented by q_i, and i is the order in which the task is queued. Arbitrarily arrange these tasks to get the queuing scheme: $\boldsymbol{Q} = \{q_1, q_2, \ldots, q_n\}$, and the computing delay of each task is $\boldsymbol{T}_c = \{t_1, t_2, \ldots, t_n\}$. Therefore, the average processing delay of any queuing algorithm scheme is expressed as:

$$T_1 = \frac{t_1 + (t_1 + t_2) + \cdots + (t_1 + t_2 + \cdots + t_n)}{n}. \quad (8)$$

As for the SJF queue, the tasks in $\boldsymbol{Q}$ are arranged in ascending order according to the task computing delay. So SJF queuing scheme is obtained as $\boldsymbol{Q}' = \{q'_1, q'_2, \ldots, q'_n\}$, where computing delay of each task is $\boldsymbol{T}'_c = \{t'_1, t'_2, \ldots, t'_n\}$. Therefore, the average processing delay of SJF is:

$$T_2 = \frac{t'_1 + (t'_1 + t'_2) + \cdots + (t'_1 + t'_2 + \cdots + t'_n)}{n}. \quad (9)$$

Comparing the cumulative items in (8) and (9), it is obvious that:

$$t'_1 \le t_1, \quad (10)$$

$$(t'_1 + t'_2) \le (t_1 + t_2), \quad (11)$$

$$\vdots$$

$$(t'_1 + t'_2 + \cdots + t'_n) \le (t_1 + t_2 + \cdots + t_n). \quad (12)$$

Thus, we have $T_2 \le T_1$, and the equality is holded only when $t_i = t'_i, \forall i \in \{1, 2, \ldots, n\}$. That is to say, the average delay of SJF is lower than that of FCFS or other queuing algorithms. ■

To solve the connection scheduling problem for GU's tasks, a framework is proposed in Fig. 2. Firstly, we sort all tasks in ascending order according to computing delay to get the sorted list $\boldsymbol{H}[t]$ in time slot t. Then the task subset in $\boldsymbol{H}[t]$ is assigned to each UAV through a connection scheduling algorithm. The principle of this framework is that we adopt the SJF queuing model on the UAVs, so the task queue on each UAV must be an ordered subset of $\boldsymbol{H}[t]$.

In order to find the corresponding task generation device from the task in $\boldsymbol{H}[t]$, we define a hash table $\boldsymbol{Hash}[t]$ to record the one-to-one correspondence between tasks in $\boldsymbol{H}[t]$ and the task generation device, and define the set of $\boldsymbol{H}[t]$ as $\mathcal{K} = \{1, 2, \ldots, N\}$. So, it has:

$$\boldsymbol{Hash}_k[t] = n, \quad \forall k \in \mathcal{K}, n \in \mathcal{N}, \quad (13)$$

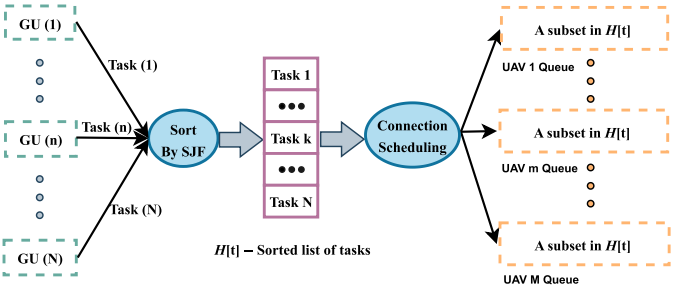


Fig. 2. Service Assignment.

which indicate that the task ranked k in $\boldsymbol{H}[t]$ is generated by the n-th GU.

The task generated by GU in any time slot can be computed locally or optionally offloaded to UAVs for computing. Consenquently, we use a binary variable $a_{k,m}[t]$ to indicate the connection state between the $\boldsymbol{Hash}_k[t]$-th GU and the m-th UAV. Then, the following constraint is yield:

$$a_{k,m}[t] \in \{0,1\}, \quad \boldsymbol{Hash}_k[t] = n, t \in \mathcal{T}, \tag{14}$$

where $a_{k,m}[t] = 1, m \neq 0$ indicates established connection, while $a_{k,m}[t] = 1, m = 0$ means the $\boldsymbol{Hash}_k[t]$-th GU compute task locally in t-th time slot, and otherwise, $a_{k,m}[t] = 0$. We define a new set $\mathcal{M}' = \{0, 1, 2, \ldots, M\}$ to represent all cases in which a task is executed, where $m = 0$ indicates that GU computes the task locally.

In the communication mode of OFDMA, a UAV can serve multiple GUs, but a GU can only connect to one UAV at most [28]. Thus, we obtain:

$$\sum_{m=0}^{M} a_{k,m}[t] = 1, \quad \forall k \in \mathcal{K}, m \in \mathcal{M}', t \in \mathcal{T}, \tag{15}$$

$$\sum_{k=1}^{N} a_{k,m}[t] \leq Z^{max}, \quad \forall k \in \mathcal{K}, m \in \mathcal{M}, t \in \mathcal{T}, \tag{16}$$

where Z^{max} is the maximum number of GUs that can be connected to each UAV, due to the limited number of subcarriers in OFDMA communication.

We define $S_n[t] = \{D_n[t], F_n[t]\}$ to denote the task generated by the n-th GU in time slot t, where $D_n[t]$ is the size of the task, $F_n[t] = \xi D_n[t]$ denotes the CPU cycles required to compute the task, and ξ represents the CPU cycles required to compute per bit of task [29]. Then, we define the variable $s_m[t]$ to represent the sum of task that the m-th UAV has processed from the beginning to the time slot t as:

$$s_m[t] = \sum_{t'=0}^{t} \sum_{k=1}^{N} a_{k,m}[t'] D_n[t], \forall k \in \mathcal{K}, \boldsymbol{Hash}_k[t] = n. \tag{17}$$

To make full use of the computing resource of UAVs and avoid some UAVs processing a large number of task data but some UAVs computing very few task data, a fairness index $L^u[t]$ is introduced to indicate the load fairness state of UAVs:

$$L^u[t] = \frac{(\sum_{m=1}^{M} s_m[t])^2}{M \sum_{m=1}^{M} s_m[t]^2}, \quad \forall m \in \mathcal{M}, t \in \mathcal{T}, \tag{18}$$

where $L^u[t]$ is Jain's Fairness index, which varies from $1/M$ to 1. In particular, $L^u[t]$ is close to 1 if the sum of task size computed by each UAV is similar at time slot t [30].

C. Communication Model

We denote coordinate of the n-th GU as $\boldsymbol{w}_n = [x_n, y_n, 0]$, so the horizontal distance between the n-th GU and the UAV m in time slot t could be obtained by:

$$d^h_{k,m}[t] = \sqrt{(x_m[t] - x_n)^2 + (y_m[t] - y_n)^2}, \boldsymbol{Hash}_k[t] = n. \tag{19}$$

Since the antenna of each UAV have a maximum azimuth α^{max}, the maximum communication coverage of m-th UAV in time slot t is:

$$R^{max}_m[t] = z_m[t] \tan(\alpha^{max}), \quad \forall m \in \mathcal{M}, t \in \mathcal{T}. \tag{20}$$

When the GU needs to establish communication with UAV, the horizontal distance between $\boldsymbol{Hash}_k[t]$-th GU and m-th UAV in time slot t should satisfy:

$$a_{k,m}[t] d^h_{k,m}[t] \leq R^{max}_m[t], \quad \boldsymbol{Hash}_k[t] = n. \tag{21}$$

Based on the assumptions in [7] [31] [32], we consider that the communication link between UAV and GU is determined by LoS channel, and the Doppler effect caused by the movement of UAV has been perfectly compensated at receivers. Furthermore, the free path loss model is introduced for channel modeling, so the average channel gain between UAV m and the $\boldsymbol{Hash}_k[t]$-th GU is defined as:

$$h_{k,m}[t] = \beta d^{-2}_{k,m}[t] = \frac{\beta}{\|p_m[t] - w_n\|^2}, \boldsymbol{Hash}_k[t] = n, \tag{22}$$

where β is the channel gain when the reference distance is $1m$, and $d_{k,m}[t]$ is Euclidean distance between m-th UAV and $\boldsymbol{Hash}_k[t]$-th GU in t-th time slot.

According to the Shannon formula, the wireless transmission rate is obtained by:

$$r_{k,m}[t] = b_{k,m}[t] \log_2(1 + \frac{p^{tr} h_{k,m}[t]}{\sigma^2}), \boldsymbol{Hash}_k[t] = n, \tag{23}$$

where $b_{k,m}[t] = B / \sum_{k=1}^{N} a_{k,m}[t]$ is the upload bandwidth of the $\boldsymbol{Hash}_k[t]$-th GU, σ^2 represents Gaussian white noise power, p^{tr} indicates transmission power of GU [33].

D. Task Computation Model

As shown in Fig. 3, each time slot t is divided into two subslots $\delta^m_1[t]$ and $\delta^m_2[t]$ for UAV m. Among them, δ^m_1 is used to receive tasks offloaded by GUs to UAV and δ^m_2 is used to compute received tasks according to the principle of SJF. In addition, the size of $\delta^m_1[t]$ is dynamically adjusted, which is equivalent to the possible maximum transmission delay of GUs within the coverage area of UAV m when the height is $h_m[t]$:

$$\delta^m_1[t] = \frac{D^{max}_m[t]}{r^{worst}_m[t]}, \quad \forall m \in \mathcal{M}, t \in \mathcal{T}, \tag{24}$$

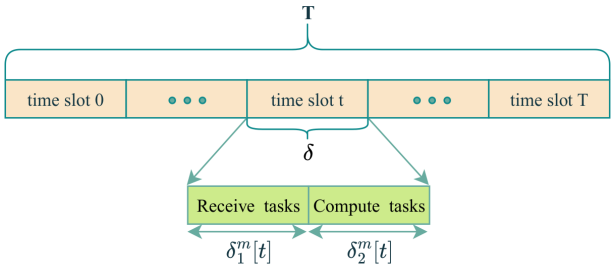


Fig. 3. Time slot allocation of UAV m.

where $D_m^{max}[t]$ is the maximum task size generated by GU within the coverage area of UAV m in time slot t. Therefore, it has:

$$\begin{aligned} D_m^{max}[t] =& max\{D_n[t]\}, \forall n \in \mathcal{N}, \\ & \boldsymbol{Hash}_k[t] = n, d_{k,m}^h[t] \leq R_m^{max}[t]. \end{aligned} \tag{25}$$

In addition, $r_m^{worst}[t]$ indicates the minimum transmission rate of UAV m at height of $h_m[t]$:

$$r_m^{worst}[t] = \frac{B}{z_m^{cov}[t]} \log_2 \left(1 + \frac{p^{tr}\beta}{(R_m^{max^2}[t] + z_m^2[t])\sigma^2}\right), \tag{26}$$

where $z_m^{cov}[t]$ indicates the number of devices covered by the m-th UAV at time slot t:

$$z_m^{cov}[t] = \sum_{k=1}^{N} \boldsymbol{F}(d_{k,m}^h[t] \leq R_m^{max}[t]), \forall k \in \mathcal{K}, \tag{27}$$

where $\boldsymbol{F}$ is a Boolean function. So if $d_{k,m}^h[t] \leq R_m^{max}[t]$ holds, then $\boldsymbol{F}(d_{k,m}^h[t] \leq R_m^{max}[t]) = 1$, otherwise $\boldsymbol{F}(d_{k,m}^h[t] \leq R_m^{max}[t]) = 0$.

Under normal condition, the delay for task to be transferred to UAV m is:

$$T_{k,m}^{tr}[t] = \frac{a_{k,m}[t]D_n[t]}{r_{k,m}[t]}, \forall k \in \mathcal{K}, n = \boldsymbol{Hash}_k[t]. \tag{28}$$

Define the CPU computing frequency of UAV and GU as f^u and f^l respectively. So the delay for task to be computed locally is [34]:

$$T_{k,0}^l[t] = \frac{F_n[t]}{f^l}, \forall k \in \mathcal{K}, n = \boldsymbol{Hash}_k[t], t \in \mathcal{T}. \tag{29}$$

When task $S_n[t]$ is offloaded to m-th UAV, the computing delay is expressed as [34]:

$$T_{k,m}^u[t] = \frac{F_n[t]}{f^u}, \forall k \in \mathcal{K}, n = \boldsymbol{Hash}_k[t], m \in \mathcal{M}. \tag{30}$$

Since the SJF queuing model is applied to UAV, the tasks in the computing queue on each UAV must be an ordered subset of $\boldsymbol{H}[t]$. In addition, the task generated in each time slot is required to be completed in this time slot, so the computing queue of each UAV will only contain tasks generated in the current time slot. Therefore, the waiting delay for k-th task in $\boldsymbol{H}[t]$ to be offloaded to UAV m at time slot t is:

$$T_{k,m}^{wait}[t] = a_{k,m}[t]\Big(\sum_{k'=1}^{k-1} a_{k',m}[t]T_{k',m}^u[t]\Big), \forall t \in \mathcal{T}. \tag{31}$$

Since the processing result transmitted from the UAVs to the GUs is very small compared to the task data transmitted from the GUs to the UAVs, the delay of the result transmission can be ignored [21] [34]. To sum up, when GU performs computation locally, the total delay $T_{k,m}[t]$ of this task is: $T_{k,m}[t] = T_{k,m}^l[t]$. If the task is offloaded to UAV m for computing, the total delay of this task is obtained by:

$$T_{k,m}[t] = \delta_1^m[t] + T_{k,m}^{wait}[t] + T_{k,m}^u[t], \forall k \in \mathcal{K}, m \in \mathcal{M}. \tag{32}$$

That is to say, the total delay of each task is equal to the sum of the receiving task sub-slot, waiting delay, and computing delay.

Considering the QoS, a user tolerance delay T^{max} is set for each task, then the total delay $T_{k,m}[t]$ of each task should satisfy the constraint:

$$T_{k,m}[t] \leq T^{max}, \forall k \in \mathcal{K}, m \in \mathcal{M}, t \in \mathcal{T}, \tag{33}$$

where T^{max} is equal to the size of the time slot.

Furthermore, the total delay of a task being offloaded to a UAV for auxiliary computing can't exceed local computing delay:

$$T_{k,m}[t] \leq T_{k,0}^l[t], \forall k \in \mathcal{K}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{34}$$

Since the user's maximum tolerable delay is the size of a time slot, tasks entering the UAV queue in each time slot will be required to complete within the current time slot. Therefore, the memory size used by the queue of UAV m in time slot t is defined as:

$$C_m[t] = \sum_{k=1}^{N} a_{k,m}[t]D_n[t], \forall k \in \mathcal{K}, n = \boldsymbol{Hash}_k[t]. \tag{35}$$

The maximum memory size of the task queue on each UAV is C_u^{max}, so it has the following constraint:

$$0 \leq C_m[t] \leq C_u^{max}, \forall m \in \mathcal{M}, t \in \mathcal{T}. \tag{36}$$

E. Problem Formulation

Define $\boldsymbol{P} = \{\Delta\theta_m[t], \Delta\phi_m[t], \Delta v_m[t], \forall m \in \mathcal{M}, t \in \mathcal{T}\}$ and $\boldsymbol{A} = \{a_{k,m}[t], \forall k \in \mathcal{K}, m \in \mathcal{M}', t \in \mathcal{T}\}$. Our goal is to jointly optimize the average latency of users and the load fairness of UAVs, that is, minimize the average latency of user tasks and maximize the average load fairness of UAVs. For simplicity, we introduce a variable to represent the objective function, so we has:

$$\begin{aligned} \eta &= \frac{\frac{1}{T}\sum_{t=0}^{T} L^u[t]}{\frac{1}{NT}\sum_{k=1}^{N}\sum_{m=0}^{M}\sum_{t=0}^{T} T_{k,m}[t]} \\ &= \frac{N\sum_{t=0}^{T} L^u[t]}{\sum_{k=1}^{N}\sum_{m=0}^{M}\sum_{t=0}^{T} T_{k,m}[t]}. \end{aligned} \tag{37}$$

Our optimization problem can be formulated as follows:

$$(\mathbf{P1}): \max_{\boldsymbol{A},\boldsymbol{P}} \quad \eta \tag{38}$$

$$s.t. \quad a_{k,m}[t] \in \{0,1\}, \forall k \in \mathcal{K}, m \in \mathcal{M}', t \in \mathcal{T}, \tag{38a}$$

$$\sum_{m=0}^{M} a_{k,m}[t] = 1, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}, \tag{38b}$$

$$\sum_{k=1}^{N} a_{k,m}[t] \leq Z^{max}, \forall m \in \mathcal{M}, t \in \mathcal{T}, \tag{38c}$$

$$0 \leq \phi_m[t] \leq \pi, \forall m \in \mathcal{M}, t \in \mathcal{T}, \tag{38d}$$

$$0 \leq \theta_m[t] \leq 2\pi, \forall m \in \mathcal{M}, t \in \mathcal{T}, \tag{38e}$$

$$0 \leq v_m[t] \leq v^{max}, \forall m \in \mathcal{M}, t \in \mathcal{T}, \tag{38f}$$

$$[0, 0, z^{min}] \leq \boldsymbol{p}_m[t], \forall m \in \mathcal{M}, t \in \mathcal{T}, \tag{38g}$$

$$\boldsymbol{p}_m[t] \leq [x^{max}, y^{max}, z^{max}], \forall m \in \mathcal{M}, t \in \mathcal{T}, \tag{38h}$$

$$\|\boldsymbol{p}_m[t] - \boldsymbol{p}_j[t]\| \geq d^{min}, \forall m, j \in \mathcal{M}, t \in \mathcal{T}, m \neq j, \tag{38i}$$

$$a_{k,m}[t] d^h_{k,m}[t] \leq R^{max}_m[t], \forall k \in \mathcal{K}, m \in \mathcal{M}, t \in \mathcal{T}, \tag{38j}$$

$$T_{k,m}[t] \leq T^{max}, \forall k \in \mathcal{K}, m \in \mathcal{M}, t \in \mathcal{T}, \tag{38k}$$

$$T_{k,m}[t] \leq T^l_{k,0}[t], \forall k \in \mathcal{K}, m \in \mathcal{M}, t \in \mathcal{T}, \tag{38l}$$

$$0 \leq C_m[t] \leq C^{max}_u, \forall m \in \mathcal{M}, t \in \mathcal{T}. \tag{38m}$$

In **P1**, constraint $(38a)$ indicates that the task of each GU can only be either local computing or offloading computing. Constraint $(38b)$ denotes that a GU's task can at most be offloaded to one UAV, while $(38c)$ implies that the number of GUs connected to each UAV can't exceed Z^{max}. Constraints $(38d)$, $(38e)$, and $(38f)$ represent limitations of UAV's movement in direction and velocity. Constraints $(38g)$, $(38h)$, and $(38i)$ indicate that UAVs must fly within the specified working area and maintain a certain distance from each other to avoid collision. Constraint $(38j)$ means that only GUs can be connected within the communication coverage of UAVs. Constraint $(38k)$ represents the maximum tolerable delay for GU. Lastly, constraint $(38m)$ indicates the limit on UAV queue memory size. Problem **P1** is a mixed integer nonlinear programming problem (MINLP), because it includes a binary integer variable $\boldsymbol{A}$ and a continuous variable $\boldsymbol{P}$, which are generally difficult to solve. Thus, we propose an algorithm to solve this complex problem based on DRL.

IV. PROPOSED ALGORITHM

A. Overall Algorithm Design

In our model, the three-dimensional trajectory of UAV has Markov characteristics, enabling us to optimize UAV trajectory through DRL [19]. To apply DRL, we define the state space, action space, and reward function as follows:

1) State Space: $\mathcal{S} = \{p_m[t], D_n[t], \forall m \in \mathcal{M}, n \in \mathcal{N}, t \in \mathcal{T}\}$, where $p_m[t]$ is the coordinate of the m-th UAV in time slot t, and $D_n[t]$ is the task size generated by the n-th GU.

2) Action Space: $\mathcal{A} = \{\Delta\theta_m[t], \Delta\phi_m[t], \Delta v_m[t], \forall m \in \mathcal{M}, t \in \mathcal{T}\}$, where the action space $\mathcal{A}$ is consisted of the flying direction variation and flying speed increment of all UAVs in each time slot.

3) Reward: According to the objective function we need to optimize, the reward function is defined as:

$$r(t) = \frac{L^u[t]}{\frac{1}{N}\sum_{k=0}^{N}\sum_{m=0}^{M} T_{k,m}[t]} - p, \tag{39}$$

where p represents the penalty for any UAV violating the constraints in **P1**.

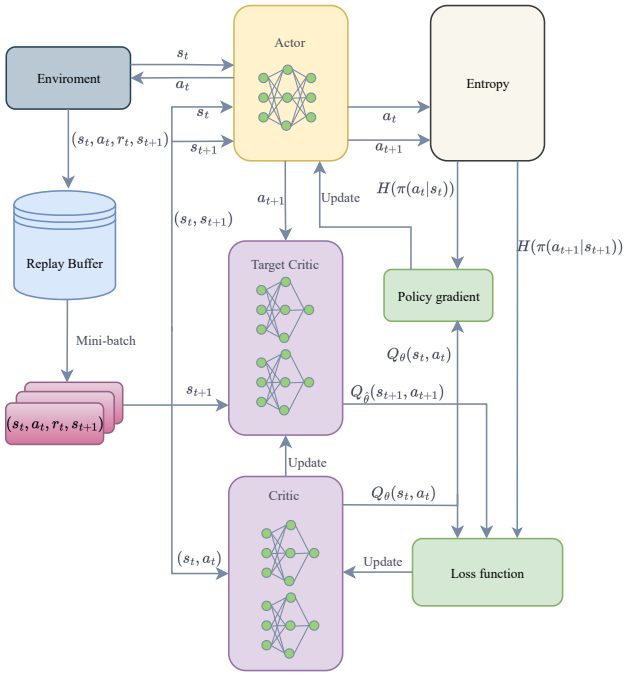


Fig. 4. SAC Framework.

Since the SAC algorithm offers a better ability to explore and can yield superior solutions compared to the traditional DDPG algorithm [35], we use the SAC algorithm to solve the three-dimensional trajectory of UAVs. As shown in Fig. 4, the SAC mainly consists of the environment, actor-network, two critic networks, and two critic-target networks. In our model scenario, all UAVs are regarded as agents. For the actor-network, it will output the corresponding action $a(t)$ based on the current state $s(t)$ and strategy π_φ of the agent. Therefore, it has:

$$a(t) \sim \pi_\varphi(\cdot|s(t)), \tag{40}$$

where φ indicates the parameters of the actor-network. After executing the action $a(t)$, the agent receives immediate reward $r(t)$ and transfers to the next state $s(t+1)$. Finally, the agent will store the quardruple $(s(t), a(t), r(t), s(t+1))$ into the experience replay buffer as a data sample for critic network training.

The SAC algorithm aims to maximize the expected cumulative reward and introduces an entropy regularization term to evaluate the agent's strategy. Consequently, the optimal strategy is defined as:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\Big\{\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(t) + \alpha H\Big(\pi_\varphi(\cdot \mid s(t))\Big)\right]\Big\}, \tag{41}$$

where γ is the discount factor, and α indicates an entropy regularization coefficient, which is used to control the importance of entropy. The calculation formula of entropy is expressed as:

$$H\Big(\pi_\varphi(\cdot|s(t))\Big) = \mathbb{E}\Big[-\log(\pi_\varphi(\cdot|s(t)))\Big]. \tag{42}$$

The critic network outputs the Q value based on the state-action pair $(s(t), a(t))$ to evaluate the strategy of the actor-

network. The Q value can be obtained through:

$$Q(s(t),a(t))=\mathbb{E}\Big[\sum_{i=t}^{T}\gamma^{i-t}r(i)|s(t),a(t)\Big]. \tag{43}$$

The critic-target network guides the parameter update of the critic network by calculating the target Q value. The target Q value is defined as:

$$\begin{aligned}y(t)=r(t)+\gamma\min_{i=1,2}Q_{\hat{\theta}_i}\Big(s(t+1),a(t+1)'\Big)\\-\log\Big(\pi_{\varphi}\Big(a(t+1)\mid s(t+1)\Big)\Big),\end{aligned} \tag{44}$$

where θ represents the parameters of the critic-target network. When training data in the experience replay buffer reaches the specified amount, the parameters of the critic network are updated by performing mini-batch sampling. The loss function of the critic network is:

$$L_{Q_i}(\theta)=\mathbb{E}\Big[\frac{1}{2}\Big(Q_{\theta_i}\Big(s(t),a(t)\Big)-y(t)\Big)^2\Big],\forall i\in\{1,2\}. \tag{45}$$

The parameters of critic-target network are updated through soft update:

$$\hat{\theta}_i=\tau\theta_i+(1-\tau)\hat{\theta}_i,\forall i\in\{1,2\}, \tag{46}$$

where τ is the updating rate.

The goal of SAC is to maximize the cumulative reward while making the strategy more exploratory. Therefore, the loss function of the actor-network is defined as:

$$L_{\pi}(\varphi)=\mathbb{E}\Big[\alpha\log\Big(\pi(a(t)'|s(t)\Big)-\min_{i=1,2}Q_{\theta_i}\Big(s(t),a(t)'\Big)\Big], \tag{47}$$

where $a(t)'$ indicates the action obtained after the state s_t is input into the actor network.

In the SAC algorithm, we must dynamically adjust the regularization coefficient α to ensure that the agent focuses more on strategy improvement during training. The loss function of the coefficient α is defined as:

$$L(\alpha)=\mathbb{E}\Big[-\alpha\log\Big(a(t)'|s(t)\Big)-\alpha H_0\Big], \tag{48}$$

where H_0 is the target entropy value, generally set to the negative value of the action dimension.

According to the above definitions, the overall algorithm framework for solving problem **P1** is proposed in **Algorithm 1**. From line 1 to line 3, we initialize the parameters of the actor-network and critic network, copy the critic network parameters to the target network, and then initialize the experience replay buffer. From lines 7 to 11, all UAVs are regarded as an agent and transferred to the next state after executing the action output by the actor-network. Then the connection scheduling is solved by **Algorithm 2**, and the system reward is obtained according to the reward function. To ensure subsequent training, the experience $[s(t),a(t),r(t),s(t+1)]$ is stored in experience replay buffer in line 12. If sample data in the experience replay buffer reaches a specified amount, mini-batch sampling will be performed on lines from 13 to 19, and agent will be trained according to the loss function of each neural network.

Algorithm 1 SAC Algorithm

1: Initialize Critic network Q_{θ_1} and Q_{θ_2} with parameters θ_1 and θ_2, respectively, and Actor network $\pi(s(t)|\varphi)$ with parameters φ;
2: Initialize target network $Q_{\hat{\theta}_1}$ and $Q_{\hat{\theta}_2}$ by parameters $\hat{\theta}_1\leftarrow\theta_1$ and $\hat{\theta}_2\leftarrow\theta_2$;
3: Initialize experience replay buffer R;
4: **for** epcho $e=1$ to E^{max} **do**
5: Initialize the state of environment $s(t)$;
6: **for** time slot $t=0,\ldots,T$ **do**
7: obtain action based on actor network $a(t)=\pi_{\varphi}(s(t))$
8: All UAV execute $a(t)$;
9: Obtain $s(t+1)$;
10: Solve connection scheduling by **Algorithm 2**;
11: Caculate reward $r(t)$ from formula (39);
12: Store experience $[s(t),a(t),r(t),s(t+1)]$ into replay buffer R;
13: **if** the replay buffer is full **then**
14: Sample $\{s_j,a_j,r_j,s_{j+1}\}_{j=\{1,2,\ldots,J\}}$ from R;
15: For each $[s_j,a_j,r_j,s_{j+1}]$, caculate $y_j(t)$ by formula (44) with target network;
16: Update θ_1 and θ_2 of critic network by minimizing their loss function according to (45);
17: Update φ of actor network by minimizing (47);
18: Update entropy regulation cofficient α by (48);
19: Update $\hat{\theta}_1$ and $\hat{\theta}_2$ of target network according to (46);
20: **end if**
21: **end for**
22: **end for**

B. Connection Scheduling Algorithm

According to the characteristic of the objective function and SJF queue, the optimal solution of connection scheduling should satisfy that user tasks offloaded to each UAV queue are arranged in ascending order of task size and the total task size processed by each UAV is as close as possible. Therefore, we propose **Algorithm 2** to approximate the optimal solution based on this idea. First, initialize the connection scheduling matrix $\boldsymbol{A}$ and the used memory size of each UAV queue to 0. Then, from line 3 to line 25, the assignment of tasks in sorted list $\boldsymbol{H}[t]$ is processed sequentially. Specifically, if GU corresponding to the task in list $\boldsymbol{H}[t]$ is not covered by any UAV, only local computing can be performed. If the task can be received by multiple UAVs, the UAV with the shortest delay is selected for offloading under the condition of satisfying constraints. Among them, the line 7 corresponds to the constraint $(38j)$, and the line 9 corresponds to the constraint $(38c)$. In addition, the 11-th line and the 13-th line represent constraints $(38k)$, $(38l)$, $(38m)$ respectively.

C. Complexity Analysis

In this section, we will briefly analyze the time complexity of the overall algorithm. The time complexity analysis in **Algorithm 1** can be mainly divided into two parts. One part involves neural network training in **Algorithm 1**, and the other part involves solving connection scheduling in **Algorithm 2**.

Algorithm 2 Connection Scheduling Algorithm

Initialize $\boldsymbol{A}, \forall k \in \mathcal{K}, m \in \mathcal{M}'$;
Initialize $C_m[t] = 0$ for each UAV;
for each task $k = 1, 2, \ldots, N$ in sorted list $\boldsymbol{H}[t]$ **do**
  Initialize $j = 0$;
  Calculate $T^l_{k,0}[t]$ accoding to the (29) and let $temp = T^l_{k,0}[t]$;
  for UAV $m = 1, \ldots, M$ **do**
    if $(38j)$ is met **then**
      Calculate $T_{k,m}[t]$ according to the (32);
      if $\sum_{k=1}^{N} a_{k,m}[t] >= Z^{max}$ **then**
        continue;
      else if $T_{k,m}[t] > temp$ **or** $T_{k,m}[t] > T^{max}$ **then**
        continue;
      else if $C_m[t] + D_n[t] > C_u^{max}$, where $n = \boldsymbol{Hash}_k[t]$ **then**
        continue;
      else
        $temp = T_{k,m}[t]$;
        $j = m$;
      end if
    end if
  end for
  Let $a_{k,j}[t] = 1$;
  if $j > 0$ **then**
    $C_j[t] = C_j[t] + D_n[t]$, where $n = \boldsymbol{Hash}_k[t]$;
  end if
end for
Return $\boldsymbol{A}$

Generally speaking, the complexity of a neural network is represented by the number of operations, which is determined by the dimensions of input state and action, the number of network layers and neurons in each layer [16].

First, define the layers of the actor network and critic network in the SAC algorithm as I^a, I^c respectively, and the number of neurons in the i-th layer as u_i^a, u_i^c. After the agent has been trained for E^{max} epochs, its time complexity is approximately:

$$O_1 = TE^{max}(\sum_{i=1}^{I^a-1} u_i^a u_{i+1}^a + \sum_{i=1}^{I^c-1} u_i^c u_{i+1}^c). \quad (49)$$

In **Algorithm 2**, we need to sort all tasks to build a list $\boldsymbol{H}[t]$, and the sorting complexity is $O(N^2)$ in the worst case. For each task in $\boldsymbol{H}[t]$, it is necessary to judge the status of its offloading to each UAV, and the complexity introduced here is $O(MN)$. In summary, the time complexity of the overall algorithm is:

$$O = TE^{max}(\sum_{i=1}^{I^a-1} u_i^a u_{i+1}^a + \sum_{i=1}^{I^c-1} u_i^c u_{i+1}^c + N^2 + MN). \quad (50)$$

V. SIMULATION RESULTS

In this section, the performance of the algorithm proposed in this paper is verified through numerical simulation. The simulation is built on an Intel i5-12500H processor, Python 3.9, and Pytorch 2.0.1. Both the actor network and the critic network use three fully-connected hidden layers with $[800, 600, 400]$ neurons. The actor network learning rate is 3×10^{-4}, and the critic network learning rate is 3×10^{-3}. For the UAVs flying area, we set a cube area with length and width of $400m$ and height of $200m$. Four UAVs are deployed to the flight area and their initial positions are $[30, 30, 125]$, $[30, 370, 125]$, $[370, 30, 125]$, $[370, 370, 125]$ respectively. The 80 GUs are randomly distributed in the experimental area, and the size of the task generated by each GU in each time slot is between $100KB$ and $240KB$. The rest of the parameters are shown in Table I [15] [29].

TABLE I
SIMULATION PARAMETERS

Parameters	Settings	Parameters	Settings
M	4	N	80
T	25	v^{max}	$30m/s$
z^{min}	$50m$	z^{max}	$200m$
d^{min}	$30m$	Z^{max}	30
α^{max}	$\frac{\pi}{6}$	B	200MHz
β	1.42×10^{-4}	σ	-90dbm
p^{tr}	0.1W	T^{max}	1s
f^l	2.5GHz	f^u	40GHz
C_u^{max}	10MB	ξ	1000cycle/bit

A. Comparison Algorithms

1). **Particle Swarm Optimization (PSO)**: PSO solves our problem by updating the population of candidate particles [36]. Each particle is composed of the increment of flying direction and speed in each time slot of the UAV, and the fitness value is calculated according to the (39). It should be noted that PSO is designed to solve complex non-convex problems in the design phase, rather than being a method for real-time fast decision-making. We introduce it for comparison to highlight the fast decision-making performance of DRL.

2). **Deep Deterministic Policy Gradient (DDPG)**: DDPG has been widely used in optimizing UAV networks [24] [19]. To make a fair comparison with SAC, DDPG is used to optimize the trajectory of the UAV and the user connection scheduling is also obtained by **Algorithm 2**. The network structure of DDPG is set to be the same as SAC, but the impact of action entropy on the agent is not considered in DDPG. Specifically, DDPG only focuses on maximizing the long-term cumulative reward without considering increasing the agent's exploration of the environment to speed up DRL convergence and obtain better solutions.

3). **Random Moving (RM)**: Each UAV randomly selects the flying direction and flying speed and obtains the connection scheduling of each task according to **Algorithm 2**.

4). **Local Computing (LC)**: All tasks are processed locally without offloading.

B. Comparison of the Different Algorithms

First, we compare the convergence of SAC and DDPG. As shown in Fig. 5a, SAC can find better solutions within the same training epochs and converge faster. Because the DDPG algorithm does not introduce action entropy, its exploration of

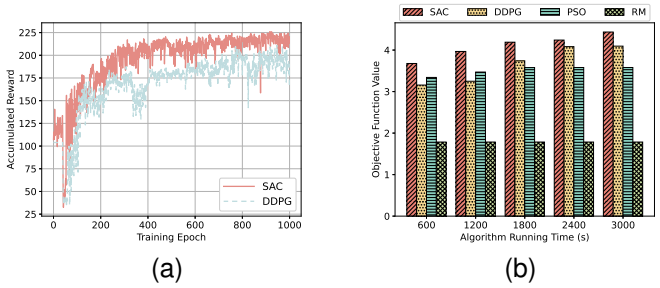


Fig. 5. Compare the convergence and objective function values of different algorithms. (a) The convergence of SAC, DDPG. (b) The objective function value of SAC, DDPG, PSO, RM.

the environment is not as effective as the SAC algorithm we proposed. The objective function values obtained by different algorithms are compared in Fig. 5b. As can be seen, our proposed SAC algorithm achieves the best optimization of the objective function value. In addition, when compared with heuristic algorithms such as PSO, DRL algorithms like DDPG and SAC can better avoid falling into local optimal solutions.

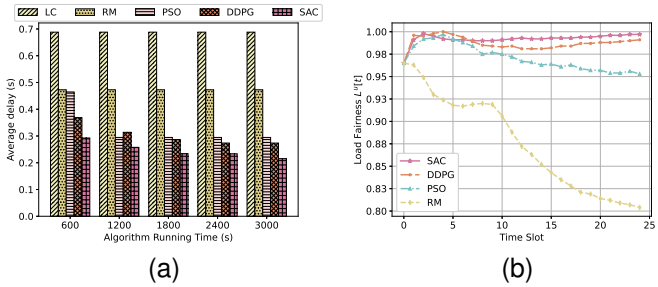


Fig. 6. Compare the average user delay and UAV load fairness of different algorithms. (a) The average delay of LC, RM, PSO, DDPG, SAC. (b) The load fairness of RM, PSO, DDPG, SAC.

As shown in Fig. 6a, we compare the average delay of different algorithms after they have run for the same time. Since the effects of the LC and RM algorithms are independent of the algorithm running time, the values in Fig. 6a remained unchanged. With the running time of PSO, DDPG, and SAC algorithms increasing, the average user delay is effectively reduced. Fig. 6b reflects the change in the UAV load fairness index within a period. The load fairness index of different algorithms at time slot 0 is the same, because the initial positions of the UAVs at time slot 0 are the same for each algorithm and **Algorithm 2** is used in these algorithms to solve connection scheduling. In addition, it is obvious that our proposed SAC algorithm is the most effective for load fairness optimization of UAVs compared with other algorithms.

To further describe the load fairness index changes of UAVs, we analyze the total task size processed by each UAV in Fig. 7. It can be found that the total task size processed by the 4 UAVs in the same time slot is very close. There is no situation where some UAVs compute a large number of task data and some UAVs process a small amount of task data, which demonstrates that our optimization of the load fairness index is effective. In Fig. 8a, we compare the number of GUs performing auxiliary offloading at each slot in different algorithms. Compare with other algorithms, the SAC algorithm can assist more GUs in task computing at the same time

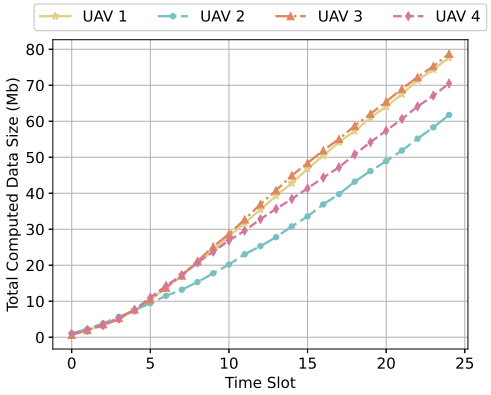


Fig. 7. The total computing data size of each UAV.

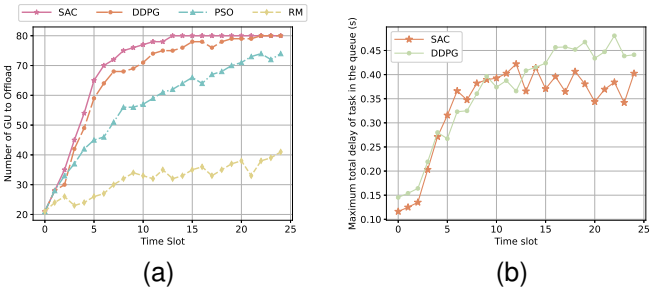


Fig. 8. Compare the number of GUs participating in offloading in each time slot within a period and the maximum total delay of last task in the queue. (a) The number of GUs participating in offloading in SAC, DDPG, PSO, RM. (b) The maximum total delay of last task in the queue in SAC, DDPG.

slot. In addition, after the UAVs have flown for a certain number of time slots, all GUs can be connected by UAVs to perform auxiliary computing. To determine whether the UAVs can effectively provide computing services for all offloaded user tasks, we analyze the maximum total delay of the last task in each UAV task queue in each time slot in Fig. 8b. We can find that although all GUs offload tasks to UAVs from the 15-th to the 25-th time slot, the maximum total delay of the tasks in the SAC algorithm does not exceed 0.45 seconds and is less than the maximum tolerable delay (1 second) of the GUs. In addition, when all GUs participate in offloading, the maximum latency of task in the proposed SAC algorithm is lower than that of the traditional DDPG algorithm.

The three-dimensional trajectories of UAVs are depicted in Fig. 9. The purple dots represent GUs, while the stars of different colors indicate the trajectories of different UAVs. We can observe that the positions of different UAVs in the same time slot are relatively scattered. This allows the computing resources of each UAV to be fully utilized to serve more GUs.

The impact of the FCFS and SJF queuing model on average delay is reflected in Fig. 10. Regardless of whether the SAC or the DDPG is used, the average processing delay of SJF is lower than that of FCFS. This is because the FCFS queuing model increases the waiting delay of user tasks in the queue.

In Fig. 11a, we analyze the impact of varying numbers of UAVs on assisted GU computing. As the number of UAVs increases, the average delay of GUs is constantly decreasing. However, it should be noted that as the number of UAVs

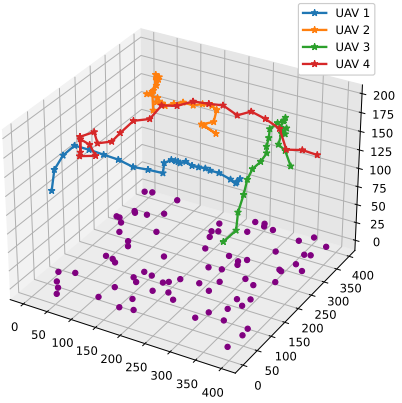


Fig. 9. Three-dimensional flying trajectories of UAVs.

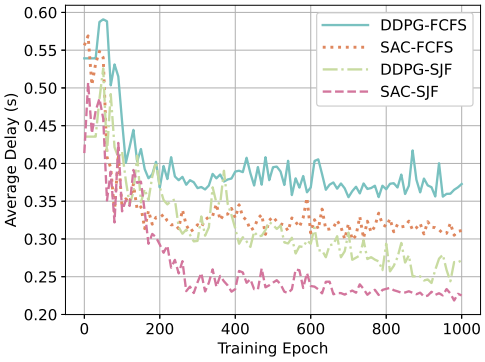


Fig. 10. Compare the average delay of FCFS and SJF in SAC, DDPG.

increases, the improvement in user delay becomes increasingly limited. To show the relative optimization effect of increasing the number of UAVs, we compare the relative difference in average delay for changes in the number of UAVs in Fig. 11b. When the number of UAVs is 2, the optimization of user delay is reduced by 0.135 seconds compared to 1 UAV. However, when the number of UAVs is 8, the optimization of user delay is only reduced by 0.011 seconds compared with 7 UAVs. In Fig. 12, we compare the impact of different numbers of GUs on the average delay. We can find that the smaller the number of GUs, the lower the average delay. This is because the UAVs will have sufficient computing resources to provide assisted computing.

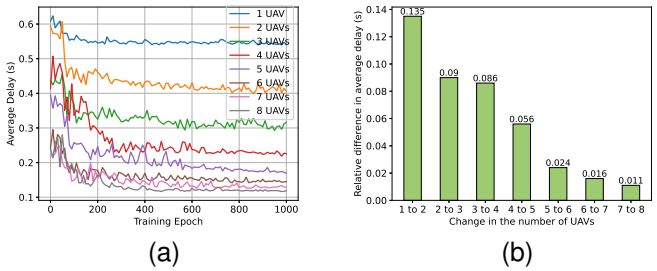


Fig. 11. Compare the impact of different numbers of UAVs on the average user delay. (a) The delay optimization. (b) The relative difference in average delay for change in the number of UAVs.

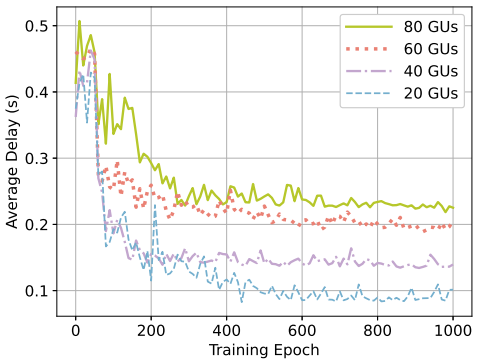


Fig. 12. Compare delay optimization of different numbers of GUs based on SAC.

C. Test Results Analysis of Different Algorithms

In order to increase the DRL model's ability to handle dynamic changes, we further train the model by continuously changing the initial positions of UAVs and the task size generated by GUs in each time slot [24]. Then the performance of the model is tested in Table II. For a clear comparison, we have introduced the running results of PSO, RM, and LC algorithms in Table II, where the SAC and DDPG algorithms are significantly better than the PSO, RM and LC algorithms in optimizing the average user delay and UAV load fairness. This shows that the model we trained has good generalization ability. In addition, we can find that DRL algorithms such as SAC and DDPG are more time-consuming in model training, taking 4936.89 seconds and 4106.68 seconds respectively, but only 0.22 seconds are consumed to obtain the solution during testing. The PSO is worse than that of SAC and DDPG, and the solution time is relatively long. Therefore, the DRL algorithm has greater advantages when dealing with dynamic changes and fast decision-making application scenarios.

TABLE II
MODEL TEST RESULTS

Algorithm	Delay (s)	Fairness	Executed Time (s)	Traing (s)
SAC	0.265	0.943	0.22	4936.89
DDPG	0.307	0.917	0.20	4106.58
PSO	0.384	0.897	2580.47	
RM	0.516	0.812	0.16	
LC	0.695		0.08	

VI. CONCLUSION

With the goal as reducing user average delay and enhancing the load fairness of the UAVs, a NP-hard problem is formulated and solved by jointly optimizing the three-dimensional trajectories of UAVs and the GUs' connection scheduling. Specifically, we use SAC algorithm to optimize the three-dimensional trajectories of UAVs and propose a SJF model to help designing a low-complexity connection scheduling algorithm.

For future research, the current work can be further expanded in the following aspects. For example, we can consider

how to establish multiple task queues under multi-core processors to serve user task concurrency scenarios. In addition, we can introduce the optimization of CPU computing frequency allocation and explore some effective algorithms to reduce the training time of DRL algorithm. Finally, we can also study how to use convex optimization to solve complex problems in multi-UAV networks to compare with the solution of DRL. The above issues will be further investigated in our future work.

ACKNOWLEDGMENTS

This work is jointly supported by A Project Supported by Scientific Research Fund of Hunan Provincial Education Department (Grant No. 23A0258), Hunan Provincial Natural Science Foundation of China (Grant No. 2021JJ30736, 2023JJ50331), Practice Innovation and Entrepreneurship Ability Enhancement Plan of Changsha University of Science and Technology (Grant No. CLSJCX23103), Natural Science Foundation of China (Grant No. 62272063).

REFERENCES

[1] Y. Zeng, S. Chen, Y. Cui, J. Yang, and Y. Fu, "Joint resource allocation and trajectory optimization in uav-enabled wirelessly powered mec for large area," *IEEE Internet of Things Journal*, vol. 10, no. 17, pp. 15 705–15 722, 2023.

[2] H. Yang, J. Zhao, Z. Xiong, K.-Y. Lam, S. Sun, and L. Xiao, "Privacy-preserving federated learning for uav-enabled networks: Learning-based joint scheduling and resource management," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3144–3159, 2021.

[3] P. Ranaweera, A. D. Jurcut, and M. Liyanage, "Survey on multi-access edge computing security and privacy," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 2, pp. 1078–1124, 2021.

[4] Y. Zeng, J. Lyu, and R. Zhang, "Cellular-connected uav: Potential, challenges, and promising technologies," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 120–127, 2019.

[5] Q. Wu, L. Liu, and R. Zhang, "Fundamental trade-offs in communication and trajectory design for uav-enabled wireless network," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 36–44, 2019.

[6] A. M. Seid, J. Lu, H. N. Abishu, and T. A. Ayall, "Blockchain-enabled task offloading with energy harvesting in multi-uav-assisted iot networks: A multi-agent drl approach," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3517–3532, 2022.

[7] X. Pang, N. Zhao, J. Tang, C. Wu, D. Niyato, and K.-K. Wong, "Irs-assisted secure uav transmission via joint trajectory and beamforming design," *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 1140–1152, 2022.

[8] Z. Lin, S. Bi, and Y.-J. A. Zhang, "Optimizing ai service placement and resource allocation in mobile edge intelligence systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7257–7271, 2021.

[9] Z. Song, J. An, H. Ding, and H. Dai, "Optimal relay probing for uav millimeter wave communications with beam training overhead," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 7351–7363, 2023.

[10] X. Pang, G. Gui, N. Zhao, W. Zhang, Y. Chen, Z. Ding, and F. Adachi, "Uplink precoding optimization for noma cellular-connected uav networks," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1271–1283, 2020.

[11] Z. Ning, P. Dong, M. Wen, X. Wang, L. Guo, R. Y. K. Kwok, and H. V. Poor, "5g-enabled uav-to-community offloading: Joint trajectory design and task scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 11, pp. 3306–3320, 2021.

[12] X. Pang, W. Mei, N. Zhao, and R. Zhang, "Intelligent reflecting surface assisted interference mitigation for cellular-connected uav," *IEEE Wireless Communications Letters*, vol. 11, no. 8, pp. 1708–1712, 2022.

[13] X. Yuan, Y. Hu, and A. Schmeink, "Joint design of uav trajectory and directional antenna orientation in uav-enabled wireless power transfer networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3081–3096, 2021.

[14] R. Chen, L. Cui, Y. Zhang, J. Chen, K. Yao, Y. Yang, C. Yao, and H. Han, "Delay optimization with fcfs queuing model in mobile edge computing-assisted uav swarms: A game-theoretic learning approach," in *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*, 2020, pp. 245–250.

[15] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 73–84, 2021.

[16] A. Gao, Q. Wang, W. Liang, and Z. Ding, "Game combined multi-agent reinforcement learning approach for uav assisted offloading," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 12 888–12 901, 2021.

[17] A. Koushik, F. Hu, and S. Kumar, "Deep Q -learning-based node positioning for throughput-optimal communications in dynamic uav swarm network," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 3, pp. 554–566, 2019.

[18] B. Li, Y. Liu, L. Tan, H. Pan, and Y. Zhang, "Digital twin assisted task offloading for aerial edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 10, pp. 10 863–10 877, 2022.

[19] A. M. Seid, G. O. Boateng, S. Anokye, T. Kwantwi, G. Sun, and G. Liu, "Collaborative computation offloading and resource allocation in multi-uav-assisted iot networks: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 203–12 218, 2021.

[20] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient uav control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059–2070, 2018.

[21] A. M. Seid, G. O. Boateng, B. Mareri, G. Sun, and W. Jiang, "Multi-agent drl for task offloading and resource allocation in multi-uav enabled iot edge network," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4531–4547, 2021.

[22] X.-H. Lin, S. Bi, G. Su, and Y.-J. A. Zhang, "A lyapunov-based approach to joint optimization of resource allocation and 3d trajectory for solar-powered uav mec systems," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[23] W. Lee and T. Kim, "Multiagent reinforcement learning in controlling offloading ratio and trajectory for multi-uav mobile-edge computing," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 3417–3429, 2024.

[24] 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 Transactions on Mobile Computing*, vol. 21, no. 10, pp. 3536–3550, 2022.

[25] Y. Zeng, S. Chen, Y. Cui, J. Yang, and Y. Fu, "Joint resource allocation and trajectory optimization in uav-enabled wirelessly powered mec for large area," *IEEE Internet of Things Journal*, vol. 10, no. 17, pp. 15 705–15 722, 2023.

[26] R. Ding, F. Gao, and X. S. Shen, "3d uav trajectory design and frequency band allocation for energy-efficient and fair communication: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 7796–7809, 2020.

[27] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "Uav-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4738–4752, 2019.

[28] Y. Nie, J. Zhao, F. Gao, and F. R. Yu, "Semi-distributed resource management in uav-aided mec systems: A multi-agent federated reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 13 162–13 173, 2021.

[29] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in c-ran with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2018.

[30] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient uav control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059–2070, 2018.

[31] N. Zhao, X. Pang, Z. Li, Y. Chen, F. Li, Z. Ding, and M.-S. Alouini, "Joint trajectory and precoding optimization for uav-assisted noma networks," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3723–3735, 2019.

[32] Q. Wu and R. Zhang, "Common throughput maximization in uav-enabled ofdma systems with delay consideration," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6614–6627, 2018.

[33] S. Zhang, H. Zhang, B. Di, and L. Song, "Cellular uav-to-x communications: Design and optimization for multi-uav networks," *IEEE*

Transactions on Wireless Communications, vol. 18, no. 2, pp. 1346–1359, 2019.
[34] J. Chen, X. Cao, P. Yang, M. Xiao, S. Ren, Z. Zhao, and D. O. Wu, "Deep reinforcement learning based resource allocation in multi-uav-aided mec networks," *IEEE Transactions on Communications*, vol. 71, no. 1, pp. 296–309, 2023.
[35] X. Fan, M. Liu, Y. Chen, S. Sun, Z. Li, and X. Guo, "Ris-assisted uav for fresh data collection in 3d urban environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 632–647, 2023.
[36] F. Marini and B. Walczak, "Particle swarm optimization (pso). a tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 153–165, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743915002117

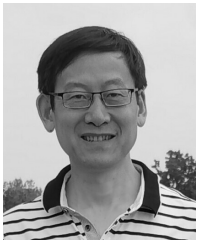
Qiang Tang received the B.E., M.S., and Ph.D. degrees in control science and engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005, 2007, and 2010, respectively. He is an academic visitor sponsored by CSC in University of Essex during 2016-2017. He is currently a Lecturer with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, China. His research interests include wireless networks, mobile edge computing, and smart grid.

Bao Li received the B.E. degree in Mechatronic Engineering from the Hunan Agricultural University, Changsha, China, in 2022. He is currently a graduate student at Changsha University of Science and Technology, Changsha, China. His current reserch interests include mobile edge computing and wireless UAV communication network.

Halvin H. Yang received the B.Eng. degree in Electronic and Electrical Engineering from Imperial College London in 2020. He is currently doing a PhD degree for Electronic and Electrical Engineering in University College London starting in 2020. His research interests include resource allocation for edge communications and performance analysis of new 6G technologies like intelligent reflective surfaces (IRS) and fluid antenna communication systems (FACS).

Yan Li received the B.S. degree in electronic information engineering from Hunan Normal University, Changsha, China, in 2013, and the M.S. degree in electronics and communication engineering and the Ph.D. degree in information and communication engineering from Sun Yat-sen University, Guangzhou, China, in 2016 and 2021, respectively.She is currently a Lecturer with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha. Her research interests include modeling and analysis of cellular networks and cooperative communications based on stochastic geometry theory.

Shiming He received the B.S. degree in information security and the Ph.D. degree in computer science and technology from Hunan University, Changsha, China, in 2006 and 2013, respectively. She is currently an Associated Professor with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha. Her research interests include machine learning, data analysis, and anomaly detection. Email:smhe_cs@csust.edu.cn



Kun Yang (Fellow IEEE) received his PhD from the Department of Electronic & Electrical Engineering of University College London (UCL), UK in 2005. He is currently a Chair Professor with the School of Computer Science & Electronic Engineering, University of Essex, leading the Network Convergence Laboratory (NCL), U.K. He is also an affiliated Professor with Nanjing University, China. Before joining in the University of Essex at 2003, he worked at UCL on several European Union (EU) research projects for several years. He has published more than 400 papers and filed 30 patents. His main research interests include wireless networks and communications, IoT networking, data and energy integrated networks and mobile computing. Prof. Yang manages research projects funded by various sources such as U.K. EPSRC, EU FP7/H2020 and industries. He serves on the editorial boards of both IEEE (e.g., IEEE TNSE, IEEE ComMag, IEEE WCL) and non-IEEE journals (e.g., Deputy EiC of IET Smart Cities). He was an IEEE ComSoc Distinguished Lecturer (2020-2021). He is a Member of Academia Europaea (MAE), a Fellow of IEEE, a Fellow of IET and a Distinguished Member of ACM.