Multi-UAV-Assisted Offloading for Joint Optimization of Energy Consumption and Latency in Mobile Edge Computing

Qiang Tang, Sihao Wen, Shiming He, and Kun Yang, *Fellow, IEEE*

***Abstract*—To address the performance limitations caused by the insufficient computing capacity and energy of edge internet of things devices (IoTDs), we proposed a multi-unmanned aerial vehicles (UAV)-assisted mobile edge computing (MEC) application framework in this article. In this framework, UAVs equipped with high-performance computing devices act as aerial servers deployed in the target area to support data offloading and task computing for IoTDs. We formulated an optimization problem to jointly optimize the connection scheduling, computing resource allocation, and UAVs' flying trajectories, considering the device offloading priority, to achieve a joint optimization of energy consumption and latency for all IoTDs during a given time period. Subsequently, to address this problem, we employed deep reinforcement learning for dynamic trajectory planning, supplemented by optimization theory and heuristic algorithm based on matching theory to assist in solving connection scheduling and computing resource allocation. To evaluate the performance of proposed algorithm, we compared it with deep deterministic policy gradient, particle swarm optimization, random moving, and local execution schemes. Simulation results demonstrated that the multi-UAV-assisted MEC significantly reduces the computing cost of IoTDs. Moreover, our proposed solution exhibited effectiveness in terms of convergence and optimization of computing costs compared to other benchmark schemes.**

***Index Terms*—Computing resource allocation, connection scheduling, deep reinforcement learning (DRL), mobile edge computing (MEC), trajectory optimization, unmanned aerial vehicles (UAVs).**

I. Introduction

THE rapid development of internet of thing (IoT) technologies has had a tremendous impact on people's lives and production. Although these various devices bring us more convenience, their performance improvement has been greatly hindered due to the limitation of computing resources and available energy.

Mobile edge computing (MEC) [1], a novel distributed computing paradigm, aims to deploy computing, storage, and network resources at the edge locations close to the end devices, in order to provide faster, more reliable, secure, and flexible service experience. It has a wide range of applications and is considered a forward-looking solution to address the performance limitations of internet of things devices (IoTDs) [2]. Meanwhile, unmanned aerial vehicles (UAVs) [3], with their maneuverability, portability, and height advantage, have garnered significant attention in the field of wireless communication when combined with MEC.

Compared with traditional ground edge nodes, UAVs as mobile edge nodes have three main advantages [4], [5]. First, UAVs can quickly and conveniently be deployed in target areas, breaking through the influence of terrain factors and height limitations that ground edge nodes face. Second, UAVs can dynamically adjust their coordinates, making it more suitable for real-time communication scenarios. Finally, UAVs can carry more powerful computing devices and sensors, that can process more data and more complex algorithms. In addition, compared to a single UAV, multiple UAVs can improve the overall performance and efficiency through collaboration. Therefore, the use of multiple UAVs is an important topic for research and exploration in the field of MEC.

As a powerful tool, computational intelligence (CI) [6], [7] has been widely applied in various disciplines and fields in recent years. Among them, deep reinforcement learning (DRL) constructs an understanding of the target environment using powerful deep neural networks in situations with limited or no prior knowledge. DRL is particularly suitable for addressing complex, challenging, and rapidly changing problems that require long-term planning. Currently, many researchers have started to model communication problems in MEC as Markov decision processes (MDPs) [8] and use DRL methods to solve them.

In simple terms, leveraging edge nodes with available computing resources to enhance the performance of edge networks has become a growing trend. Exploring the use of multiple UAVs as aerial servers to support MEC is a promising avenue for research. In addition, employing intelligent tools such as DRL to solve communication issues in MEC represents a novel and promising idea. Based on the above observations, we propose

Qiang Tang, Sihao Wen, and Shiming He are with the School of Computer Science & Communication Engineering, Changsha University of Science & Technology, Changsha 410114, China (e-mail: tangqiang@csust.edu.cn; sihaowen@stu.csust.edu.cn; smhe_cs@csust.edu.cn).

Kun Yang is with the School of Computer Technology and Engineering, Changchun Institute of Technology, Changchun 130012, China, and also with the School of Computer Science and Electronic Engineering, University of Essex, CO4 3SQ Colchester, U.K. (e-mail: kunyang@essex.ac.uk).

a DRL-based multi-UAV supported MEC architecture, where multiple UAVs equipped with high-performance computing devices assist IoTDs in offloading. Our goal is to minimize energy consumption and latency of all IoTDs while considering device offloading priority (DOP). In this case, it is crucial to study methods for connection scheduling, resource allocation and UAVs' trajectories.

II. Related Work

There are now some studies that have investigated issues related to UAV-based edge computing systems in depth. Based on the solving methods, we roughly classify these research works into two categories: those solved via optimization theory and those solved via DRL methods.

Hu et al. [9] studied an architecture in which UAVs and access points (APs) collaborate to assist MEC. The results show that using UAVs as computing servers to assist user equipment (UE) in carrying out their tasks, or as relays to further offload their computing tasks to APs, can greatly improve the computing performance of the system. Tang et al. [10] conducted research on a UAV-supported wireless power transfer and communication network. An alternate optimization algorithm is proposed and the idea of bottleneck awareness is employed to reduce problem complexity. Lin et al. [11] utilized a synergistic approach combining heuristic algorithms and optimization theory to reduce problem complexity and achieve the minimization of energy consumption and latency. Wang et al. [12] modeled UAV trajectory as a traveling salesman problem, significantly reducing UAV energy consumption while achieving appropriate load balancing.

Looking back at these works [9], [10], [11], [12], we found that they mostly focused on the placement of a single UAV, and generally used optimization theory or metaheuristic algorithms to solve the problem. Although research on algorithms [13], [14] such as block coordinate descent and successive convex approximation has made it possible for nonconvex problems to converge and be solved through multiple iterations, there is undoubtedly these methods may have some impact on solution accuracy.

Abegaz et al. [15] introduced how to utilize a cluster of multiple UAVs to provide services for IoTDs. Its results show that compared to heuristic algorithms, the DRL-based method has significant advantages in many aspects. Wang et al. [16] proposed a multi-UAV trajectory control algorithm based on DRL. Simulation results show that, the DRL-based algorithm can quickly adapt to different takeoff points of UAVs. In addition, Gao et al. [17] combined game theory with multiagent deep deterministic policy gradient (MADDPG) to simultaneously achieve obstacle avoidance and minimize offloading latency. Li et al. [18] effectively tackled the problem of connection scheduling and UAV trajectory using the double DQN (DDQN). Qian et al. [19] conducted experiments using DRL under different channel conditions, demonstrating the advantages of non-orthgonal multiple access (NOMA)-assisted multitasking over traditional orthogonal multiple access schemes. Guo et al. [20] took into account the data security issue in eavesdropping environments while achieving a joint optimization of system energy consumption and latency.

We have found that DRL-based optimization methods are less sensitive to the convexity of mathematical models and constraints of problems, and exhibit stronger adaptability, robustness, scalability, and efficiency. This provides us with more possibilities to solve complex optimization problems in wireless communication scenarios, making it a promising solution for issues such as dynamic trajectory control of UAVs.

The main contributions of this article can be summarized as follows.

1) A framework for multi-UAV-assisted MEC was proposed, which involves deploying multiple UAVs with a certain coverage range as airborne servers in edge IoT networks to assist in offloading and computing data tasks for IoTDs. Considering the vast number and diverse distribution of IoTDs, we achieve a joint optimization of energy consumption and latency of IoTDs by optimizing the dynamic trajectories of UAVs, connection scheduling, and allocation of computing resources through collaborative optimization in the environment.
2) Given the heterogeneity of resources, we introduced the concept of DOP to differentiate the various computing task attributes generated by different IoTDs. The value of DOP is primarily related to the confidentiality, timeliness, and data volume of the tasks generated by the device. Simulation results have demonstrated that IoTDs with a high degree of DOP are given priority when it comes to offloading.
3) To tackle the complex mixed integer nonlinear problems (MINLP), we have developed a dynamic trajectory control scheme based on DRL. At the same time, we incorporated convex optimization theory and heuristic algorithm based on matching theory to assist in solving connection scheduling and computing resource allocation.
4) We compared DDPG, particle swarm optimization (PSO), random moving (RM), and local execution (LE) schemes through extensive simulations, evaluating the algorithm performance in terms of convergence, trajectory, UAVs' workload, computing overall cost, energy cost, and delay cost. The simulation results showed that our proposed algorithm had significant advantages in many aspects of performance compared to the other comparison schemes.

The rest of this article is organized as follows. Section III describes the system model in detail and formulates the related problems. Section IV simplifies the problems and introduces the DRL-based solution method. Section V presents the experimental results and numerical analysis. Finally, Section VI concludes this article.

III. System Model

As illustrated in Fig. 1, we introduce a MEC system assisted by multiple UAVs. The system consists of N UAVs and K IoTDs, all equipped with communication circuits and computing

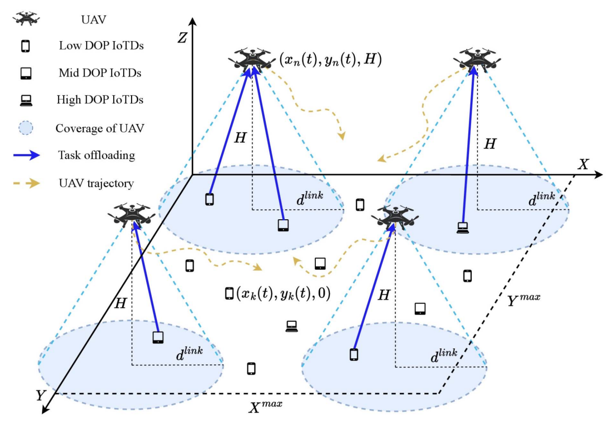

Fig. 1. System model.

TABLE I
LIST OF SYMBOLS

Parameter	Meaning
$n, N, \mathcal{N}$	index,number,set of UAVs
$k, K, \mathcal{K}$	index,number,set of IoTDs
$t, T, \mathcal{T}$	index,number,set of time slots
τ	time slot size
$c_{k,n}$	connection between IoTD k and UAV n
$C^{\max}$	maximal number of IoTDs connected per UAV
$X^{\max}, Y^{\max}$	length, width of task area
$(x_k, y_k, 0)$	coordinate of IoTD k
$\theta_n(t), v_n(t)$	flying angle and speed of UAV n in slot t
$(x_n(t), y_n(t), H)$	coordinate of UAV n in slot t
$v^{\max}$	UAVs' maximal flying speed
d^{safe}	minimal safe distance among UAVs
d^{link}	horizontal coverage radius of UAVs
$d_{k,n}^{IU}(t)$	distance between IoTD k and UAV n in slot t
κ_k, κ_n	effective capacitance coefficient of IoTDs and UAVs
δ^2	power of additive white Gaussian noise
B, P^{off}	channel bandwidth, transmitting power
$D_k(t), T_k(t)$	task size and tolerable delay of IoTD k in slot t
P_k	device offloading priority of IoTD k
$f_k^{\max}, f_n^{\max}$	maximal computing frequency of IoTDs and UAVs
$s(t), a(t), r(t)$	state, action, reward
$\pi(\cdot), Q(\cdot), L(\cdot)$	policy function, Q function, loss function
$\theta, \omega_1, \omega_2$	parameter of Actor and Critic network

processors. The DOP of IoTDs, which had been broadcasted before the system optimization took place, can be divided into three levels: high, medium, and low. We assume that time slot size is τ, and during a continuous duration of T time slots, each IoTD generates a randomly sized computing task in every time slot. These tasks can be computed locally by the IoTD or offloaded to a UAV for computation. Meanwhile, UAVs communicate with multiple IoTDs within their coverage simultaneously, using orthogonal frequency division multiplexing [10], to assist with task offloading and computing.

We introduce $\mathcal{N} = \{1, 2, \ldots, N\}$ to present the collection of UAVs, $\mathcal{K} = \{1, 2, \ldots, K\}$ to denote the group of IoTDs, $\mathcal{T} = \{1, 2, \ldots, T\}$ to represent the set of time slots. For clarification of the important symbols used in this article, detailed explanations are provided in Table I.

It is worth noting that each IoTD can only be connected to at most one UAV in each time slot due to the single antenna configuration. Additionally, each UAV can be connected to a maximum of $C^{\max}$ IoTDs in each time slot due to the limited number of antennas. Therefore, we have the following constraints for connection scheduling in each time slot:

$$c_{k,n}(t) \in \{0, 1\}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}', \forall t \in \mathcal{T} \tag{1}$$

$$\sum\nolimits_{n=0}^{N} c_{k,n}(t) = 1, \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \tag{2}$$

$$\sum\nolimits_{n=1}^{K} c_{k,n}(t) \leq C^{\max}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{3}$$

where $\mathcal{N}' = \{0, 1, \ldots, N\}$. It should be noted $c_{k,0}(t) = 1$ means that in time slot t, the kth IoTD performs local computing without any offloading, while $c_{k,n}(t) = 1, n \neq 0$ suggests that in time slot t, the kth IoTD offloads its data tasks to the nth UAV for collaborative computing.

A. Mobility Model

We use a 3-D Euclidean coordinate system [9] with units in meters. The task area is a rectangular region with a length of $X^{\max}$ and a width of $Y^{\max}$.

Assuming that the coordinates of all IoTDs are fixed and randomly distributed on the ground within the task area, the location of the kth IoTD is represented by $\boldsymbol{q_k^I} = (x_k, y_k, 0), k \in \mathcal{K}$.

All UAVs are deployed in the airspace above the task area at a fixed altitude of $H > 0$, which is adapted to the task terrain to avoid frequent descending and ascending of UAVs during the period of task performance. The initial coordinate of the nth UAV is represented by $\boldsymbol{q_n^U(0)} = (x_n(0), y_n(0), H), n \in \mathcal{N}$.

To describe the motion of UAVs, we introduce the flying angle

$$0 \leq \theta_n(t) \leq 2\pi, n \in \mathcal{N}, t \in \mathcal{T} \tag{4}$$

as well as the flying speed

$$0 \leq v_n(t) \leq v^{\max}, n \in \mathcal{N}, t \in \mathcal{T}. \tag{5}$$

Here, $\theta_n(0)$ and $v_n(0)$ are the initial flying angle and speed of the nth UAV, respectively, and $v^{\max}$ is the maximum flying speed that UAV can achieve. Let $\Delta\theta_n(t)$ and $\Delta v_n(t)$ denote the changes in angle and speed for the nth UAV in the tth time slot, then its flying angle and speed for $\forall n \in \mathcal{N}, \forall t \in \mathcal{T}$ can be expressed as $\theta_n(t) = \theta_n(t-1) + \Delta\theta_n(t), v_n(t) = v_n(t-1) + \Delta v_n(t)$.

Based on the above information, we can represent the coordinate of the nth UAV at time slot t as $\boldsymbol{q_n^U(t)} = (x_n(t), y_n(t), H), \forall n \in \mathcal{N}, \forall t \in \mathcal{T}$, where

$$\begin{cases} x_n(t) = x_n(t-1) + v_n(t) \sin\theta_n(t)\tau \\ y_n(t) = y_n(t-1) + v_n(t) \cos\theta_n(t)\tau \end{cases}. \tag{6}$$

Each UAV is restricted to move within the task area, thus for $n \in \mathcal{N}, t \in \mathcal{T}$, the following constraints are obtained:

$$0 \leq x_n(t) \leq X^{\max} \tag{7}$$

$$0 \leq y_n(t) \leq Y^{\max}. \tag{8}$$

Furthermore, each UAV must keep a certain distance from other UAVs to avoid collisions. Therefore, for $\forall i, j \in \mathcal{N}, i \neq$

$j, \forall t \in \mathcal{T}$, we have

$$d_{i,j}^{UU}(t) = \sqrt{\left\| \boldsymbol{q_i^U(t)} - \boldsymbol{q_j^U(t)} \right\|^2} \geq d^{\text{safe}} \tag{9}$$

where d^{safe} represents the minimum distance that each UAV must maintain to ensure safe flying.

B. Communication Model

As the altitude of UAVs remains fixed during flying, there is no need to consider their elevation changes. Therefore, we assume that the coverage range of UAVs is a circular area centered on their location with a radius of d^{link}. In other words, during the tth time slot, if the horizontal distance $d_{k,n}^G(t) = \sqrt{(x_n(t) - x_k)^2 + (y_n(t) - y_k)^2}$ between the kth IoTD and the nth UAV satisfies the condition that $d_{k,n}^G(t) \leq d^{\text{link}}$, then the kth IoTD will be within the coverage range of the nth UAV in the tth time slot. Taking into account connection scheduling, we have

$$c_{k,n}(t) d_{k,n}^G(t) \leq d^{\text{link}}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{10}$$

We assume the adoption of millimeter-wave communication between UAVs and UEs. According to [21], the channel gain between them in time slot t can be expressed as

$$h_{k,n}(t) = h_{k,n}^l(t) h^f h^m \tag{11}$$

where $h_{k,n}^l(t)$ represents path loss, h^f denotes multipath fading, and h^m corresponds to misalignment fading.

1) *Path loss:* $h_{k,n}^l(t)$ primarily consists of propagation loss and molecular absorption: $h_{k,n}^l(t) = h_{k,n}^{lp}(t) h_{k,n}^{la}(t)$. According to Friis equation, we have $h_{k,n}^{lp}(t) = \frac{c\sqrt{G_t G_r}}{4\pi f d_{k,n}^{IU}(t)}$. Here, c represents the speed of light, G_t and G_r are the transmit and receive gains, respectively, f denotes the frequency band occupied by millimeter waves, and $d_{k,n}^{IU}(t) = \sqrt{(d_{k,n}^G(t))^2 + H^2}$ is the distance between the nth UAV and the kth IoTD in the tth time slot. Additionally, $h_{k,n}^{la}(t) = e^{-\frac{1}{2}\kappa_\alpha(f) d_{k,n}^{IU}(t)}$, where $\kappa_\alpha(f)$ represents the absorption coefficient describing the relative area per unit of volume. Please refer to [21] for its computing details.
2) *Multipath fading:* Rayleigh distribution is commonly used to describe the probability characteristics of multipath fading in wireless communication. As referred in [21], its probability density function can be expressed as: $f_{h^f} = \frac{2x}{(\widehat{h^f})^2 \Gamma(1)} e^{-\frac{x^2}{(\widehat{h^f})^2}}$. Here, $\widehat{h^f}$ denotes the fading channel envelop with root mean value $\epsilon = 2$.
3) *Misalignment fading:* Similar to [21], we assume r represents the radius of the IoTD detection beam, while $R (0 \leq R \leq R_m)$ denotes the radius of the UAV beam. The maximum radius of the beam within the UAV coverage area is denoted as R_m, and the pointing error between the UAV beam center O_{UAV} and the IoTDs beam center O_{IoTD} is represented by l. Based on [22], we can derive $h_m = P_0 e^{-\frac{2l^2}{R_q^2}}$. Here, P_0 represents the received power when $l = 0$, and R_q is the equivalent beamwidth. Referring to [22], we have $P_0 = erf(\zeta)^2, R_q = \frac{\sqrt{\pi} erf(\zeta)}{2\zeta e^{-\zeta^2}} R_m^2$, where $erf(\cdot)$ denotes the Gaussian error function, and $\zeta = \frac{\sqrt{\pi} r}{\sqrt{2} R_m}$.

Therefore, the corresponding signal-to-noise ratio received from the kth IoTD to the nth UAV can be modeled as

$$\gamma_{k,n}(t) = \frac{P^{\text{off}} h_{k,n}(t)}{\delta^2} \tag{12}$$

where P^{off} is the transmission power of IoTDs, and δ^2 is the power of Additive White Gaussian Noise. According to Shannon's formula [9], [23], if B is the bandwidth of each communication channel, the achievable offloading rate $R_{k,n}(t)$ (bits/second) from the kth IoTD to the nth UAV in the tth time slot can be expressed as

$$R_{k,n}(t) = B \log_2(1 + \gamma_{k,n}). \tag{13}$$

C. Computing Task Model

The computing task generated by the kth IoTD in the tth time slot can be donated as $S_k(t) = \{D_k(t), T_k(t)\}$, where $D_k(t)$ represents the data size of the task (in bits); $T_k(t)$ means the maximum tolerable delay (in seconds) for the task to be executed. In this article, for the sake of convenience in computation and expression, similar to [9], [16], we set $T_k(t) = \tau$ $(\forall k \in \mathcal{K}, \forall t \in \mathcal{T})$ uniformly. It is worth noting that due to the small size of the computing result, the time and resources required for the computing result to be transmitted back are not considered [17], [18].

1) Data Computed Locally at the IoTDs: Let $f_k^{\text{com}}(t)$ (cycles/second) present the CPU frequency and $t_k^{\text{com}}(t)$(second) denote the computing time of the kth IoTD during the tth time slot. Then its local computing time is $t_k^{\text{com}}(t) = C_k \frac{D_k(t)}{f_k^{\text{com}}(t)}$, where C_k (cycles/bit) represents the amount of CPU cycles required by the kth IoTD to process 1 b of task data. For $\forall k \in \mathcal{K}, \forall t \in \mathcal{T}$, we can obtain expression

$$0 \leq t_k^{\text{com}}(t) \leq T_k(t) \tag{14}$$
$$0 \leq f_k^{\text{com}}(t) \leq f_k^{\max}. \tag{15}$$

Similar to [24], for $\forall k \in \mathcal{K}, \forall t \in \mathcal{T}$, the energy consumption of the kth IoTD for task computing in the tth time slot can be expressed as $e_k^{\text{com}}(t) = t_k^{\text{com}}(t) \kappa_k (f_k^{\text{com}}(t))^3$, where κ_k is the effective capacitance coefficient of the $k - thIoTD$.

2) Data Computed Remotely at the UAVs by Offloading: We employ a full offloading approach, whereby when the kth IoTD decides to offload in the t-th time slot, it incurs only offloading energy without any computing energy in that slot. We can obtain the time, $t_{k,n}^{\text{off}}(t) = \frac{D_k(t)}{R_{k,n}(t)}$, it takes for offloading. By referring to [10], [25], we can then derive the corresponding offloading energy consumption $e_{k,n}^{\text{off}}(t) = t_{k,n}^{\text{off}}(t) P^{\text{off}}$.

We assume that UAVs can perform parallel computing, processing data offloaded by multiple IoTDs simultaneously. Let $f_{k,n}^{\text{com}}(t)$(cycles/second) and $t_{k,n}^{\text{com}}(t)$(second) denote the CPU frequency and computing time allocated by the nth UAV to the kth IoTD in the tth time slot, respectively. Then, we have $t_{k,n}^{\text{com}}(t) = C_n \frac{D_k(t)}{f_{k,n}^{\text{com}}(t)}$.

It is important to note that the sum of offloading time $t_{k,n}^{\text{off}}(t)$ and computing time $t_{k,n}^{\text{com}}(t)$ should not exceed the maximum tolerable delay of the task $T_k(t)$. In addition, the sum of the CPU frequencies allocated by UAV to IoTDs cannot exceed the maximum CPU frequency of UAV. Therefore, for $\forall n \in \mathcal{N}, \forall t \in \mathcal{T}$, we have the following constraint:

$$0 \le t_{k,n}^{\text{com}}(t) + t_{k,n}^{\text{off}}(t) \le T_k(t), \forall k \in \mathcal{K} \tag{16}$$

$$0 \le f_{k,n}^{\text{com}}(t), \forall k \in \mathcal{K} \tag{17}$$

$$\sum\nolimits_{k=1}^{K} c_{k,n}(t) f_{k,n}^{\text{com}}(t) \le f_n^{\max}. \tag{18}$$

D. Problem Formulation

Our objective is to minimize the overall energy consumption and task execution delay of all IoTDs, taking into account the existence of DOP. Therefore, we can incorporate DOP as a weight for local computing energy consumption and task execution delay in the objective function, i.e., the larger the DOP of an IoTD, the higher its local computing cost, and the more inclined it is to offload. In this article, we use P_k to represent the DOP value of the kth IoTD.

According to [11], [20], we denote the local computing cost of the kth IoTD in the tth time slot as

$$\psi_k^{\text{local}}(t) = P_k(e_k^{\text{com}}(t) + t_k^{\text{com}}(t)) \tag{19}$$

and the cost of offloading the kth IoTD's computing task to the nth UAV in the tth time slot is

$$\psi_{k,n}^{\text{off}}(t) = e_{k,n}^{\text{off}}(t) + t_{k,n}^{\text{off}}(t) + t_{k,n}^{\text{com}}(t). \tag{20}$$

Thus, for $\forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}$, the cost of the kth IoTD completing its task in the tth time slot can be expressed as

$$\psi_{k,n}(t) = c_{k,0}(t)\psi_k^{\text{local}}(t) + c_{k,n}(t)\psi_{k,n}^{\text{off}}(t). \tag{21}$$

Let $C = \{c_{k,n}(t), \forall k \in \mathcal{K}, \forall n \in \mathcal{N}', \forall t \in \mathcal{T}\}$, $F = \{f_k^{\text{com}}(t), f_{k,n}^{\text{com}}(t), \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}\}$, $Q = \{\theta_n(t), v_n(t), \forall n \in \mathcal{N}, \forall t \in \mathcal{T}\}$, then in mathematics, our optimization problem can be expressed as

$$(\mathbf{P1}): \min_{C,F,Q} \quad \sum\nolimits_{t=1}^{T} \sum\nolimits_{n=0}^{N} \sum\nolimits_{k=1}^{K} \psi_{k,n}(t) \tag{22}$$

s.t.

$$c_{k,n}(t) \in \{0,1\}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}', \forall t \in \mathcal{T} \tag{22a}$$

$$\sum\nolimits_{n=0}^{N} c_{k,n}(t) = 1, \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \tag{22b}$$

$$\sum\nolimits_{n=1}^{K} c_{k,n}(t) \le C^{\max}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{22c}$$

$$0 \le \theta_n(t) \le 2\pi, n \in \mathcal{N}, t \in \mathcal{T} \tag{22d}$$

$$0 \le v_n(t) \le v^{\max}, n \in \mathcal{N}, t \in \mathcal{T} \tag{22e}$$

$$0 \le x_n(t) \le X^{\max}, n \in \mathcal{N}, t \in \mathcal{T} \tag{22f}$$

$$0 \le y_n(t) \le Y^{\max}, n \in \mathcal{N}, t \in \mathcal{T} \tag{22g}$$

$$d_{i,j}^{UU}(t) \ge d_{\text{safe}}, \forall i, j \in \mathcal{N}, i \ne j, \forall t \in \mathcal{T} \tag{22h}$$

$$c_{k,n}(t) d_{k,n}^{G}(t) \le d^{\text{link}}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{22i}$$

$$0 \le t_k^{\text{com}}(t) \le T_k(t), \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \tag{22j}$$

$$0 \le t_{k,n}^{\text{com}}(t) + t_{k,n}^{\text{off}}(t) \le T_k(t), \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{22k}$$

$$0 \le f_k^{\text{com}}(t) \le f_k^{\max}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \tag{22l}$$

$$0 \le f_{k,n}^{\text{com}}(t), \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{22m}$$

$$\sum\nolimits_{k=1}^{K} c_{k,n}(t) f_{k,n}^{\text{com}}(t) \le f_n^{\max}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{22n}$$

IV. Algorithms Design

It is evident that problem (*P1*) is not only composed of continuous variables F and Q, but also includes discrete variable A, making it a MINLP that is difficult to solve directly. Therefore, in this section, we employed DRL for dynamic trajectory planning, supplemented by optimization theory and heuristic algorithm based on matching theory to assist in solving connection scheduling and computing resource allocation.

A. Preliminary Work

1) DRL Background Knowledge: DRL uses deep neural networks to approximate value or policy functions, which enables adaptive learning and generalization. It has demonstrated good performance in solving practical problems with high-dimensional and continuous state and action spaces.

In DRL, the agent starts from a current state s_t, selects an action a_t from the action space based on the policy π, and executes it. Then, the agent observes a new state s_{t+1} and obtains a reward r_t. This process is repeated until a terminal state is reached or a certain time step T is exceeded. During this process, the agent aims to maximize the cumulative reward, and therefore our objective is to find the policy $\pi^* = \arg\max_\pi \mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t]$, that maximizes the expected reward. In this equation, $\gamma \in (0,1)$ is the discount factor.

SAC [26], whose main idea is to use soft Q-learning in the actor–critic framework to learn the policy, and employ an entropy regularization term to promote exploration, is a commonly used algorithm in DRL. In addition to maximizing cumulative rewards, it also aims to make the policy more stochastic. We use entropy $H(\pi(\cdot|s_t)) = \mathbb{E}_\pi[-\log \pi(\cdot|s_t)]$ to measure the level of randomness of the policy π. Therefore, the optimal policy of SAC can be donated as $\pi^* = \arg\max_\pi \mathbb{E}[\sum_{t=0}^{T} r_t + \alpha H(\pi(\cdot|s_t))]$. Here, α is a regularization coefficient used to control the importance of the entropy, and its loss function can be represented as

$$L(\alpha) = \mathbb{E}\left[-\alpha \log(a_t'|s_t) - \alpha H_0\right] \tag{23}$$

where a_t' represents the action obtained by inputting s_t into the actor network according to the current policy, rather than sampled from the replay buffer.

Based on the idea of DDQN, SAC uses two Q-networks, but only selects the one with lower Q-value during each use in order to alleviate the problem of overestimation of Q-values. Let's first review the Soft Bellman equation for the action-value function $Q(s_t, a_t) = r_t + \gamma\mathbb{E}[V(s_{t+1})]$, where the state-value function is represented as $V(s_t) = \mathbb{E}[Q(s_t, a_t) + H(\pi(\cdot|s_t))]$. Then, we can define the target value of the critic network as

$$y_t = r_t + \gamma\left(\min_{i=1,2} Q_{\omega_i^-}(s_{t+1}, a'_{t+1}) - \alpha\log\pi_\theta(\cdot|s_{t+1})\right), \tag{24}$$

where ω_i^- is the parameter of the target Q-network.

Both two critic networks are updated by minimizing their loss function to learn the Q-value function, which can be expressed as

$$L_Q(\omega_i) = \mathbb{E}\left[\frac{1}{2}\left(Q_{\omega_i}(s_t, a_t) - \left(r_t + \gamma V_{\omega_i^-}(s_{t+1})\right)\right)^2\right]. \tag{25}$$

The actor network, on the other hand, is updated by maximizing the Soft Q function to learn the policy π_θ, and its loss function is obtained from the Kullback–Leibler divergence and can be simplified to

$$L_\pi(\theta) = \mathbb{E}\left[\alpha\log(\pi_\theta(a'_t|s_t)) - Q_{\omega_i}(s_t, a'_t)\right]. \tag{26}$$

2) Computing Resource Allocation: Since $C_k D_k(t)$ is known at the t-th time slot, it is obvious that $\psi_k^{\text{local}}(t)$ is a convex function with respect to $t_k^{\text{com}}(t)$. We first take its derivative to obtain its slope

$$\frac{d\psi_k^{\text{local}}(t)}{dt_k^{\text{com}}(t)} = P_k\left[1 - 2\kappa_k\frac{(C_k D_k(t))^3}{(t_k^{\text{com}}(t))^3}\right] \tag{27}$$

and then obtain its stationary point $t_k^{\text{best}}(t) = C_k D_k(t)\sqrt[3]{2\kappa_k}$.

It is evident that $\psi_k^{\text{local}}(t)$ is monotonically non-increasing over interval $t_k^{\text{com}}(t) \in [0, t_k^{beast}(t)]$ and monotonically nondecreasing over interval $t_k^{\text{com}}(t) \in [t_k^{\text{best}}(t), +\infty]$. Therefore, $t_k^{\text{best}}(t)$ represents the optimal local computing time of the kth IoTD in the tth time slot without considering constraints.

Based on constraint (22l), we can obtain the minimal time required for the kth IoTD to complete local computation in the tth time slot as $t_k^{\min}(t) = \frac{C_k D_k(t)}{f_k^{\max}}$. In conjunction with constraint (22k) that limits the task completion time, we can determine the optimal local computing time for the kth IoTD in the tth slot as

$$t_k^{\text{com}}(t)^* = \begin{cases} t_k^{\text{best}}(t), & \text{if } t_k^{\min}(t) \le t_k^{\text{best}}(t) \le T_k(t) \\ t_k^{\min}(t), & \text{if } t_k^{\text{best}}(t) \le t_k^{\min}(t) \le T_k(t) \\ T_k(t), & \text{if } t_k^{\min}(t) \le T_k(t) \le t_k^{\text{best}}(t). \end{cases} \tag{28}$$

Then, we can obtain the optimal local CPU frequency as $f_k^{\text{com}}(t)^* = \frac{C_k D_k(t)}{t_k^{\text{com}}(t)^*}$.

In the tth time slot, given the connection schedule $c_{k,n}(t)$ ($\forall k \in \mathcal{K}, \forall n \in \mathcal{N}'$) and the coordinates $\boldsymbol{q_n^U(t)}$ ($\forall n \in \mathcal{N}$) of UAVs, the channel gain $h_{k,n}(t)$ between the kth IoTD and the nth UAV can be determined, and thus the offloading time $t_{k,n}^{\text{off}}(t)$ can be considered as known. At this point, the size of $\psi_{k,n}^{\text{off}}(t)$ is only related to $f_{k,n}^{\text{com}}(t)$, and we only need to find the optimal CPU frequency allocated to the kth IoTD by the nth UAV to obtain the minimal offloading cost in the tth time slot for the kth IoTD.

Lemma 1: Based on the connection schedule, we can obtain the set of IoTDs, denoted as $\mathcal{M}_n = \{1_n, 2_n, \ldots, k_n, \ldots, M_n\}$, that the nth UAV is connected to during the tth time slot. Then, in order to minimize the total offloading cost of these IoTDs, the optimal CPU frequency allocated to each IoTD can be denoted as

$$f_{k_n,n}^{\text{com}}(t)^* = f_n^{\max}\frac{\sqrt{C_n D_{k_n}(t)}}{\sum_{m_n=1_n}^{M_n}\sqrt{C_n D_{m_n}(t)}}. \tag{29}$$

Proof: For the nth UAV, the total offloading cost of all the connected IoTDs during the tth time slot can be denoted as

$$\sum_{m_n=1_n}^{M_n}\psi_{m_n,n}^{\text{off}}(t) = \sum_{m_n=1_n}^{M_n}\left(\underbrace{t_{m_n,n}^{\text{off}}(t)\left(1+P^{\text{off}}\right)}_{\text{known}} + C_n\frac{D_k(t)}{f_{m_n,n}^{\text{com}}(t)}\right). \tag{30}$$

Combining constraints (22l) and (22m), we can formulate this problem as

$$(\mathbf{P1.1}):\ \min_f \quad \sum_{m_n=1_n}^{M_n}\left(C_{m_n} + \frac{D_{m_n}}{f_{m_n}}\right) \tag{31}$$

$$s.t.\quad 0 \le f_{m_n}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}, \tag{31a}$$

$$\sum_{m_n=1_n}^{M_n} f_{m_n} \le f_n^{\max}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}, \tag{31b}$$

where $C_{m_n} = t_{m,n}^{\text{off}}(t)(1 + P^{\text{off}}),\ D_{m_n} = C_n D_k(t),\ f_{m_n} = f_{m,n}^{\text{com}}(t)$. If we ignore the constraint (31a) first, then the Lagrangian dual function of (31) can be expressed

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{\lambda}) = \sum_{m_n=1_n}^{M_n}\left(C_{m_n} + \frac{D_{m_n}}{f_{m_n}}\right) + \lambda\left(\sum_{ma_n=1_n}^{M_n} f_{m_n} - f_n^{\max}\right), \tag{32}$$

where $\lambda(\lambda \ne 0)$ is the Lagrange multiplier. Based on the complementary relaxation condition and stability condition in the Karush–Kuhn–Tucker (KKT) condition [27], we have $f_{m_n}^* = f_n^{\max}\frac{\sqrt{D_{m_n}}}{\sum_{m_n=1_n}^{M_n}\sqrt{D_{m_n}}}, \forall m_n \in \mathcal{M}_n$.

Obviously, for $\forall m_n \in \mathcal{M}_n, 0 \le f_{m_n}^*$ must be established, so the constraint (31a) must is undoubtedly satisfied. Therefore, the *Lemma 1* is proved. ■

3) Connection Scheduling: We introduce a heuristic algorithm based on matching theory for solving the connection scheduling problem given the coordinates of UAVs. The details are presented in Algorithm 1.

As demonstrated in lines 1–23, we first initialize all IoTDs to be computed locally (line 1). Next, we iterate through all IoTDs and check their coverage by UAVs (lines 5–10). If the kth IoTD is not covered by any UAV, its task can only be computed locally. If the kth IoTD is covered by only one UAV, we tentatively assign the task to that UAV (lines 1119), as usually, the local computing cost of IoTDs is much higher than the offloading cost, whether in terms of energy consumption or task execution delay. If the kth IoTD is covered by multiple UAVs, we temporarily record its ID in the set MI and determine its connection scheduling after completing one round of traversal (lines 20–22).

For each IoTD in set MI (lines 24–33), whether it is from the perspective of load balancing of multiple UAVs or in order to obtain more computing resources, it is preferred to choose a UAV with minimal total assistive computing load (MTACL) for offloading. If there are multiple UAVs connected to the IoTD with MTACL, then for better channel quality, it is more inclined to choose the UAV that is closer to it.

As shown in lines 14–18, every time we make a decision on IoTD scheduling, we need to check if constraint (22c) is satisfied. If not, we sort the IoTDs connected to the UAV and change the IoTD with minimal product of DOP and task size (MPDTS) to perform local computing. This is because the larger the task size of an IoTD, the more cost savings can be achieved by offloading the task for computing, and we also have to take into account the existence of DOP

We can quickly obtain a suboptimal solution for connection scheduling based on the coordinates of the UAVs using Algorithm 1. Although this solution may be not optimal, its low complexity ensures that it will not have a significant impact on the training time of DRL. Moreover, in our simulation, this algorithm has achieved good optimization results.

Algorithm 1: Connection Schedule Algorithm.

```
Input: $q_j^U(t), \forall j \in \mathcal{N}$;
Output: $c_{i,j}(t), \forall i \in \mathcal{K}, \forall j \in \mathcal{N}'$;
 1: Initialize $c$ with $c_{i,j}(t) = 0, \forall i \in \mathcal{K}, \forall j \in \mathcal{N}$,
    $c_{i,0}(t) = 1, \forall i \in \mathcal{K}$;
 2: Initialize $MI = \{\}, L_j^U = \{\}, \forall j \in \mathcal{N}$;
 3: for IoTD $i$ in $\mathcal{K}$ do
 4:   Initialize $L_i^I = \{\}$;
 5:   for UAV $j$ in $\mathcal{N}$ do
 6:     Calculate $d_{i,j}^G(t)$ according to $q_j^U(t)$
 7:     if $d_{i,j}^G \le d^{link}$ then
 8:       Store $[j, d_{i,j}^G(t)]$ into $L_i^I$;
 9:     end if
10:   end for
11:   if only one item $[l, d_{i,l}^G(t)]$ in $L_i^I$ then
12:     Store $[i, P_i, D_i(t)]$ into $L_l^U$;
13:     $c_{i,0}(t) = 0, c_{i,l}(t) = 1$;
14:     if the length of $L_l^U > C^{max}$ then
15:       Push $[m, P_m, D_m(t)]$ with MPDTS from $L_l^U$;
16:       Remove item $[l, d_{ml}^G(t)]$ from $L_m^I$;
17:       $c_{m,0}(t) = 1, c_{m,l}(t) = 0$;;
18:     end if
19:   end if
20:   if the length of $L_i^I > 1$ then
21:     Store i into $MI$;
22:   end if
23: end for
24: for IoTD $i$ in $MI$ do
25:   if multiple UAVs with MTACL in $L_i^I$ then
26:     j = the UAV with MTACL closest to IoTD $i$ in $L_i^I$
27:   else
28:     j = the UAV with MTACL in $L_i^I$
29:   end if
30:   Store $[i, P_i, D_i(t)]$ into $L_j^U$;
31:   $c_{i,0}(t) = 0, c_{i,j}(t) = 1$;
32:   Do steps 14–18 again for $L_j^U$
33: end for
34: return $c_{i,j}(t), \forall i \in \mathcal{K}, \forall j \in \mathcal{N}'$;
```

B. Overall Algorithm Design

In this section, we will design the SAC based MOC algorithm to solve problem *P1*. First, we define the state, action, and reward in DRL as follows:

1) *State:* $s_t = (x_n(t), y_n(t), \forall n \in \mathcal{N}\}$ is the agent's observation of the environment, where $x_n(t), y_n(t)$ are the abscissa and ordinate of the nth UAV respectively. The set of all possible states forms the state space.
2) *Action:* $a_t = \{\Delta\theta_n(t), \Delta v_n(t), \forall n \in \mathcal{N}\}$ describes the agent's behavior, where $\Delta\theta_n(t)$ and $\Delta v_n(t)$ are the increments of the UAVs' flying angle and speed, respectively. The set of all possible actions forms the action space.
3) *Reward:*
$$r_t = -\sum\nolimits_{n=0}^{N} \sum\nolimits_{k=1}^{K} \psi_{k,n}(t) - p(t) \tag{33}$$
evaluates the feedback that the agent receives when it executes action a_t in state s_t. As we aim to minimize cost while DRL maximizes reward, we take the negative of the cost here. The penalty signal $p(t)$ is imposed on the agent when it violates constraints.

According to [28], the algorithm architecture presented in this article is illustrated in Fig. 2, which consists of an actor network π_θ, two critic networks Q_{ω_1} and Q_{ω_2}, and two critic target networks $Q_{\omega_1^-}$ and $Q_{\omega_2^-}$. π_θ learns a policy function that maps the current state to an action. Q_{ω_1} and Q_{ω_2} learn two value functions that evaluate the goodness of the current state and action. The purpose of them is to reduce estimation errors and improve the stability of the algorithm. $Q_{\omega_1^-}$ and $Q_{\omega_2^-}$ provide a stable target Q value for calculating the temporal difference (TD) error of the critic networks, which in turn updates the policy of actor network. For reducing the magnitude of the target Q value changes and increasing the stability of the algorithm, the parameters of these two critic target networks are updated from the parameters of the critic networks through soft update

$$\left[\omega_i^- \leftarrow \lambda\omega_i + (1 - \lambda\omega_i^-)\right]_{i=1,2} \tag{34}$$

where λ is the soft update parameter. We describe the specific algorithm flow as the pseudocode shown in Algorithm 2.

First, the agent interacts with the environment to generate action a_t based on the current state s_t and policy π_θ. After executing a_t, the coordinates of all UAVs are updated, and the environment transitions from s_t to s_{t+1}. Based on the coordinates of the UAVs, we use Algorithm 1 to solve the connection schedule problem, and then solve the resource allocation problem based on the connection scheduling, resulting in a reward r_t. We then store the tuple (s_t, a_t, r_t, s_{t+1}) in the replay buffer. Once the number of tuples in the cache pool reaches the set minimal size, we can use mini-batch sampling to collect J tuples for updating the neural network. The specific update process is given in lines 14–21. As we have already discussed the relevant knowledge of SAC updates in the earlier section, we will not elaborate on it again here.

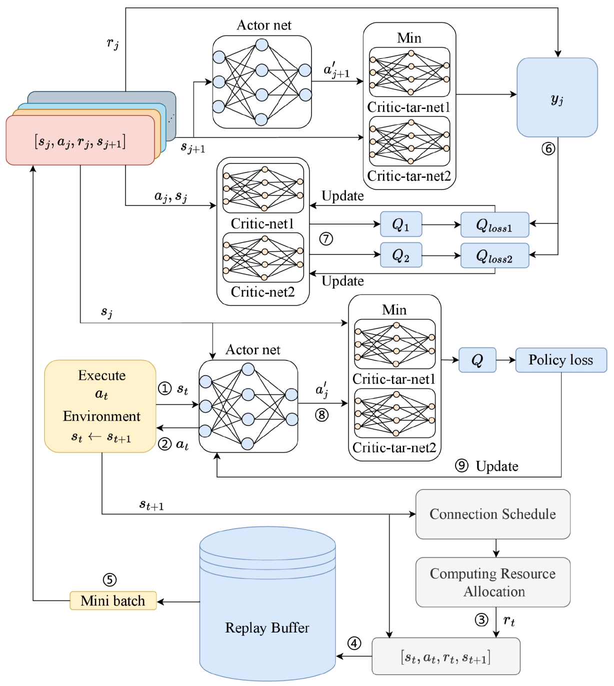

Fig. 2. Structure of MOC algorithm.

Algorithm 2: MOC Algorithm.

```
1:  Initialize Critic networks $Q_{\omega_1}$ and $Q_{\omega_2}$ with
    parameters $\omega_1$ and $\omega_2$, respectively, and Actor network
    $\pi_\theta$ with parameter $\theta$.
2:  Initialize target networks $Q_{\omega_1^-}$ and $Q_{\omega_2^-}$ by parameters
    $\omega_1^- \leftarrow \omega_1$ and $\omega_2^- \leftarrow \omega_2$;
3:  Initialize the experience replay buffer $R$;
4:  for epoch $e$ in $\mathcal{E}$ do
5:    Initialize the state of environment $s_t$;
6:    for time slot $t$ in $\mathcal{T}$ do
7:      Select an action based on current policy
        $a_t = \pi_\theta(s_t)$
8:      All UAVs execute $a_t$;
9:      Obtain new state $s_{t+1}$;
10:     Solve connection scheduling by Algorithm 1;
11:     Solve resource allocation by (28) and (29);
12:     Calculate reward $r_t$ by (33);
13:     Store $(s_t, a_t, r_t, s_{t+1})$ into $R$;
14:     if $min\_size$ elements in $R$ then
15:       Sample a mini-batch
          $\{(s_j, a_j, r_j, s_{j+1})\}_{j=\{1,\dots,J\}}$ from $R$;
16:       For each tuple, compute $y_j$ by (24) with target
          network;
17:       Update both $Q_{\omega_1}$ and $Q_{\omega_2}$ by minimizing (25);
18:       Sampling $a'_j$ with the reparameterization trick;
19:       Update $\pi_\theta$ by minimizing (26);
20:       Update entropy regularization coefficient $\alpha$ by
          (23);
21:       Update $Q_{\omega_1^-}$ and $Q_{\omega_2^-}$ by (34).
22:     end if
23:   end for
24: end for
```

C. Algorithm Analysis

We can analyze the computational complexity of MOC in three parts.

When dealing with the connection scheduling problem, it is necessary to determine for each IoTD which UAVs cover it. For IoTDs covered by multiple UAVs, the optimal UAV needs to be selected for connection. Therefore, the worst computational complexity for this problem is $\mathcal{O}(2KN)$.

For the resource allocation problem, according to (29), after determining the connection scheduling, it is necessary to first calculate the square root sum of the task loads of all IoTDs connected to the same UAV in order to determine the computing resources allocated to each IoTD. Therefore, the worst computational complexity for this problem is $\mathcal{O}(2K)$.

For the SAC algorithm, we can estimate its computational complexity through the neural network architecture. Since we use a basic multilayer perceptron to construct the actor network, critic network, and critic target network, these networks are isomorphic.

Let n_m^a be the number of neurons in the mth layer of the actor network, and n_l^c be the number of neurons in the lth layer of the critic network, where $m \in \{0, 1, \dots, M^a\}$ and $l \in \{0, 1, \dots, L^c\}$. Here, M^a and L^c represent the number of layers in the actor and critic networks, respectively. According to [29], [30], the computational complexity of the SAC neural network is approximately $\mathcal{O}(TE(\sum_{m=0}^{M^a-1} n_m^a n_{m+1}^a + \sum_{l=0}^{L^a-1} n_l^c n_{l+1}^c + J))$ after T time slots and E epochs of convergence.

Based on the previous discussion, it can be concluded that the overall computational complexity of MOC is approximately $\mathcal{O}(2K(N+1)TE(\sum_{m=0}^{M^a-1} n_m^a n_{m+1}^a + \sum_{l=0}^{L^a-1} n_l^c n_{l+1}^c + J))$.

V. Simulation Results

This section presents the comprehensive performance evaluation of SAC-based MOC deployed on AMD R7-5800H, NVIDIA RTX 3060, Python 3.10, Pytorch 2.0.0, and Gym 0.25.2.

A. Scenario Configuration

In our simulation experiments, we assume that the task area has a length of $X^{\max} = 400$ and a width of $Y^{\max} = 400$. The area is a densely distributed region of IoTDs, with 100 IoTDs randomly and uniformly distributed, including 20 high-DOP IoTDs with $P_k = 3$, 25 medium-DOP IoTDs with $P_k = 2$, and 55 low-DOP IoTDs with $P_k = 1$. In each time slot, each IoTD generates a task that needs to be completed within one time slot $\tau = 1$s, with a data size of $D_k(t) \in [1200, 1600]$ Kb. There are also 4 UAVs deployed in this field as edge servers, which need to adjust their flying trajectories to assist these IoTDs in offloading computation.

TABLE II
SIMULATION PARAMETERS

Parameters	Value	Parameters	Value
N	4	K	100
T	15	τ	1 s
$X^{\max}$	400 m	$Y^{\max}$	400 m
$C^{\max}$	30	$v^{\max}$	15 m/s
d^{safe}	20 m	d^{link}	100 m
H	100 m	f	60 GHz
G_t	55 dBi	G_r	55 dBi
r	0.35 m	R_m	0.55 m
h^f	1	l	0.25 m
κ_k	1e-28	κ_n	1e-28
$D_k(t)$	[1200, 1600] Kb	δ^2	-90 dBm
B	10 MHz	P^{off}	0.1 W
$f_k^{\max}$	2.5 GHz	$f_n^{\max}$	100 GHz

For the neural network part, we set the learning rate of the Actor l^a to 3e-4, the learning rate of the critic l^c to 3e-3, and initialize the entropy regularization coefficient α to 3e-4. In both the actor and critic networks, we deploy three fully connected hidden layers, each containing 800, 600, and 400 neurons, respectively.

The details of parameter settings used in the simulation experiments are based on the works in [21], [31], and presented in Table II.

To evaluate the performance of MOC, by referring to [10], [16], we proposed several algorithms for comparison, which are listed as follows.

1) *DDPG:* DDPG learns the optimal policy by approximating the Q-function and policy function. Its parameter settings are the same as actor and critic networks in MOC.
2) *PSO:* All the actions of UAVs within an epoch (T time slots) are concatenated to form the velocity of a particle in PSO. Then, the fitness of the velocity is calculated based on the execution cost and penalty of these actions.
3) *Random Moving RM:* Each UAV will take random flying angles and distances to move in each time slot.
4) *LE:* All IoTDs complete their computing tasks locally without offloading.

It should be noted that the connection scheduling and resource allocation in DDPG, PSO, and RM are the same as that in MOC.

B. Results and Performance Evaluation

1) Convergence Analysis: Fig. 3 compares the reward convergence of MOC with other algorithmic approaches. PSO has excellent global search capability and can achieve convergence in approximately 500 epochs. In contrast, both MOC and DDPG showed unsatisfactory initial performance. However, as they gained experience through interactions with the environment, their performance gradually improved and ultimately converged. This is because both MOC and DDPG use experience replay to improve sample utilization and use gradient descent to update neural network parameters. As long as the learning rate is appropriate, they can reach convergence after a limited number of iterations, thus ensuring the convergence of the algorithm. It can be seen that MOC achieved convergence in approximately 800 epochs, while DDPG required approximately 1100 epochs

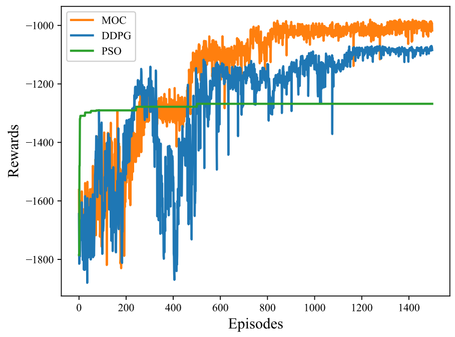

Fig. 3. Convergence comparison of different algorithms.

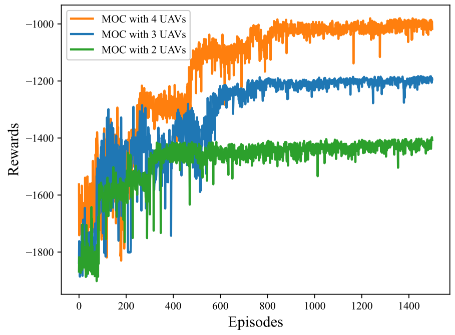

Fig. 4. Convergence comparison of different numbers of UAVs with MOC.

to achieve convergence. We can conclude that although MOC may not be as fast as PSO in terms of convergence speed, it appears to be more efficient and stable with faster convergence speed and higher cumulative rewards compared to DDPG, which is also a DRL algorithm.

Fig. 4 compares the reward convergence of MOC in scenarios with different numbers of UAVs. It shows that the algorithm converges fastest in the scenario with 2 UAVs, around 300 epochs, while it takes about 700 epochs to converge with 3 UAVs. From this figure, we can observe that MOC exhibits good convergence performance in different scenarios. Moreover, after convergence, the algorithm rarely shows oscillations or overfitting except for some exploratory actions. This is attributed to the advanced techniques employed in MOC, such as "soft update" and "entropy regularization."

2) Trajectory Analysis: Fig. 5 shows the trajectories of UAVs obtained by MOC, with the blue dashed line representing the coverage range of UAVs in the last time slot. It can be observed that in order to cover more IoTDs faster, all UAVs flew towards the dense area of IoTDs and maintain a relatively fast flying speed and stable flying angle in the initial time slots. In the later time slots, considering the constraints of UAVs' maximum number of connected users and load balancing among UAVs, it was not only necessary to increase the number of covered users, but also to expand the overlapping coverage area between UAVs to give IoTDs more choices. So, UAVs began to adjust their coordinates while maintaining a safe distance. Finally, in the last time slot, UAVs successfully covered almost all IoTDs.

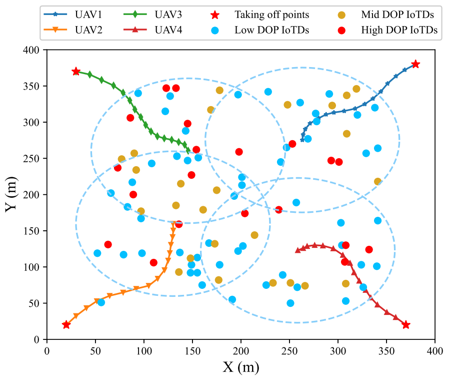

Fig. 5. Trajectories of UAVs with MOC.

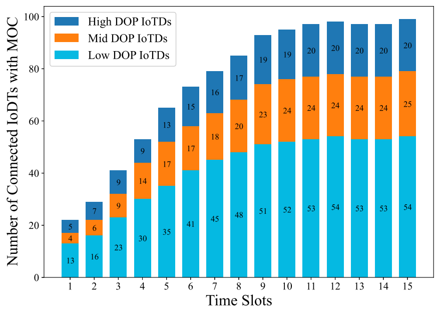

Fig. 6. Number of connected IoTDs with MOC.

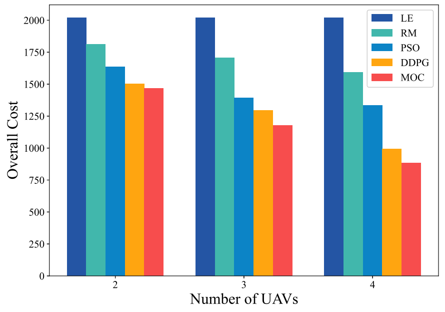

Fig. 7. Comparison of overall cost with different number of UAVs.

To visually and conveniently observe the trajectory changes of UAVs, we show in Fig. 6 the number of different DOP IoTDs connected by UAVs in each time slot. It can be seen that the total number of IoTDs connected by UAVs generally increased as the time slots increased. In the 11-th time slot, UAVs connected to all IoTDs with high DOP, and in the 15-th time slot, all IoTDs with medium DOP were also connected, indicating that the trajectory optimized by MOC did consider serving the IoTDs with higher DOP while reducing the overall cost of IoTDs.

3) Performance Analysis: First, we analyzed the overall cost of MOC, DDPG, PSO, RM, and LE in scenarios with different number of UAVs in Fig. 7. As we can observe, MOC performed the best among these schemes, reducing the overall cost to the minimum compared to other compared schemes. DDPG was only second to MOC, while PSO, RM, and LE showed poorer performances. In addition, we can also find that the total cost of LE does not change with the number of UAVs because there is no assistance from UAVs, which is in line with expectations. Meanwhile, the overall costs consumed by MOC, DDPG, PSO, and RM all showed a decreasing trend as the number of UAVs increased. This is because with the increase in the number of UAVs, there are more computing resources available in the scenario, and UAVs have a larger coverage range, which enables them to provide better services to more IoTDs. This is beneficial in reducing the cost of energy consumption and task execution delay of IoTDs.

Fig. 8(a)–(c), respectively, illustrate how the cost of IoTD changes over time slots in scenarios with different numbers of UAVs. The overall trend is consistent with that in Fig. 7. LE performs the worst among all algorithms since all IoTDs execute their computing tasks locally without the assistance of UAVs, resulting in inevitably higher costs of energy consumption and task execution delay. The performance of RM is better than LE, but its cost fluctuated as the number of assisted IoTDs varied due to the UAVs' completely random flying angles and distances in each time slot. MOC achieves the best optimization effect in all three scenarios, followed by DDPG, which ranked between MOC and PSO. PSO performs worse than these two DRL-based algorithms. We also observed that, except for LE, the overall trend of other algorithms was a decrease in cost as the number of time slots increased, which indicates that UAVs were gradually flying toward better coordinates to assist more IoTDs in offloading, thereby reducing the local cost of IoTDs and achieving better optimization effects in these algorithms.

Fig. 9(a) and (b), respectively, display the performance of the energy cost and delay cost of IoTDs consumed by different algorithms in the scenario with four UAVs. The trends observed in these figures are consistent with those in Fig. 8(c) and are related to the number of IoTDs connected to UAVs in the given scenario. In terms of energy consumption optimization, the effectiveness of different algorithms is quite apparent, and MOC achieving the greatest improvement by reducing the cost from around 35 J to nearly 1 J. This is because in the last few time slots, almost all IoTDs have been connected by UAVs, causing most users to consume only offloading energy, and their tasks' computing energy costed in UAVs, which is much higher. However, in terms of latency optimization, the numerical improvements are not as significant. Even with the best-performing MOC, the optimization effect was less than half as it only reduced the cost from around 72 s to around 39 s. This is because although UAVs have much higher computing performance than IoTDs, they also require a considerable amount of time to complete the task of offloading to them. Additionally, according to *Lemma 1*, the UAVs' load conditions can affect the allocation of computing resources for IoTDs, further affecting their latency during the offloading process. In summary, the greater the load gap between UAVs, the worse the optimization effect on latency.

Fig. 10 compares the variation of task loads among four UAVs in MOC over time slots. It can be observed that the overall trend is an increase in task loads with time slots. This is because at the beginning, the UAVs were deployed in the edge area, covering fewer IoTDs. In subsequent time slots, the UAVs began

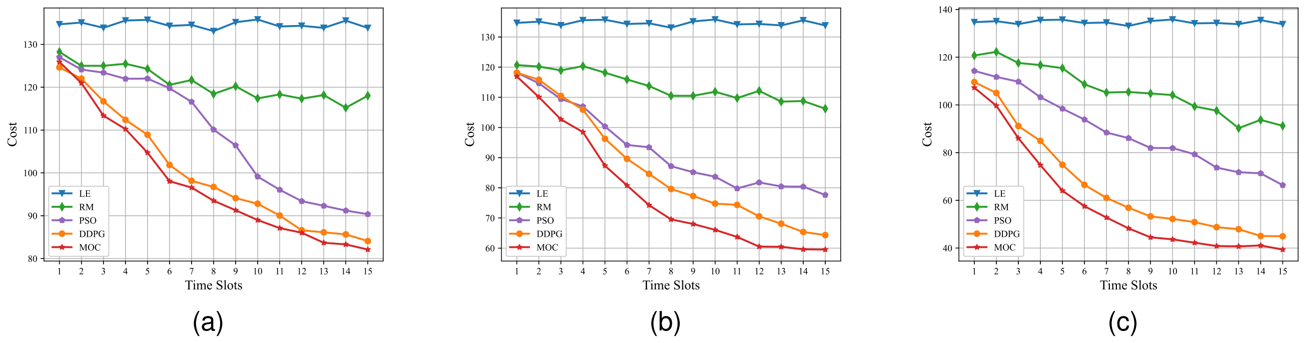

Fig. 8. Comparison of cost over time slots with different numbers of UAVs. (a) Cost of IoTDs with 2 UAVs. (b) Cost of IoTDs with 3 UAVs. (c) Cost of IoTDs with 4 UAVs.

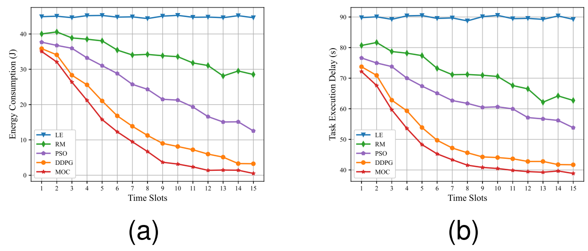

Fig. 9. Comparison of energy and delay cost with 4 UAVs. (a) Energy consumption of IoTDs with 4 UAVs. (b) Delay of IoTDs with 4 UAVs.

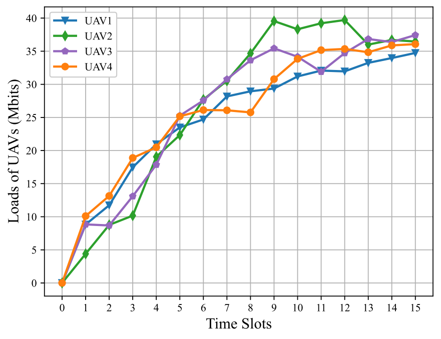

Fig. 10. Comparison of loads of UAVs with MOC.

to search for more optimal coordinates to assist more IoTDs in offloading computations to save overall costs, leading to an increase in their task loads. Additionally, we found that there was a significant difference in the task load of the 2-nd UAV compared to the others from time slots 9–12, but this difference began to decrease from the 13-th time slot. This is because the area near the 2-th UAV had a higher density of IoTDs than other areas, and it was able to connect to more IoTDs from the 9-th time slot, covering this area earlier than other UAVs. However, by the 13-th time slot, the coverage areas of multiple UAVs overlapped significantly, and some IoTDs were possibly covered by multiple UAVs simultaneously. To obtain more computing resources, these IoTDs would choose UAVs with smaller task loads for offloading. It can be seen that from the 13-th time slot, the difference in task loads among UAVs was not significant, which also reflects the good performance of our connection scheduling algorithm.

VI. Conclusion

In this article, in order to minimize the energy consumption and delay of IoTDs while considering the DOP, we comprehensively consider factors such as connection scheduling, computing resource allocation, and UAVs' flying trajectories, and then propose an optimization method MOC that combines convex optimization with DRL. Specifically, we developed a dynamic trajectory control scheme based on DRL and incorporated convex optimization theory as well as heuristic algorithm to assist in solving computing resource allocation and connection scheduling respectively. Simulation results show that, compared to other comparison schemes, MOC has significant advantages in many aspects of performance.

In future work, we will extend our work to consider the flying height variation of UAVs and study the optimization of more complex wireless communication problems in 3-D scenarios.

References

[1] M. Mehrabi, H. Salah, and F. H. P. Fitzek, "A survey on mobility management for mec-enabled systems," in *Proc. IEEE 2nd 5G World Forum*, 2019, pp. 259–263.

[2] S. S. D. Ali, H. Ping Zhao, and H. Kim, "Mobile edge computing: A promising paradigm for future communication systems," in *Proc. TENCON IEEE Region 10 Conf.*, 2018, pp. 1183–1187.

[3] S. A. Huda and S. Moh, "Survey on computation offloading in UAV-enabled mobile edge computing," *J. Netw. Comput. Appl.*, vol. 201, 2022, Art. no. 103341. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804522000108

[4] Z. Qin, Z. Liu, G. Han, C. Lin, L. Guo, and L. Xie, "Distributed UAV-BSs trajectory optimization for user-level fair communication service with multi-agent deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 12290–12301, Dec. 2021.

[5] S. Zhang, Z. Jiang, and R. Cao, "Joint optimization of computing offloading in multi-uavs-assisted mec system," in *Proc. IEEE 5th Int. Conf. Electron. Technol.*, 2022, pp. 617–622.

[6] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and ai for UAVs: Opportunities and challenges," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15435–15459, Sep. 2022.

[7] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 19–25, Jan. 2020.

[8] S. Mukhopadhyay and B. Jain, "Multi-agent markov decision processes with limited agent communication," in *Proc. IEEE Int. Symp. Intell. Control*, 2001, pp. 7–12.

[9] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4738–4752, Oct. 2019.

[10] Q. Tang, Y. Yang, L. Liu, and K. Yang, "Minimal throughput maximization of uav-enabled wireless powered communication network in cuboid building perimeter scenario," *IEEE Trans. Netw. Serv. Manag.*, vol. 20, no. 4, pp. 4558–4571, Dec. 2023.

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

[12] D. Wang, J. Tian, H. Zhang, and D. Wu, "Task offloading and trajectory scheduling for UAV-enabled MEC networks: An optimal transport theory perspective," *IEEE Wireless Commun. Lett.*, vol. 11, no. 1, pp. 150–154, Jan. 2022.

[13] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5505–5516, Aug. 2020.

[14] Z. Wang, G. Zhang, Q. Wang, K. Wang, and K. Yang, "Completion time minimization in wireless-powered UAV-assisted data collection system," *IEEE Commun. Lett.*, vol. 25, no. 6, pp. 1954–1958, Jun. 2021.

[15] 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 Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4531–4547, Dec. 2021.

[16] 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.

[17] A. Gao, Q. Wang, W. Liang, and Z. Ding, "Game combined multi-agent reinforcement learning approach for uav assisted offloading," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 12888–12901, Dec. 2021.

[18] B. Li, Y. Liu, L. Tan, H. Pan, and Y. Zhang, "Digital twin assisted task offloading for aerial edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 10, pp. 10863–10877, Oct. 2022.

[19] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5688–5698, Aug. 2021.

[20] Y. Guo, R. Zhao, S. Lai, L. Fan, X. Lei, and G. K. Karagiannidis, "Distributed machine learning for multiuser mobile edge computing systems," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 3, pp. 460–473, Apr. 2022.

[21] B. Chang, W. Tang, X. Yan, X. Tong, and Z. Chen, "Integrated scheduling of sensing, communication, and control for mmwave/thz communications in cellular connected UAV networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 7, pp. 2103–2113, Jul. 2022.

[22] A.-A. A. Boulogeorgos, E. N. Papasotiriou, and A. Alexiou, "Analytical performance assessment of thz wireless systems," *IEEE Access*, vol. 7, pp. 11436–11453, 2019.

[23] Q. Tang, L. Liu, C. Jin, J. Wang, Z. Liao, and Y. Luo, "An UAV-assisted mobile edge computing offloading strategy for minimizing energy consumption," *Comput. Netw.*, vol. 207, 2022, Art. no. 108857. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622000688

[24] Y. Du, K. Yang, K. Wang, G. Zhang, Y. Zhao, and D. Chen, "Joint resources and workflow scheduling in UAV-enabled wirelessly-powered MEC for IoT systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 10187–10200, Oct. 2019.

[25] X. Wang et al., "Dynamic resource scheduling in mobile edge cloud with cloud radio access network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2429–2445, Nov. 2018.

[26] A. R. Heidarpour, M. R. Heidarpour, M. Ardakani, C. Tellambura, and M. Uysal, "Soft actor–critic-based computation offloading in multiuser mec-enabled IoT—a lifetime maximization perspective," *IEEE Internet Things J.*, vol. 10, no. 20, pp. 17571–17584, Oct. 2023.

[27] X. Yu, B. Wang, and H. Dong, "A distributed algorithm based on KKT conditions for convex intersection computation," in *Proc. Chin. Automat. Congr.* 2017, pp. 7676–7680.

[28] Y. Cheng and Y. Song, "Autonomous decision-making generation of UAV based on soft actor-critic algorithm," *Proc. 39th Chin. Control Conf.*, , pp. 7350–7355, 2020, doi: 10.23919/CCC50068.2020.9188886.

[29] T. Yuan, C. E. Rothenberg, K. Obraczka, C. Barakat, and T. Turletti, "Harnessing UAVs for fair 5G bandwidth allocation in vehicular communication via deep reinforcement learning," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 4, pp. 4063–4074, Dec. 2021.

[30] Y. Liu, J. Yan, and X. Zhao, "Deep reinforcement learning based latency minimization for mobile edge computing with virtualization in maritime UAV communication network," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 4225–4236, Apr. 2022.

[31] 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 Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2021.

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.

Sihao Wen received the B.E. degree in computer science and technology from the Changsha University of Science and Technology, Changsha, China, in 2020.

His current research interests include mobile edge computing and wireless UAV communication network.

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.

Kun Yang (Fellow, IEEE) received the Ph.D. degree from the Department of Electronic & Electrical Engineering of University College London (UCL), U.K., in 2000.

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.

Dr. 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.