QoS-aware Energy-efficient Multi-UAV Offloading Ratio and Trajectory Control Algorithm in Mobile Edge Computing

Jiajie Yin, Zhiqing Tang, *Member, IEEE,* Jiong Lou, *Member, IEEE,* Jianxiong Guo, *Member, IEEE,* Hui Cai, *Member, IEEE,* Xiaoming Wu, Tian Wang, *Senior Member, IEEE,* Weijia Jia, *Fellow, IEEE*

***Abstract*—Multiple Unmanned Aerial Vehicle (UAV)-assisted Mobile Edge Computing (MEC) leverages UAVs equipped with computational resources as mobile edge servers, providing flexibility and low-latency connections, especially beneficial in smart cities and the Internet of Things (IoT). Maximizing Quality of Services (QoS) while minimizing energy consumption necessitates developing a suitable offloading ratio and trajectory control algorithm for UAVs. However, existing research on UAV control algorithms overlooks significant challenges like the heterogeneity of User Equipments (UEs) and offloading failures. Furthermore, there is a dearth of experimental validation in large-scale UAV-assisted MEC scenarios. To bridge these gaps, we introduce a QoS-aware Energy-efficient Multi-UAV Offloading ratio and Trajectory control algorithm (QEMUOT). Specifically, 1) A composite UE mobility model is proposed to enhance system heterogeneous modeling, encompassing models for high-speed, low-speed, and fixed UEs. 2) QEMUOT is devised using multi-agent reinforcement learning algorithms to determine offloading ratio and trajectory control decisions. To tackle sparse reward space and offloading failures, we employ expert demonstrations for pretraining and enhance reward mechanisms. 3) Experimental simulations illustrate that our algorithm outperforms baseline algorithms in user QoS with reduced energy consumption and demonstrates superior scalability in scenarios with numerous UAVs and UEs.**

Jiajie Yin is with Faculty of Arts and Sciences, Beijing Normal University, Zhuhai 519087, China and also with Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China. (E-mail: jiajieyin@mail.bnu.edu.cn)

Zhiqing Tang is with Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China, and also with Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China. (E-mail: zhiqingtang@bnu.edu.cn)

Jiong Lou is with Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. (E-mail: lj1994@sjtu.edu.cn)

Jianxiong Guo and Weijia Jia are with Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China and also with Guangdong Key Lab of AI and Multi-Modal Data Processing, BNU-HKBU United International College, Zhuhai 519087, China. (E-mail: {jianxiongguo, jiawj}@bnu.edu.cn)

Hui Cai is with School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China. (E-mail: carolinecai@njupt.edu.cn)

Xiaoming Wu is with Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China, and also with the Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan, China. (E-mail: wuxm@sdas.org)

Tian Wang is with Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China. (E-mail: tianwang@bnu.edu.cn)

(Corresponding author: Zhiqing Tang.)

***Index Terms*—Mobile Edge Computing, Multi-agent Deep Reinforcement Learning, Unmanned Aerial Vehicle, Heterogeneous Mobility Pattern**

I. Introduction

MOBILE Edge Computing (MEC) emerges as a promising solution in smart city and Internet of Things (IoT) by decentralizing computational resources to the network edge, thereby enhancing the Quality of Services (QoS) within the radio access network (RAN) [1]. Fixed-edge MEC encounters challenges such as single-point failure [2] and high deployment costs, necessitating redundancy [3]. In contrast, Unmanned Aerial Vehicle (UAV)-assisted MEC, using UAVs as mobile edge servers, offers flexible deployment in dynamic scenarios [4]. UAVs establish Line of Sight (LoS) communication links at elevated altitudes for low-latency connections and enhance robustness through dynamic path planning.

In UAV-assisted MEC systems, scheduling UAV clusters is a crucial issue. Achieving load balance across each UAV and ensuring comprehensive service coverage for all users demand sophisticated trajectory control for UAVs [5]. Furthermore, given the constrained computing resources [4], UAV control involves managing not just the trajectory but also utilizing UAVs as airborne relay stations. These UAV stations offload computational tasks exceeding their capabilities to ground base stations (BS), hence necessitating control of the offloading ratio [6]. When UAVs fly along different routes, they will be connected to different User Equipments (UEs) and receive various computing requests. This will affect communication delays and energy consumptions, resulting in different outcomes with the same offloading ratio. Therefore, to improve QoS and reduce energy consumption, it is essential to address trajectory and offloading ratio control decisions simultaneously.

Unlocking the full potential of UAVs in MEC can effectively provide users with higher QoS. However, several challenges need to be addressed. *The first challenge is how to accurately model the mobility of UEs, considering the dynamically changing UE distribution.* UE's high mobility leads to frequent changes in the location, resulting in varying feasibility in the allocation of communication and computation resources over time [7]. Consequently, this poses significant challenges to MEC systems [8]. In practical smart city and IoT scenarios, UEs demonstrate heterogeneous mobility characteristics [9]. Some UEs are highly mobile, such as smart vehicles and logistics robots, while others have limited mobility, like wearable

Extended Reality (XR) devices used by pedestrians. Additionally, some UEs remain stationary, such as smart furniture. Ignoring the varied mobility patterns of UEs in design assumptions disconnects from real-world situations. Developing algorithms based on inaccurate UE mobility models presents significant challenges [10] and can compromise the reliability of algorithm validation experiments. It is essential to integrate realistic UE mobility models into algorithm design to ensure their practicality and adaptability in real-world environments. *Moreover, offloading failure (i.e., offloading interruption) is a typical issue due to the mobility of UEs* [11], [12]. UEs need to ensure a stable communication link with the server while offloading within the coverage area. Otherwise, interruptions in the connection can cause offloading failures, resulting in significant wastage of computational resources and a decline in QoS [13].

Limited attention has been given to studying the diverse movement patterns of the UE and offloading failures in UAV-MEC research. To address these gaps, we have enhanced our model by introducing a composite UE motion model and redesigning the reward function in our algorithms, as explained in the next paragraph. This improvement not only enhances connection stability but also decreases decision-making costs for users, resulting in significant QoS improvements.

Another challenge is how to make offloading and trajectory decisions for each UAV. In dynamic environments with real-time information, traditional optimization algorithms like Successive Convex Approximation [14], [15] and Block Alternating Descent [16] are impractical due to their high computational complexity. As a result, researchers have increasingly turned to Multi-Agent Reinforcement Learning (MARL) as a promising alternative [5], [17]. To address this complex non-convex optimization problem, we convert it into a Decentralized Partially Observable Markov Decision Process (Dec-POMDP). To tackle this challenge, we introduce a QoS-aware Energy-efficient Multi-UAV Offloading ratio and Trajectory control algorithm (QEMUOT) based on the Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3) framework [18], where each UAV is treated as an intelligent agent.

However, the widespread adoption of IoT has led to an increasing demand for the number of UAV servers in MEC systems [19]. This causes the joint action space and state space of MARL to expand exponentially with the number of agents [20], forming a more complex and reward-sparse environment. Traditional exploration methods easily become trapped in low-reward regions, posing challenges in collecting effective policy experiences with high rewards [21]. This results in low training efficiency and difficulties in convergence to the optimal solution. To address this challenge, we draw inspiration from imitation learning to enhance the pretraining process of the QEMUOT algorithm. This is achieved by leveraging an expert algorithm which combines the Sailfish optimization algorithm [22] with a greedy algorithm.

In this paper, we present a novel composite UE mobility model aimed at addressing the diverse mobility patterns of users. We propose the QEMUOT algorithm, which leverages MATD3 for making joint offloading ratio and trajectory control decisions. To expedite the training process, we integrate expert demonstrations into the algorithm using a novel expert strategy. Furthermore, We introduce a modified reward mechanism to prevent offloading failures by penalizing actions that lead to such failures. Through a series of experiments, we evaluate the performance of the QEMUOT algorithm, demonstrating its superior convergence speed and effectiveness in catering to high mobility and diverse UEs. The algorithm shows an increase in reward of up to 62% compared to baseline algorithms. Moreover, it proves to be applicable in larger-scale experiments and exhibits stability over baseline approaches. The key contributions of this paper can be summarized as follows:

1) We classify UEs into 3 categories according to UEs' different mobility abilities and patterns of movement: High-speed UEs along city road network, Low-speed UEs not along city road network and Fixed UEs. Then we propose a composite UE mobility model to better manage the heterogeneous of edge devices.
2) To optimize offloading ratio and trajectory control decisions, we introduce the QEMUOT algorithm within the MATD3 framework. To tackle the challenge of sparse rewards, we integrate expert demonstrations for pretraining. Additionally, the reward mechanism is improved by introducing a penalty for offloading failures.
3) Experimental results show that, compared to traditional scheduling strategies, the QEMUOT algorithm demonstrates superior convergence speed and effectiveness in addressing the requirements of high mobility, diverse UEs, and large-scale UAV-assisted MEC networking scenarios.

The remainder of the paper is organized as follows. In Section II, the related work of our topic is illustrated. The system model and problem formulation are described in Section III and then reformulated as a Dec-POMDP process in Section IV. QEMUOT algorithm is proposed in Section V. Performance is evaluated by experiment in Section VI. In Section VII, several issues are further discussed. Finally, Section VIII gives a conclusion of the paper and some possible future research directions.

II. Related Work

A. Mobility Model

Current research has not extensively explored the high mobility of UEs and the differentiation among different mobility patterns. Many models operate under the premise of UEs being stationary and their positions being constant, thereby overlooking their mobility or considering all UEs as uniform entities [5], [6], [16], [29], [30]. Various mobility models for individuals in urban environments have been put forth in previous studies.

1) Mathematical model:

- Random Walk (RW) [31]: It aims to simulate the unpredictable stochastic movement characteristics of individuals. In [32], UEs are initialized at random positions within a rectangular area and commence random walks.

TABLE I
COMPARATIVE ANALYSIS OF RELATED WORKS.

Reference	No. of UAVs	Mobility of UEs	Offloading Decision	QoS-aware	Energy-efficient	Algorithm
[23]	1	Single mobile UE	Ratio	✓	✗	Analytical (Closed-form)
[14]	1	Fixed	No decision	✗	✓	Heuristic (SCA)
[16]	1	Fixed	Binary	✗	✓	Heuristic (BAD)
[15]	3	Fixed	Binary	✓	✗	Heuristic (SCA)
[24]	10	Fixed	Binary	✗	✓	Meta-heuristic (FCM)
[25]	1	Fixed	Binary	✓	✗	Meta-heuristic (D-WOA)
[26]	4	Markovian mobility model	Binary	✓	✗	RL (Q-learning)
[27]	1	Gauss-Markov model	Binary	✓	✓	RL (Double DQN)
[5]	4	Fixed	Binary	✗	✓	MARL (MADDPG)
[28]	9	Fixed	No decision	✓	✓	MARL (MADDPG)
[29]	4	Fixed	Binary	✓	✓	MARL (Nash Q-learning)
[30]	2	Fixed or Random walk model	Ratio	✓	✓	MARL (MATD3)
[6]	3	Fixed	Ratio	✓	✓	MARL (IPPO)
Proposed	$8 \sim 14$	Heterogeneous mobility patterns	Ratio	✓	✓	MARL (QEMUOT)

- Random WayPoint (RWP) [33]: Widely used to simulate user mobility in wireless cellular networks, involving individuals alternating between staying put and moving towards a random destination. The RWP-Ci model is an enhancement of RWP based on urban street maps, offering a more accurate simulation of the movement trajectories of urban users in real scenarios [34].
- Gauss-Markov [35]: It assumes that an individual's velocity is correlated over time and is modeled with a Gaussian-Markov process, which has been utilized in several recent MEC models [36].
- Individual Mobility (IM) [37]: It proposes an enhancement to the RW model by introducing two human-specific mobility mechanisms. The single-hop mobility under the IM model is assessed in [38], examining the practicality of simulating UE mobility in 5G small-cell network scenarios.

2) Traffic simulation software: In MEC scenarios, some researchers have started using traffic simulation software such as SUMO [39] to generate UE trajectories for simulation experiments [40].

3) Real-world data: Leveraging real-world data, such as GPS trajectory data from mobile devices or traffic data from cities, offers insights into genuine scenarios [41], [42].

To enhance the transition of models from the lab to practical applications, real data grounding is crucial. However, the scarcity of datasets with varied UE mobile trajectories and real-time upload records poses a challenge for data collection. While simulation software can mimic real-world results, its complex algorithms require substantial computational resources and time [43]. Therefore, this paper focuses on introducing a composite UE motion model. This model, despite its lightweight design, exhibits strong simulation capabilities for a range of UE movements.

Hence, considering their simplicity, flexibility, and interpretability, mathematical models are widely applied in the field of communications. However, some of the existing mathematical models often only excel at simulating certain types of UEs. Therefore, our work focused on proposing a composite UE motion model. This model, while maintaining a lightweight structure, also demonstrates robust simulation performance for diverse UE movements.

B. MARL for UAV-assisted MEC

In Table I, we present a comparative analysis of our work against key related studies on UAV-assisted MEC systems. The comparison includes the number of UAVs scheduled (No. of UAVs), consideration of UE mobility, type of offloading decisions, optimization objective of the algorithms, and the method employed.

In the realm of UAV-assisted MEC systems, various studies have investigated the use of MARL methods to address scheduling and offloading decisions faced by drones. Wang *et al.* [5] utilize the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm to improve fairness in serving user devices while reducing device energy consumption. However, this approach prioritizes QoS while neglecting the energy consumption of the entire MEC system. The MADDPG algorithm is employed in [28], demonstrating superior convergence properties compared to traditional single-agent algorithms and heuristic methods. Gao *et al.* [28] emphasize simulation realism by considering three-dimensional UAV movement and obstacle avoidance in urban scenarios but overlook the mobility of UE. Lee *et al.* [6] use an Independent Proximal Policy Optimization (IPPO)-based algorithm but do not compare it with other MARL algorithms. Furthermore, their experimental evaluation lacks generalizability. Zhao *et al.* [30] employ the MATD algorithm, providing comprehensive considerations for system models and optimization objectives. Ning *et al.* [44] adopt the MADDPG algorithm with a Prioritized Experience Replay (PER) technique. However, their experiments only assess the scheduling of 2-3 UAVs, failing to explore larger-scale networking scenarios.

Furthermore, Uchendu *et al.* [45] conduct a study on MARL utilizing Behavior Cloning (BC) pretraining. They highlight that initializing the critic network randomly could result in the loss of a well-performing initial policy by the end of pretraining, leading to a notable decline in actor network performance. Traditional expert demonstrations commonly involve offline learning with datasets. Qiu *et al.* [46] introduce a demonstration method in algorithmic form and integrated it with MADDPG. Their experimentation in a classic multi-agent particle environment notably enhance sample efficiency and policy performance in cluster control. However, they did

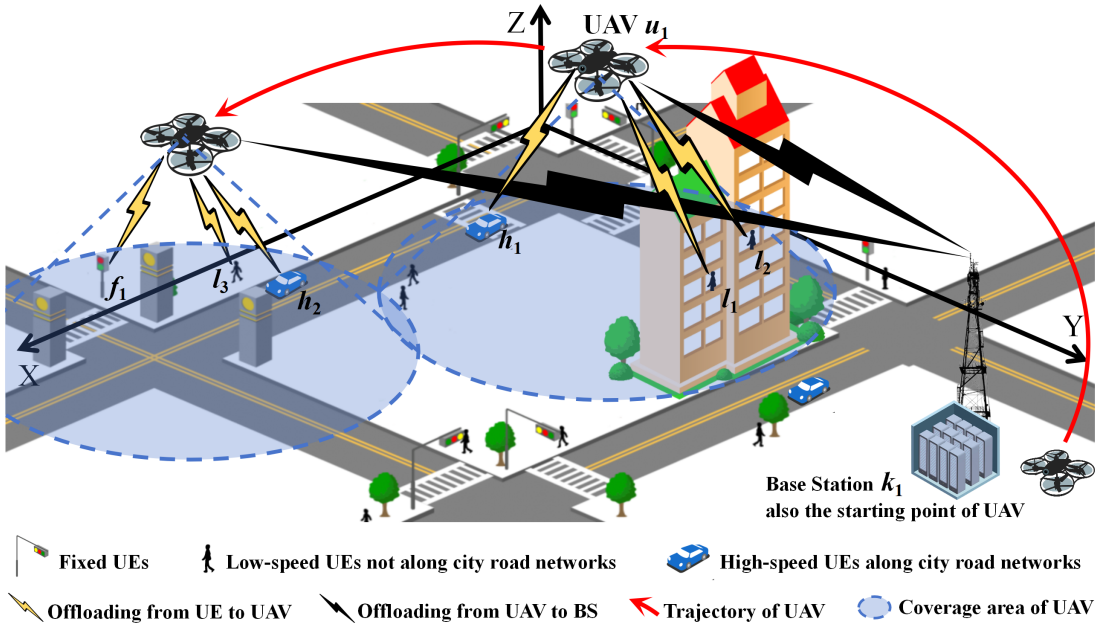


Fig. 1. Overall system model architecture in smart city IoT scenario.

not experiment with more advanced MARL algorithms such as MATD3, and the application of this pretraining technique in the UAV-assisted MEC field remains unexplored.

III. System Model and Problem Formulation

Consider a multi-UAV MEC network operating in discrete time, comprising a set of UAVs $\mathbf{U}$, a set of UEs $\mathbf{M}$, and a set of BS $\mathbf{K}$. As shown in Fig. 1, UAVs take off from the base station, establishing a network. The UEs are randomly distributed in the square-shaped area with a side length s, while multiple UAVs fly over this region and directly communicate with UEs to provide MEC services. UEs are classified into three categories based on their motion characteristics: $\mathbf{H}$ for High-speed UEs, $\mathbf{L}$ for Low-speed UEs, and $\mathbf{F}$ for Fixed UEs, where $\mathbf{M} = \mathbf{H} \cup \mathbf{L} \cup \mathbf{F}$. In each time slot t, every UE $m \in \mathbf{M}$ generates a computation-intensive task $W_m(t)$ that needs to be offloaded. $D_m(t)$ and $C_m(t)$ denote the size of task data and the number of CPU cycles required for each bit of data, respectively. QoS refers to the overall performance of a network or a network service, as perceived by the end users. High QoS ensures that the network provides satisfactory service to its users by meeting specific performance metrics.

A. UE Mobility Model

(1) High-speed UEs along city road networks. Examples include vehicles and devices mounted on them. These UEs utilize a RWP-Ci model [33], which integrates an exploration mechanism and a return mechanism. They move at a constant speed V_h on city streets. UE $h \in \mathbf{H}$ selects a destination and moves to it following the shortest path. Upon reaching the destination, h remains at the current location for a specified time t_h, after which it selects another destination, repeating this process.

- Exploration Mechanism: The UE h may choose an intersection point that has never been reached as the destination with the probability $P_{new} = \rho_h n_S^{-\psi}$, where n_S is the number of reached points, $\rho_h \in (0, 1]$, and $\psi > 0$.
- Return Mechanism: The UE h selects an intersection point that has been reached before as the destination with a probability of $P_{old} = 1 - P_{new}$.

(2) Low-speed UEs not along city road networks. Examples include pedestrians carrying user devices and intelligent robots. This category of individuals utilizes the Gauss-Markov model [47] to capture their movement patterns, which are not dependent on road networks. For UE $l \in \mathbf{L}$, the velocity at time t is denoted as $\boldsymbol{v}_l(t)$, and $\boldsymbol{v}_l(t+1)$ is calculated as follows:

$$\boldsymbol{v}_l(t+1) = \alpha \boldsymbol{v}_l(t) + (1-\alpha)\overline{\boldsymbol{v}}_l + \overline{\sigma}_l \sqrt{1-\alpha^2}\boldsymbol{w}_l(t), \quad (1)$$

where $\boldsymbol{w}_l(t) \sim \mathcal{N}(0, \sigma_w^2)$. α, $\overline{\boldsymbol{v}}_l$, and $\overline{\sigma}_l$ represent the memory level, asymptotic mean, and standard deviation of velocity, respectively. Then, the coordinates of user l at time t, $\boldsymbol{p}_l(t) = [x_l(t), y_l(t)]$, are updated as $\boldsymbol{p}_l(t+1) = \boldsymbol{p}_l(t) + \boldsymbol{v}_l(t)\Delta t$, where Δt is the time interval. To constrain UEs from leaving the specified area, if the calculated $\boldsymbol{p}_l(t+1)$ is outside the area, the UE maintains its current position.

(3) Fixed UEs. Examples include smart furniture, where individuals are randomly distributed within the region and remain stationary.

B. UAV Mobility Model

It is assumed that UAVs fly at a fixed altitude Z with a maximum speed of $V_{\max}$. The motion of UAV u at time t is represented by the tuple $(v_u(t), \theta_u(t))$, where $v_u(t) \in [0, V_{\max}]$ and $\theta_u(t) \in [-\pi, \pi]$ are the constant velocity of uniform flight and direction angle within the time slot $(t, t+\Delta t)$, respectively.

The flight distance is $\Delta d_u(t) = v_u(t)\Delta t$. The propulsion power is obtained as [48]:

$$P^{pro}(V) = P_0\left(1+\frac{3V^2}{{V_{tip}}^2}\right) + P_i\left(\sqrt{1+\frac{V^4}{4{v_0}^4}}-\frac{V^2}{2{v_0}^2}\right)^{\frac{1}{2}} + \frac{1}{2}d_0\rho r_s s_d V^3, \tag{2}$$

where P_0 is blade profile power in hovering and V_{tip} is the tip speed of rotor blade. P_i and v_0 denote induced power and the mean rotor induced velocity under the hover condition. As for parasite power, d_0, ρ, r_s, and s_d denote the fuselage drag ratio,air density,rotor solidity, and rotor disc area, respectively.

C. Communication Cost

(1) Offloading transmission from UEs to UAVs. At time t, the coordinates of UAV u, denoted as $\boldsymbol{X}_u(t)$, are expressed as $(x_u(t), y_m(t), Z)$. The position of UE m, denoted as $\boldsymbol{X}_m(t)$, is represented as $(x_m(t), y_m(t), 0)$, and the position of BS k is given by $(x_k, y_k, 0)$. The service area of the UAV is characterized by a circular region [49]. The coverage radius of a UAV at work is $r_c = \frac{Z}{\tan(\Theta)}$, where Θ denotes the maximum coverage angle. The elevation angle between UAV u and UE m at time t is denoted as $\theta_{um}(t)$. The probabilities of establishing LoS and non-LoS (NLoS) connections can be expressed as:

$$P_{um}^{LoS} = \frac{1}{1+a\exp\left(-b\left[\theta_{um}(t)-a\right]\right)}, \tag{3}$$

$$P_{um}^{NLoS} = 1 - P_{um}^{LoS}, \tag{4}$$

where a and b are constants determined by the communication environment.

The channel gain between u and m during offloading is obtained as:

$$g_n(t) = \frac{1}{K_0\left(P_{um}^{LoS}\mu_{LoS} + P_{um}^{NLoS}\mu_{NLoS}\right)\left[Z^2 + {d_{um}}^2(t)\right]}, \tag{5}$$

where $K_0 = (4\pi f_c/c)^2$, $1/K_0$ represents the channel power gain at the reference distance $d_0 = 1$m, f_c is the carrier frequency, c is the speed of light, μ_{LoS} and μ_{NLoS} are the attenuation factors for LoS and NLoS links. $d_{um}(t)$ is the horizontal distance between u and m.

The offloading data rate is calculated as:

$$R_{um}(t) = \left(B_U/N_u^M(t)\right)\log_2\left[1+g_{um}(t)P_M/{\sigma_U}^2\right], \tag{6}$$

where B_U is the bandwidth of the UAV, $N_u^M(t)$ is the number of UEs offloading computational tasks to u in time slot t. We assume that the bandwidth is equally shared among all UEs. P_M is the transmit power of the UE. σ_U^2 is the additive Gaussian white noise power for UAV communication.

The transmission delay and energy consumption are obtained as:

$$T_{um}^{trans}(t) = D_m(t)/R_{um}(t), \tag{7}$$

$$E_{um}^{trans}(t) = \left[P_M + P_U^r/N_u^M(t)\right]T_{um}^{trans}(t), \tag{8}$$

where P_U^r is the receiving power of UAVs. A UAV can only provide offloading services to UEs within its coverage area, i.e., $d_{um}(t) < r$, and at one time slot, a UE can only offload tasks to one UAV.

An offloading indicator variable $\xi_{um}(t)$ is defined, where $\xi_{um}(t) = 1$ when UE m is served by UAV u, and $\xi_{um}(t) = 0$ otherwise. Assuming that each UE can be served by at most one UAV at any given time, satisfying $\sum_u^{|\mathbf{U}|}\xi_{um}(t) \in \{0,1\}$. 0 indicates that the UE is currently in the coverage blind spot of the UAV. Thus, there is no UAV available for offloading computational tasks, and 1 otherwise. The UE selects the offloading UAV $\hat{u}$ with the minimal transmission delay when it is within the overlapping coverage zone of multiple UAVs: $\hat{u} = \operatorname{argmin}\left\{T_{um}^{trans}(t)\right\}, m \in \mathbf{U}_m$, where $\mathbf{U}_m$ is the available UAVs set of UE m.

(2) Offloading transmission from UAVs to BSs. The data rate of the wireless link between UAV u and BS k at time t is calculated as follows:

$$R_{uk}(t) = B_k\log_2\left[1+g_{uk}(t)P_U^t/\left(N_u^K(t){\sigma_K}^2\right)\right], \tag{9}$$

where P_U^t represents the transmission power of UAV, and $N_u^K(t)$ denotes the quantity of tasks that u intends to offload to the BS during time slot t. It is assumed that the BS can provide a connection bandwidth B_K to the UAV.

Each UAV has a finite capacity $\epsilon_{\max}$. A task queue model is employed following a First-In-First-Out (FIFO) policy. When the incoming tasks surpass $\epsilon_{\max}$, the UAV must offload them to the nearest BS. Furthermore, UAV retains the option to either process the tasks or offload them entirely to the BS. In each time slot, the UAV makes an offloading ratio decision, denoted as $\delta_u(t) \in [0,1]$. Let $\epsilon_u(t)$ denote the number of tasks in the task queue of UAV u in the current time slot. Define the indicator variable $\beta_{um}(t)$ for UAV u deciding whether to offload $W_m(t)$. In time slot t, UAV u processes the first $\nu_u(t)$ tasks locally in its queue, $\beta_{um}(t)$ is set to 1. The subsequent tasks are offloaded to the BS and $\beta_{um}(t) = 0$, $\nu_u(t) = \lfloor\epsilon_u(t)\left[1-\delta_u(t)\right]\rfloor$. The transmission delay is determined by:

$$T_{umk}^{trans}(t) = \frac{\left[1-\beta_{um}(t)\right]D_m(t)}{R_{mk}(t)}, \tag{10}$$

$$E_{umk}^{trans}(t) = \frac{P_M^t}{N_u^K(t)}T_{umk}^{trans}(t). \tag{11}$$

D. Computation Cost

(1) Computation at UAVs. In the conventional FIFO queue, tasks are typically executed in a sequential manner. However, this sequential execution approach, when applied to MEC servers, can result in timeouts for subsequent tasks. In our pursuit of equitable service provision for each user, we propose the incorporation of a parallel processing mechanism in UAV-MEC. This mechanism allows tasks in the queue to be executed in parallel to a certain extent.

Given that tasks do not actually arrive simultaneously at UAV-MEC, this assumption may introduce some degree of error. We have conducted an analysis of this error, as depicted in Fig. 2. Notably, the transmission delay of tasks is significantly smaller than the computation delay, exhibiting a difference of approximately 3 orders of magnitude. Consequently, this error

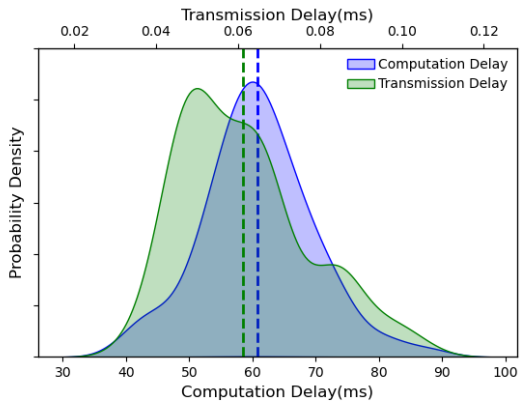


Fig. 2. Kernel Density plot for the transmission delay and computation delay of tasks.

can be deemed negligible and falls within an acceptable range.

The total computing resources, denoted as F_U, are equitably distributed among all tasks presently in progress. The computation delay and energy consumption for UAV u are obtained as [41]:

$$T_{um}^{comp}(t) = \frac{\beta_{um}(t) D_m(t) C_m(t)}{f_{um}(t)}, \tag{12}$$

$$E_{um}^{comp}(t) = \kappa f_{um}(t)^3 T_{um}^{comp}(t), \tag{13}$$

where $\kappa = 10^{-26}$ is a hardware related constant and $f_{um}(t)$ is the computing resource allocated by the UAV to the task. Due to the assumption of fair distribution of total computing resources, $f_{um}(t) = F_U / N_u^M(t)$.

(2) Computation at BSs. The UAV always offloads to the BS $\hat{k}$ with the minimum transmission delay, i.e., the closest BS in horizontal distance: $\hat{k} = \operatorname{argmin}\{d_{uk}(t)\}, k \in \mathbf{K}$. The BS computation delay for UE m's task can be calculated by $T_{umk}^{comp}(t) = \beta_{um}(t) D_m(t) C_m(t) / F_K$, where F_K is the computing resources allocated by the BS to each task.

E. Problem Formulation

We aim to maximize QoS while minimizing energy consumption. In our work, maximizing QoS involves addressing three critical aspects: maximizing service coverage, minimizing delay, and reducing offloading failure. In time slot t, UAV u's energy consumption can be calculated as:

$$E_u^{task}(t) = \sum_{m=1}^{|\mathbf{M}|} \xi_{um} \left[E_{um}^{trans}(t) + E_{umk}^{trans}(t) + E_{um}^{comp}(t) \right], \tag{14}$$

$$E_u^{pro}(t) = \Delta P^{pro}\left(v_u(t)\right), \tag{15}$$

where $E_u^{task}(t)$ and $E_u^{pro}(t)$ denote the task-processing and propulsion energy consumption, respectively. The total energy consumption is $E_u(t) = E_u^{task}(t) + E_u^{pro}(t)$. The total computation delay on UAV u at time t is expressed as:

$$\tau_u(t) = \sum_{m=1}^{|\mathbf{M}|} \xi_{um} \left[T_{um}^{trans}(t) + \max\{ T_{um}^{comp}(t), T_{umk}^{trans}(t) + T_{umk}^{comp}(t) \} \right]. \tag{16}$$

The weighted sum of $E_u(t)$ and $\tau_u(t)$ is represented as the system cost $C_u(t) = \omega_1 E_u(t) + \omega_2 \tau_u(t)$, where ω_1 and ω_2 are weights signifying the relative importance of energy consumption and execution delay, respectively. By simultaneously optimizing UAV's mobility decisions $(v_u(t), \theta_u(t))$ and offloading ratio $\delta_u(t)$, the optimization problem is formulated as follows:

$$\min_{v_u(t), \theta_u(t), \delta_u(t)} \sum_{t=1}^{|\mathbf{T}|} \sum_{u=1}^{|\mathbf{U}|} C_u(t) \tag{17}$$

$$\mathbf{s.t.} \quad \omega_1 + \omega_2 = 1 \tag{18a}$$

$$(0,0) \le (x_u(t), y_u(t)) \le (s,s)), \quad \forall u \in \mathbf{U} \tag{18b}$$

$$(x_u(0), y_u(0)) \in \mathbf{V}, \quad \forall u \in \mathbf{U} \tag{18c}$$

$$(x_k, y_k) \in \mathbf{V}, \quad \forall k \in \mathbf{K} \tag{18d}$$

$$d_{uu'}(t) \ge D_{\min}, \quad \forall u, u' \in \mathbf{U},\ u \ne u' \tag{18e}$$

$$d_{um}\left(t + \tau_n(t)\right) \xi_{um}(t) \beta_{um}(t) \le r, \quad \forall u \in \mathbf{U},\ m \in \mathbf{M} \tag{18f}$$

where $D_{\min}$ in Eq.(18e) is defined as the minimum flying distance established to prevent collisions among UAVs, and $\mathbf{V}$ in Eq.(18d) denotes the set of vertices within the specified square area, coinciding with the location of BSs. As defined in Eq.(18c) and Eq.(18b), UAVs initiate their operation from the BS position and are mandated to remain within the predefined area. To prevent offloading failures, Eq.(18f) guarantees the UE stays within the service range of the UAV during the transmission of computing results. Each UAV naturally serves as an agent, rendering it highly suitable for exploration within the framework of MARL.

IV. POMDP Formulation

The joint optimization of the UAV-assisted MEC system can be formulated as a Dec-POMDP process: $\langle \mathbf{N}, \mathbf{S}, \mathbf{A}, \mathcal{P}, \mathcal{R}, \mathbf{O}, n, \gamma \rangle$ [50], where $\mathbf{N}$ is the set of agents, $\mathbf{S}$ is the set of states, $\mathbf{A}$ is the set of actions, $\mathcal{P}$ is the transition function of state, $\mathcal{R}$ is the reward function shared by all the agents, $\mathbf{O}$ is the set of observations, n is the amount of the agents and γ is identified as the discount factor. The details are as follows:

1) Agent: Each UAV serves as an agent and $\mathbf{N}$ is a finite set of $n = |\mathbf{U}|$ agents.

2) State: The state at time t includes the location information, motion state, and connection status of all UAVs and UEs, denoted as $s(t) \in \mathbf{S}$.

3) Action: UAV decisions encompass both mobility strategy and task offloading ratio. At time slot t, the action for UAV u is represented as $a_u(t) = \{v_u(t), \theta_u(t), \delta_u(t)\}, a_u(t) \in \mathbf{A}$, while the global action is represented as $a(t) = \{a_u(t) \mid \forall u \in \mathbf{U}\}$.

4) Transition: When the agents interact with the environment by performing actions $a(t)$, the state transitions to $s(t+1)$ based on the transition function $\mathcal{P}(s(t+1)|s(t), a(t))$.

5) Observation: The observation set is denoted as O. UAV u's local observation $o_u(t) \in \mathbf{O}$ at time t is a partial information obtained from $s(t)$, including the relative positions of all UEs and other UAVs with respect to u, the motion states of all agents, and the offloading decision between all

UEs and u. Formally, $o_u(t) = \{\{\boldsymbol{X}_u(t) - \boldsymbol{X}_{u'}(t) \mid \forall u' \in \mathbf{U}\}, \{\boldsymbol{X}_u(t) - \boldsymbol{X}_m(t) \mid \forall m \in \mathbf{M}\}, \{\xi_{um}(t) \mid \forall m \in \mathbf{M}\}\}$.

6) Reward: The reward function $\mathcal{R}_u(t)$ for UAV u is defined as follows:

$$\mathcal{R}_u(t) = \begin{cases} \eta_1/C_u(t), & \text{if satisfying constraints,} \\ -\eta_2 N_u^C(t) - \eta_3 N_u^F(t) - \eta_4 \left[|\mathbf{U}| - \sum_{u=1}^{|\mathbf{U}|} N_u^M(t)\right] \\ \quad + \eta_5 \epsilon_u(t), & \text{otherwise,} \end{cases} \tag{19}$$

where η_1 represents the hyperparameter tied to the system cost. On the other hand, η_2 constitutes the collision constraint that penalizes both UAVs if their distance falls short of the predetermined safety parameters. In this equation, $N_u^C(t)$ denotes the count of UAVs within the safety perimeter of u. Specially, η_3 is identified as the offloading failure constraint where N_u^F indicates the number of tasks on u subjected to offloading lapses. Essentially, η_3 serves as a deterrent against offloading failures, aiming to prompt UAVs to dynamically adjust to variations in UE locations. This, in turn, reduces the occurrence of connection interruptions, ensuring the robust execution of tasks on the UAV. As the equation progresses, η_4 symbolizes the no-service constraint, which imposes a penalty on all UAVs should any UEs be left unattended. Finally, η_5 ascribes to the service compensation, offsetting the no-service penalty in proportion with the current number of UEs attended to by UAV u.

Therefore, induced by the expected reward of UAV agents, the action-value function is defined as follows:

$$Q_u(s(t), a_u(t)) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_u(t) | s(t), a_u(t)\right], \tag{20}$$

where $\gamma \in [0, 1)$.

V. QEMUOT Algorithm

The QEMUOT algorithm strategically determines offloading ratios and trajectory controls based on the MATD3 [18] architecture. Within the MADDPG algorithm framework, each agent is equipped with an actor network (Policy function $\mu_u(o)$) responsible for selecting actions to maximize the expected return, and a critic network (Value function $Q_u(s, a_1, a_2, ..., a_U)$) evaluating the future return expectancy associated with specific actions. MATD3 adopts a dual-critic mechanism, where each agent has an additional critic network to reduce estimation bias, thereby enhancing training stability. Following the Centralized Training with Decentralized Execution (CTDE) paradigm, in the QEMUOT algorithm, the critic networks undergo centralized training, while actor networks undergo decentralized training.

As stated in Algorithm 2, we first randomly initialized the actor network $\mu_u(o)$ with weights θ_u, and critic networks $\{Q_{u,i}(s, a_1, a_2, ..., a_U)\}_{i=1,2}$ with weights $\{\omega_{u,i}\}_{i=1,2}$ for each agent u. The target networks, denoted as $\hat{\mu}_u$ and $\{\hat{Q}_{u,i}\}_{i=1,2}$, are initialized as copies of the actor and critic networks, respectively. In order to enhance sample efficiency and stabilize the training process, a replay buffer D with a capacity of 10^5 is employed by each UAV. The target networks are gradually updated using a soft update mechanism defined by the parameter τ. The soft update method ensures that $\hat{\mu}_u$ and $\{\hat{Q}_{u,i}\}_{i=1,2}$ manifest a delayed adaptation to their learned network counterparts, and thus, their gradual path towards synchronization preserves the balanced operation of the learning system. The equation for network weights updating can be summarized as follows:

$$\omega'_{u,i} \leftarrow \tau\omega_{u,i} + (1-\tau)\omega'_{u,i}, \quad i = 1, 2, \tag{21}$$

$$\theta'_u \leftarrow \tau\theta_u + (1-\tau)\theta'_u. \tag{22}$$

The target networks not only facilitate smoother training but also define the optimization target for the critic network as follows:

$$y_u = r_u + \gamma \hat{Q}_{u,i}(s(t+1), a_1(t+1), a_2(t+1), ..., a_U(t+1))|_{a_u(t+1)=\mu'_u(o_u(t))}. \tag{23}$$

As demonstrated in Fig. 3, the training of QEMUOT algorithm can be divided into two phases: the pretraining phase (Section V-B) and the exploration phase (Section V-C). During the pretraining phase, we quickly improve the performance of the action network to a fairly optimal level through expert policy demonstrations, as shown in Section V-A. Meanwhile, a "warm-up" period is implemented for the critic network to prevent potential errors that may lead to a catastrophic decline in training performance during the upcoming exploration phase. As the algorithm transitions into the exploration phase, our model diverges from the expert policy and autonomously explores potentially superior decisions using an ϵ-greedy exploration strategy.

Algorithm 1: Greedy-Sailfish Algorithm

Input: Global state $s(t)$ and the function $g(s(t), a(t))$)
Output: Global action $\{a_u(t) \mid \forall u \in \mathbf{U}\}$

```
for each agent u do
    if ε_u(t) ≠ 0 then
        Select m_f, the UE farthest from u in u's
          offloading queue ;
        Fly towards m_f to prevent offloading failure of
          m_f ;
    else
        Select m_c, the UE closest from m globally ;
        Fly towards m_c to improve m_c's QoS ;
Uitilize the Sailfish optimizer [51] for determining
  {v_u(t), δ_u(t) | ∀u ∈ U} ;
```

A. Expert Algrithom

Due to limited high-quality expert data, we propose an expert algorithm named the Greedy-Sailfish Algorithm (GSF), which combines the Sailfish optimization algorithm with a greedy algorithm. Sailfish optimization is a currently popular metaheuristic algorithm [22], and there have been many successful applications in IoT and MEC scenarios, demonstrating outstanding performance [52], [53]. Therefore, we chose it as our expert algorithm.The greedy algorithm guides flight

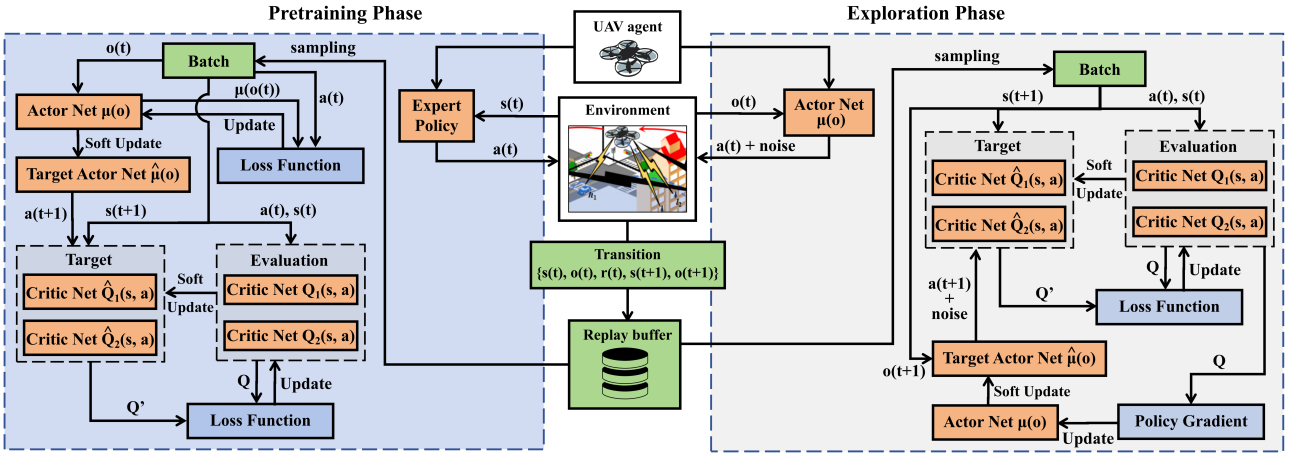


Fig. 3. The framework of QEMUOT algorithm.

direction decision-making, while the subsequent decisions on offloading ratio and flight speed are treated as a simplified constrained optimization problem. We employ the Sailfish algorithm to optimize these decisions.

As summarized in Algorithm 1, if $\epsilon_m(t) \neq 0$, u selects the UE m_f farthest from it in its offloading queue. Subsequently, u fly towards m_f at $V_{\max}$ to prevent offloading failure for m_f. Conversely, if $\epsilon_m(t) = 0$, u selects the no-service UE m_c that is closest to it globally, and fly towards m_c. Given $\{\theta_u(t) \mid \forall u \in \mathbf{U}\}$, the utilization of the Sailfish optimizer for determining flight speed v_u and offloading ratio δ_u involves reformulating the problem along with its associated constraints:

$$\textbf{consider} \quad x = \{v_u(t), \delta_u(t) \mid \forall u \in \mathbf{U}\}, \tag{24}$$

$$\textbf{Min.} \quad f(x) = \sum_{u=1}^{|\mathbf{U}|} C_u(t), \tag{25}$$

$$\textbf{s.t.} \quad (18a) - (18f)$$

The global action execute function is represented as $f(x) = g(s(t), a(t))$, where for a given state $s(t)$ and global action $a(t)$, the function g returns the value of $f(x)$ by stepping forward and backtracking in the computer simulated experimental environment.

B. Pretraining Phase

To tackle the challenge posed by training with randomly initialized network parameters in sparse reward spaces, $E_{\text{pretraining}}$ episodes of pretraining are conducted in the initial stages of training. As depicted in Algorithm 2, during the pre-training phase, the expert policy GSF is utilized instead of the actor network for decision-making. This process generates expert-demonstrations experience samples, which are subsequently stored in the buffer. Each iteration, a random mini-batch B consisting of tuples $(s(t), o_u(t), a_u(t), r_u(t), s(t+1), o_u(t+1))$ is sampled from D for updating the network.

Specifically, behavior cloning pretraining [45] is executed on the actor network, with the learning objective aimed at minimizing the disparity between the decisions made by the policy network and those made by the expert policy. The loss function for behavior cloning is defined as follows:

$$L_{\text{BC}}(\theta_u) = \mathbb{E}\left[\left(\mu_u(o_u(t)) - a_u(t)\right)^2\right], \tag{26}$$

The behavior cloning pretraining for actor network eliminates the inefficiency of exploring better actions only through random interactions with the environment when the policy is poor. Instead, the policies rapidly attain a higher level by imitating the expert algorithm, establishing a strong starting point for learning and facilitating more effective exploration in high-reward regions right from the outset.

Warm-up training is then conducted on the critic network to mitigate excessively biased value estimates from an untrained (cold start) critic network. Such biases could potentially result in the forgetting of a well-performing policy [45]. The loss function for warm-up training using expert-guided experience is defined as follows:

$$L_{\text{Warm-up}}(\omega_{u,i}) = \mathbb{E}\left[\left(Q_{u,i}(s(t), a_1(t), a_2(t), ..., a_U(t)) - y_u\right)^2\right]. \tag{27}$$

C. Exploration Phase

During this stage, the network training basically follows the conventional online MATD3 algorithm. To achieve enhanced performance through fine-tuning and to prevent overfitting to the expert policy, we employ an decaying ϵ-greedy exploration strategy. When the model chooses to explore, Gaussian noise $\xi_\epsilon \sim clip\left(\mathcal{N}\left(0, \sigma_\epsilon^2\right), -c, c\right)$ is introduced to the output of the policy network. The actions are computed as follows:

$$a_u(t) = \begin{cases} \mu_u(o_u(t)), & \text{with probability } 1-\epsilon, \\ \mu_u(o_u(t)) + \xi_\epsilon, & \text{with probability } \epsilon, \end{cases} \tag{28}$$

where the value of ϵ decays gradually as the number of training iterations increases, meaning that the intensity of exploration decreases as the network performance improves.

The expert policy is no longer utilized in this phase. Instead, the QEMUOT algorithm captures experiences by interacting with the environment using its own policy network $\mu_u(o)$. Various methods are employed to reduce the overestimation

Algorithm 2: QEMUOT Algorithm

Input: Randomly Initialize Actor networks $\mu_u(o)$ with weights θ_u and Critic networks $\{Q_{u,i}(s, a_1, a_2, ..., a_U)\}_{i=1,2}$ with weights $\{\omega_{u,i}\}_{i=1,2}$ for each agent u

Output: Target networks $\hat{\mu}_u(o)$ and $\{\hat{Q}_{u,i}\}_{i=1,2}$ with weights θ'_u and $\{\omega'_{u,i}\}_{i=1,2}$ for each agent u

1 **for** $e = 1 \rightarrow E$ **do**
2 Initialize a random process ;
3 **for** $t = 1 \rightarrow T$ **do**
4 **if** $e < E_{pretraining}$ **then**
5 Get global state $s(t)$;
6 Use expert policy (**Algorithm 1**) to determine global action $\{a_u(t)|\forall u \in \mathbf{U}\}$;
7 **else**
8 **for** *each agent* u **do**
9 Get observations $o_u(t)$;
10 Use Actor network $\mu_u(o_u(t))$ to select action $a_u(t)$ with ϵ-greedy noise ;
11 **for** *each agent* u **do**
12 Execute action $a_u(t)$, get reward $r_u(t)$, and new observation $o_u(t+1)$;
13 Store $(s(t), o_u(t), a_u(t), r_u(t), s(t+1), o_m(t+1))$ in replay buffer D ;
14 Sample B batch of data from D ;
15 **for** *each agent* u **do**
16 **if** $e < E_{pretraining}$ **then**
17 Update the Critic network by optimizing loss $L_{\text{Warm-up}}(\omega_{u,i})$;
18 Update the Actor network by optimizing loss $L_{\text{BC}}(\theta_u)$;
19 Update target networks ;
20 **else**
21 Update the Critic network by optimizing loss $L_{\text{E}}(\omega_{u,i})$;
22 **if** t *mod* T_D **then**
23 Update the Actor network by computing gradient $\nabla_{\theta_u} J(\mu_u)$;
24 Update target networks ;

bias of the critic network, including the dual-critic mechanism and adding noise to the predictions of the target actor network. By taking the lower value from the outputs of the two critic networks, the original optimization target for the critic network y_u is reshaped into y'_u as follows:

$$y'_u = r_u + \gamma \min_{i=1,2} \hat{Q}_{u,i}(s(t+1), a_1(t+1), a_2(t+1), ..., a_U(t+1))|_{a_u(t+1)=\mu'_u(o_u(t))+\xi_r}, \tag{29}$$

where $\xi_r \sim clip\left(\mathcal{N}\left(0, \sigma_r^2\right), -c, c\right)$, serves as a regularization.

Therefore, the critic networks are optimized by minimizing a specific loss function as follows:

$$L_{\text{E}}(\omega_{u,i}) = \mathbb{E}\left[\left(Q_{u,i}(s(t), a_1(t), a_2(t), ..., a_U(t)) - y'_u\right)^2\right], \quad i = 1, 2. \tag{30}$$

It is worth noting that, despite the warm-up process during pretraining, we cannot ensure that the current critic network has achieved a sufficiently high performance level. Moreover, the experiences utilized in the warm-up phase are generated by the expert policy rather than the policy network itself, resulting in different distributions. Consequently, during exploration, the critic network might still offer erroneous guidance to the actor network, leading to the degradation of the policy network. To mitigate this issue, the QEMUOT algorithm adopts a delayed updating strategy for the policy network, giving the trainer time to wait for the critic network to stabilize. Specifically, as depicted in line 22 of the Algorithm 2, after every T_D updates of the critic network, the actor network undergoes an update based on the policy gradient defined as:

$$\nabla_{\theta_u} J(\mu_u) = \mathbb{E}\Big[\nabla_{\theta_u} \mu_u(o_u) \nabla_{a_u} Q_{u,1}(s(t), a_1(t), a_2(t), ..., a_U(t))|_{a_u(t)=\mu_u(o_u(t))}\Big]. \tag{31}$$

D. Algorithm Analysis

First, we explore the computational complexity of the expert algorithm we introduced. The worst-case complexity of the greedy algorithm is $O(|\mathbf{U}||\mathbf{M}|)$, where $|\mathbf{U}|$ is the number of UAVs and $|\mathbf{M}|$ is the number of UEs. Moreover, considering the population size N_{pop}, the maximum iterations M_{iter}, the function's dimension D_{ob} and the complexity of evaluating f_{evl}, the computational complexity of the Sailfish Optimizer Algorithm (SFO) can be estimated as $O\left(M_{iter}\left(N_{pop} f_{evl} + D_{ob}\right)\right)$, while $O(D_{ob}) = O(|\mathbf{U}|)$ and $O(f_{evl}) = O(|\mathbf{U}||\mathbf{M}|)$. To sum up, the overall computational complexity of Algorithm 1 is approximately equal to that of SFO, which can be calculated as $O\left(M_{iter}\left(N_{pop} O(|\mathbf{U}||\mathbf{M}|) + O(|\mathbf{U}|)\right)\right)$.

According to the model, the decision-making process for each UAV requires the current location information of all UEs, and there is also the sharing of movement information between UAVs. Therefore, the communication complexity should be $O(|\mathbf{U}| + |\mathbf{M}|)$. The critic and actor networks for each UAV are both DNN networks: the input dimension for the Critic includes state and action information, and the output is the Q value, with dimensions of $(|\mathbf{U}| + |\mathbf{M}| + (|\mathbf{U}| + |\mathbf{K}|)|\mathbf{M}| + 3)$ and 1, respectively. Thus the computational complexity of the critic network can be considered as $O\left(|\mathbf{U}| + |\mathbf{M}| + (|\mathbf{U}| + |\mathbf{K}|)|\mathbf{M}| + 3\right)$. The Actor's input and output dimensions are $(|\mathbf{U}| + |\mathbf{M}|)$ and 3, hence, its computational complexity can be viewed as $O(3(|\mathbf{U}| + |\mathbf{M}|))$. Since CTDE paradigm is used, the overall system complexity of Algorithm 2 is $O(|\mathbf{U}|(|\mathbf{U}| + |\mathbf{M}|)^2)$. Additionally, the training process complexity is also influenced by the batch size and the number of episodes.

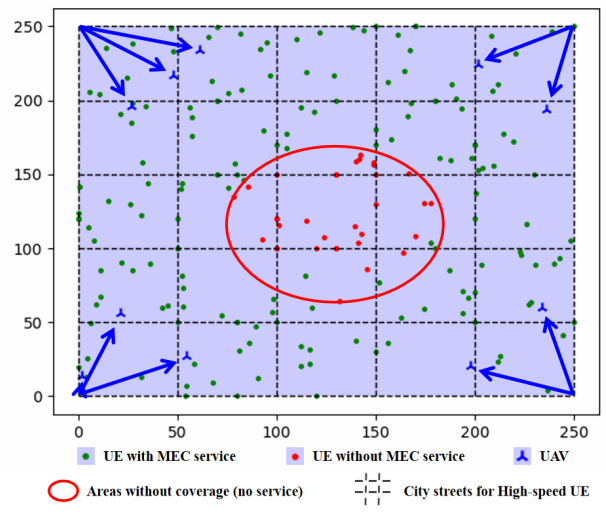


Fig. 4. Visualization of simulation experiment environment.

VI. Experiment

A. Experimental Settings

As depicted in Fig. 4, the simulation area encompasses a square with a side length of $s = 250\,\mathrm{m}$. Each corner hosts a BS, and a grid street network facilitates High-speed UE movement within the area. 10 UAVs take off from the BSs, then work with a flying height of $Z = 100\,\mathrm{m}$ and a maximum speed of $V_{\max} = 10\,\mathrm{m/s}$. In each random process, the simulation iterates for 50 steps starting from the moment the UAVs take off, denoted as $T = 50$. Due to multiple UAVs taking off from the same starting point, and since our experiment neglects the process of UAV ascent, collision constraints are not considered in the first 5 steps, i.e., the value of η_2 is set to 0. The edge angle of the coverage area is set to $\Theta = 50°$ [49]. There are 200 UEs with a ratio of 2:5:1 for three types of UEs ($|\mathbf{H}|$:$|\mathbf{L}|$:$|\mathbf{F}|$). The minimum safe distance between UAVs is set to $D_{\min} = 5\,\mathrm{m}$, and the propulsion energy consumption parameters are referenced from [48]. See Table II for all the main parameters of the simulation network environment.

Fig. 4 illustrates the scenario where UAVs have just taken off from the BSs, beginning to network and cover UEs in the area. We assume our experimental environment is symmetrical, and each BS and UAV is homogeneous. Therefore, we performed a simple fair allocation for the assignment of 10 UAVs to 4 BSs as follows: the BSs in the top-left and bottom-left corners each host 3 UAVs, while the BSs in the top-right and bottom-right corners each accommodate 2 UAVs. The UAVs "converge" from the four corners towards the center, progressively diminishing the size of the unserviced area in the center of the region, thereby augmenting the system's service coverage rate.

The simulations are performed using Python and PyTorch. In both the actor and critic networks, we utilized four fully-connected hidden layers, with $[400, 800, 800, 400]$ neurons. All the networks are trained with a learning rate of 10^{-5} and updated using the Adam Optimizer. For the decaying ϵ-greedy exploration strategy, ξ_ϵ is initialized to 0.8 and decays with a rate of 0.999. Additionally, σ_ϵ and σ_r are set to $0.2 \times c$ and 0.2, respectively. The policy update frequency T_D is fixed at 5.

TABLE II
NETWORK ENVIRONMENT PARAMETERS.

Parameters	Value
Side length of simulation area s	$250\,\mathrm{m}$
Flying height of UAVs Z	$100\,\mathrm{m}$
Maximum velocity of UAVs $V_{\max}$	$10\,\mathrm{m/s}$
Minimum velocity of UAVs $V_{\min}$	$0\,\mathrm{m/s}$
Elevation angle of UAVs Θ	$50°$ [49]
Ratio of 3 types of UEs $\|\mathbf{H}\|$:$\|\mathbf{L}\|$:$\|\mathbf{F}\|$	2 : 5 : 1
Minimum safe distance between UAVs $D_{\min}$	$5\,\mathrm{m}$
Blade profile power in hover P_0	$79.86\,\mathrm{W}$
Induced power in hover P_i	$88.63\,\mathrm{W}$
Tip speed of rotor spade V_{tip}	$120\,\mathrm{m/s}$
Mean rotor induced velocity in hover v_0	$4.03\,\mathrm{m/s}$
Fuselage drag ratio d_0	0.6
Tip speed of rotor spade ρ	$1.225\,\mathrm{kg/m^3}$
Rotor disc area s_d	$0.503\,\mathrm{m^2}$
Rotor solidity r_s	0.05
Size of task data D_m	$\mathcal{N}(8, 4)$ Mbits
Number of CPU cycles required for each bit of data $C_m(t)$	$\mathcal{N}(150, 50)$ cycles/bit
Constant velocity of $\mathbf{H}$ V_h	$10\,\mathrm{m/s}$
Fixed stay time of $\mathbf{H}$ t_h	$10\,\mathrm{s}$
Exploration parameter of $\mathbf{H}$ ρ_h	0.2
Exploration parameter of $\mathbf{H}$ ψ	0.5
Memory level of $\mathbf{L}$ α	0.8
Symptotic mean of $\mathbf{L}$'s velocity $\overline{\boldsymbol{v}}_l$	$2\,\mathrm{m/s}$
Standard deviation of $\mathbf{L}$'s velocity $\overline{\sigma}_l$	0.2
Maximum task capacity of UAVs $\epsilon_{\max}$	10
Attenuation factors for LoS links μ_{LoS}	2 dB
Attenuation factors for NLoS links μ_{NLoS}	20 dB
Carrier frequency f_c	3 GHz
Bandwidth of UAVs B_U	10MHz
Bandwidth of BSs B_K	10MHz
Noise power for UAV communication σ_U^2	100 dBm
Transmitting power of UEs P_M	20 dBm
Receiving power of UAVs P_U^r	100 dBm
Transmitting power of UAVs P_U^t	100 dBm
Total computing resources of UAVs F_U	20 GHz
Computing resources for each task of BSs F_K	30 GHz

Five baseline algorithms are conducted:

- **Random:** In which each action is chosen randomly and follows a uniform distribution.
- **Naive-Greedy:** The greedy algorithm, as discussed in Section V-A, is employed for flight direction selection. However, it's important to note that the flight speed $v_u(t)$ is consistently set to the maximum value $V_{\max}$, and the offloading ratio $\delta_u(t)$ remains fixed at 50%.
- **GSF:** As illustrated in Section V-A. We set the initial population N_{pop} to 30. The algorithm process is repeated for $M_{iter} = 500$ iterations. And parameter values of A_{SF} and ϵ_{SF} are considered, 4 and 0.001, respectively [22].
- **MARL:** We also conduct training with conventional **MADDPG** and **MATD3** aproach. Furthermore, we maintain the same network structure, learning rate, optimizer,

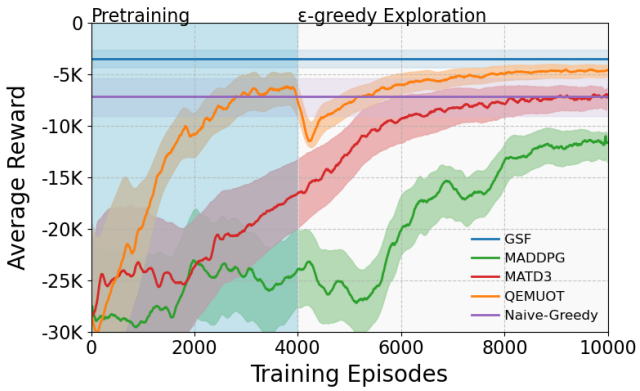


Fig. 5. Average system reward.

and epsilon exploration strategy as those used in the QEMUOT algorithm, along with other main parameters to ensure a fair comparison.

B. Experimental Results

*1) **Trainning performance of the MARL algorithms**:* Fig. 5 displays the training curves of the reinforcement learning algorithms. The QEMUOT algorithm achieves a 36.62% and 62.47% improvement in reward compared to the conventional MADDPG and MATD3 algorithms, respectively. Compared to MATD3, QEMUOT only takes 36.59% of episodes in pretraining to converge to the reward of Naive-Greedy algorithm. When transitioning from the pretraining phase to the exploration phase, the policy network experienced a slight performance degradation, approximately 23.89% of the previous training reward increments. It is noteworthy that no further performance degradation occurred. Subsequently, after only 1000 episodes, it quickly recovered to performance comparable to that of the Naive-Greedy algorithm, and further explored potentially superior solutions, surpassing all other MARL algorithms in the baselines.

From an overall performance perspective, QEMUOT did not achieve a higher average reward than GSF, our expert algorithm. This gap is primarily due to the difference in information input between the two. GSF utilizes the SFO meta-heuristic algorithm, which requires knowledge of the objective function and allows for repeated substitutions for optimization. In other words, the GSF algorithm benefits from additional environmental information for the next time slot, which is unknown to QEMUOT. The policy network of QEMUOT must make decisions based solely on the current state. Furthermore, this discrepancy highlights the highly unpredictable environmental changes in this scenario and the diverse behavior patterns of UEs. Consequently, the observation space for the agent becomes extremely complex, suggesting that there is still room for improvement in our policy network structure.

*2) **Algorithm time cost comparison**:* It's crucial to note that, as depicted in the Table III, although the reward achieved by the QEMUOT algorithm in the experiments did not surpass that of our designed expert algorithm GSF, the QEMUOT algorithm exhibits significant superiority in practical usability

TABLE III
AVERAGE DECISION TIME PER SYSTEM ITERATION PER AGENT (MS)

UEs	Random	Naive-Greedy	GSF	QEMUOT
200	5.04×10^{-3}	5.24×10^{-2}	1.03×10^{2}	5.33×10^{0}
400	5.04×10^{-3}	1.02×10^{-1}	1.74×10^{2}	7.51×10^{0}
600	5.04×10^{-3}	1.55×10^{-1}	2.68×10^{2}	8.93×10^{0}
800	5.04×10^{-3}	2.08×10^{-1}	3.53×10^{2}	1.01×10^{1}
1000	5.04×10^{-3}	2.53×10^{-1}	4.54×10^{2}	1.25×10^{1}

compared to GSF. Firstly, Table III presents a comparison of the average time taken for each decision in the simulation by the algorithms. It is evident that the decision time of the QEMUOT algorithm remains within an acceptable range, typically below 10 milliseconds, whereas the decision time required by GSF consistently exceeds hundreds of milliseconds. Another fundamental reason is that GSF requires the objective function to be known and can be repeatedly substituted for optimization. In the experimental simulation, we can repeatedly substitute action decisions, i.e., the solution to the problem, into the virtual environment for optimization by stepping forward and backtracking to obtain the objective function value. However, in practical applications, stepping forward and backtracking is practically impossible. Therefore, this algorithm lacks practical usability, which indirectly highlights an advantage of MARL methods.

*3) **Performance with different numbers of UEs and UAVs**:* Furthermore, we conduct experiments by varying the number of UAVs and UEs, as depicted in Fig. 6 and Fig. 7. The simulation results consistently demonstrate that our algorithm outperforms baselines across various metrics. Service Coverage Rate refers to the proportion of users within UAV coverage, which reflects UAVs' basic network deployment and service coverage capabilities. As observed, the service coverage of QEMUOT exceeds 95%, reaching parity with GSF and surpassing all other baseline algorithms. Notably, compared to traditional MARL algorithms, QEMUOT demonstrates a distinct energy-efficient advantage, consistently exhibiting the lowest system energy consumption across all scenarios.

With an increase in the number of UEs, the energy consumption of traditional MADDPG algorithms exceeds that of the Greedy algorithm. In contrast, the energy consumption of QEMUOT not only remains at a low level, but even surpassing GSF by 5.26% in scenarios with 1000 UEs as shown in Fig. 6. This energy efficiency translates to extended UAV endurance and reduced operational costs.

QEMUOT's performance is particularly noteworthy in reducing offloading failure rates, which significantly contributes to achieving the lowest average user latency performance. Compared to MADDPG and MATD3 algorithm, it reduces latency by 28.13% to 40.67% and offloading failure rates by 22.23% to 44.74%, respectively. Fig. 8 shows the cumulative distribution functions of the offloading failure rate accumulated by all algorithms in the primary experimental environment, with the dashed line indicating the mean value of the offloading failure rate. It can be observed that the QEMUOT algorithm ensures lower offloading failure rates at more instances and achieves the lowest average offloading

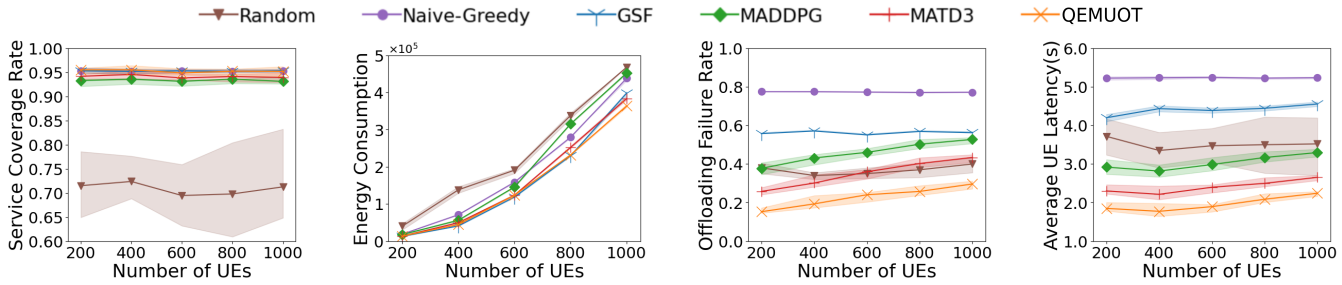


Fig. 6. Performance with different numbers of UEs.

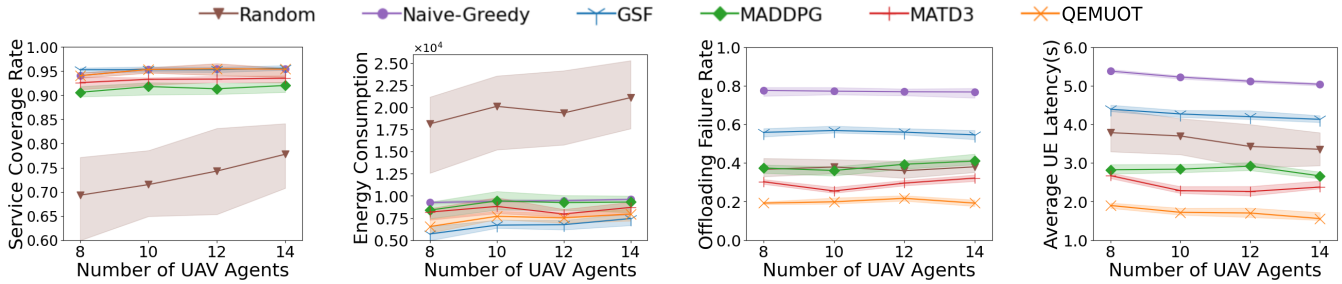


Fig. 7. Performance with different numbers of UAVs.

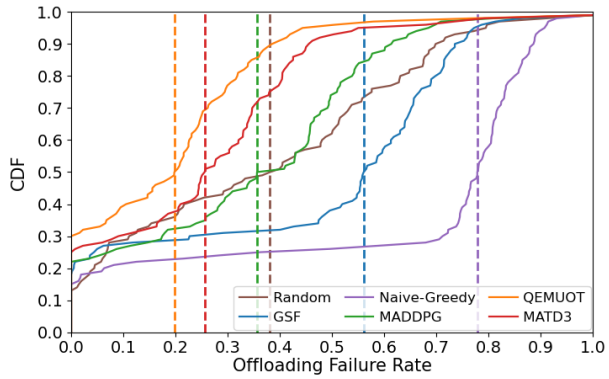


Fig. 8. Cumulative distribution of the offloading failure rate.

failure rate among all baseline algorithms. The reduction in offloading failure is attributed to QEMUOT's optimized task scheduling and resource allocation mechanisms, which also contribute to lower system energy consumption by minimizing unnecessary task retransmissions.

It is important to note that although the offloading failure rate of the Random algorithm is significantly lower than that of the GSF expert algorithm, this does not necessarily indicate that the random algorithm is more effective in avoiding offloading failures compared to the GSF algorithm. This phenomenon actually occurs because the premise of offloading failure is the initiation of task offloading. As depicted in the first diagrams on the left of Fig. 6 and Fig. 7, the Random algorithm fails to achieve high user coverage, resulting in the inability to connect to the UAV server initially, thus avoiding offloading failure incidents altogether. In contrast, the QEMUOT algorithm, which achieves high user coverage comparable to the GSF algorithm, also ensures a low offloading failure rate. This demonstrats the positive impact of our designed reward mechanism, providing a compelling solution for enhancing QoS and mitigating offloading failure issues.

*4) **Performance with different preferences for energy consumption and QoS**:* Finally, the algorithm's flexibility and controllability are further demonstrated by the ability to fine-tune the preference between energy consumption and QoS through adjustments to the weights ω_1 and ω_2, as illustrated in Fig. 9. For QEMUOT, by increasing ω_1, the system's energy consumption can be decreased by an additional 20.55%, albeit at the cost of sacrificing QoS. Conversely, increasing ω_2 prioritizes QoS improvement over energy savings, resulting in a further 2.74% improvement in service coverage rate, a 10.16% reduction in offloading failure rate, and an 11.24% decrease in latency. This underscores the algorithm's adaptability to various optimization objectives and its capability to strike a balance between conflicting performance metrics.

The fine-tuning capability of QEMUOT allows for the optimization of system performance according to dynamic environments and user demands. By adjusting ω_1 and ω_2 appropriately, operators can effectively manage the balance between energy efficiency and service quality to meet diverse application requirements. This flexibility positions QEMUOT as an ideal solution for future MEC systems, where effective resource management and excellent QoS are crucial.

VII. Discussion

The effectiveness of our proposed algorithm is evident from the experimental results. However, several issues require further discussion and clarification.

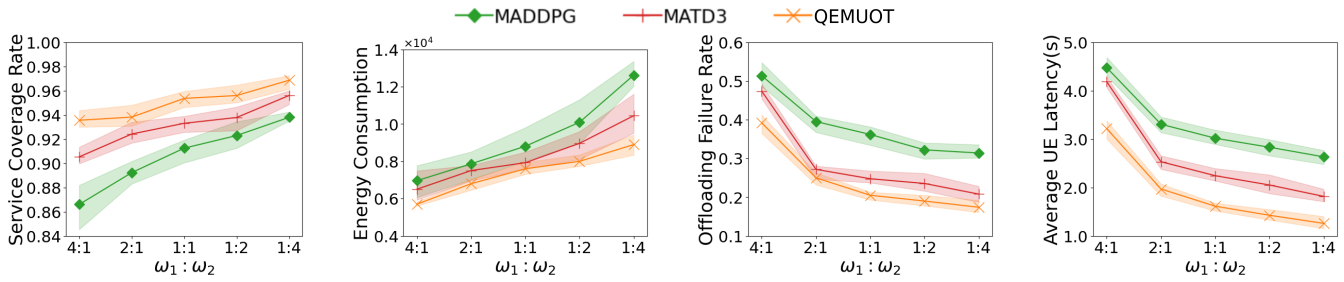


Fig. 9. Performance with different preferences for energy consumption and QoS.

*1) **Mitigating the "Slippery Slope" at the start of exploration**:* Upon detaching from the expert strategy's guidance, the reward mechanism shifts from behavior cloning to autonomous exploration. During this phase, the critic network is susceptible to significantly biased value estimations, leading to poor reward signals that can cause the initially effective strategy to be forgotten. This problem becomes apparent as the training performance shows a "slippery slope" when the episode count hits 4000.

However, it is apparent that this decline was promptly mitigated. This improvement is attributed to the warm-up operation applied to the critic network and the delayed updating process in actor network training. These measures prevented further catastrophic degradation of the network. This outcome highlights the effectiveness of the proposed pretraining algorithm.

*2) **Practical implementation**:* To deploy the system described in our work in real-world scenarios, several critical aspects must be considered:

- The system's task offloading service follows the time slot partition protocol, dividing operational time into distinct slots. This method ensures organized task management, efficient resource allocation, and improved system performance.
- QEMUOT's scheduling decisions rely on GPS positioning data for all UEs and the task offloading relationships. UAV clusters exchange location and operational status information. Therefore, protocols such as MQTT or CoAP can be used for efficient real-time communication [54].
- UAVs depend on LoS communication links to maintain reliable connections, requiring optimal flight altitudes and distribution. Currently, mature regulations on the density, flight altitude, and communication coverage angle of urban drone clusters are lacking. Parameters from previous studies can be used [49].

By addressing these gaps, our proposed algorithm can be practically deployed, guiding our future work.

VIII. Conclusion and Future Work

Our work focused on addressing challenges in Multi-UAV-assisted MEC. We have introduced a composite UE mobility model to refine system modeling and proposed an MDRL-based algorithm, namely QEMUOT. Notably, the offloading failure problem was tackled for the first time in UAV-assisted MEC. Our study contends that due to the distinctive mobility of UAVs, UAV-MEC systems leads to a paradigm shift from conventional user-side offloading decision designs to the optimization of server-side scheduling mechanisms. Experimental simulations illustrated that the proposed QEMUOT algorithm outperformed baseline algorithms in terms of QoS, energy consumption reduction, and greater scalability in large networks. Our algorithm exhibited rapid convergence and low overhead, highlighting its practical applicability. Future work will consider using containers to virtualize UAV services and further optimize offloading costs from the perspective of the container layer. Furthermore, to address the challenge of highly complex environment spaces in reinforcement learning methods, replacing the policy network with a diffusion model could be a promising research direction.

References

[1] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[2] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.

[3] T. Pathirana and G. Nencioni, "Availability model of a 5g-mec system," in *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2023, pp. 1–10.

[4] Y. Yazid, I. Ez-Zazi, A. Guerrero-Gonzalez, A. El Oualkadi, and M. Arioua, "Uav-enabled mobile edge-computing for iot based on ai: A comprehensive review," *Drones*, vol. 5, no. 4, p. 148, 2021.

[5] L. Wang, K. Wang, C. Pan *et al.*, "Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 73–84, 2020.

[6] W. Lee and T. Kim, "Multi-agent reinforcement learning in controlling offloading ratio and trajectory for multi-uav mobile edge computing," *IEEE Internet of Things Journal*, 2023.

[7] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 360–374, 2020.

[8] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5g mobile edge computing: Architectures, applications, and technical aspects," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021.

[9] M. Dai, Y. Wu, L. Qian, Z. Su, B. Lin, and N. Chen, "Uav-assisted multi-access computation offloading via hybrid noma and fdma in marine networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 1, pp. 113–127, 2022.

[10] S. D. A. Shah, M. A. Gregory, S. Li, R. dos Reis Fontes, and L. Hou, "Sdn-based service mobility management in mec-enabled 5g and beyond vehicular networks," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13 425–13 442, 2022.

[11] C. Li, H. Wang, and R. Song, "Intelligent offloading for noma-assisted mec via dual connectivity," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2802–2813, 2020.

[12] T. Tan, M. Zhao, and Z. Zeng, "Joint offloading and resource allocation based on uav-assisted mobile edge computing," *ACM Transactions on Sensor Networks (TOSN)*, vol. 18, no. 3, pp. 1–21, 2022.

[13] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2516–2529, 2015.

[14] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a uav-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, 2017.

[15] S. Sun, G. Zhang, H. Mei, K. Wang, and K. Yang, "Optimizing multi-uav deployment in 3-d space to minimize task completion time in uav-enabled mobile edge computing systems," *IEEE Communications Letters*, vol. 25, no. 2, pp. 579–583, 2020.

[16] J. Ji, K. Zhu, C. Yi, and D. Niyato, "Energy consumption minimization in uav-assisted mobile-edge computing systems: Joint resource allocation and trajectory design," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8570–8584, 2020.

[17] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration modeling and learning algorithms for containers in fog computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 712–725, 2018.

[18] J. Ackermann, V. Gabler, T. Osa, and M. Sugiyama, "Reducing overestimation bias in multi-agent domains using double centralized critics," *arXiv preprint arXiv:1910.01465*, 2019.

[19] L. Zhang and N. Ansari, "Latency-aware iot service provisioning in uav-aided mobile-edge computing networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 573–10 580, 2020.

[20] X. Lou, J. Zhang, Y. Du, C. Yu, Z. He, and K. Huang, "Leveraging joint-action embedding in multi-agent reinforcement learning for cooperative games," *IEEE Transactions on Games*, 2023.

[21] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 2469–2478.

[22] S. Shadravan, H. R. Naji, and V. K. Bardsiri, "The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 80, pp. 20–34, 2019.

[23] S. Huang, J. Zhang, and Y. Wu, "Altitude optimization and task allocation of uav-assisted mec communication system," *Sensors*, vol. 22, no. 20, p. 8061, 2022.

[24] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in uav-enabled mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4576–4589, 2019.

[25] L. X. Nguyen, Y. K. Tun, T. N. Dang, Y. M. Park, Z. Han, and C. S. Hong, "Dependency tasks offloading and communication resource allocation in collaborative uav networks: A metaheuristic approach," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 9062–9076, 2023.

[26] X. Liu, Y. Liu, and Y. Chen, "Reinforcement learning in multiple-uav networks: Deployment and movement design," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8036–8049, 2019.

[27] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for uav-mounted mobile edge computing with deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5723–5728, 2020.

[28] 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.

[29] W. Lu, Y. Mo, Y. Feng, Y. Gao, N. Zhao, Y. Wu, and A. Nallanathan, "Secure transmission for multi-uav-assisted mobile edge computing based on reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1270–1282, 2022.

[30] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in uav-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[31] M. Sánchez and P. Manzoni, "Anejos: a java based simulator for ad hoc networks," *Future generation computer systems*, vol. 17, no. 5, pp. 573–583, 2001.

[32] 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.

[33] J. Kraaier and U. Killat, "Random direction or random waypoint? a comparison of mobility models for urban environments," *European Transactions on Telecommunications*, vol. 19, no. 8, pp. 879–894, 2008.

[34] W. Li, X. Chen, and S. Lu, "Content synchronization using device-to-device communication in smart cities," *Computer Networks*, vol. 120, pp. 170–185, 2017.

[35] B. Liang and Z. J. Haas, "Predictive distance-based mobility management for pcs networks," in *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, vol. 3. IEEE, 1999, pp. 1377–1384.

[36] S. Zhang, L. Zhang, F. Xu, S. Cheng, W. Su, and S. Wang, "Dynamic deployment method based on double deep q-network in uav-assisted mec systems," *Journal of Cloud Computing*, vol. 12, no. 1, p. 130, 2023.

[37] C. Song, T. Koren, P. Wang, and A.-L. Barabási, "Modelling the scaling properties of human mobility," *Nature physics*, vol. 6, no. 10, pp. 818–823, 2010.

[38] X. Ge, J. Ye, Y. Yang, and Q. Li, "User mobility evaluation for 5g small cell networks based on individual mobility model," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 528–541, 2016.

[39] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," *International journal on advances in systems and measurements*, vol. 5, no. 3&4, 2012.

[40] L. Zhao, K. Yang, Z. Tan, H. Song, A. Al-Dubai, A. Y. Zomaya, and X. Li, "Vehicular computation offloading for industrial mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7871–7881, 2021.

[41] Z. Tang, F. Mou, J. Lou, W. Jia, Y. Wu, and W. Zhao, "Multi-user layer-aware online container migration in edge-assisted vehicular networks," *IEEE/ACM Transactions on Networking*, 2024.

[42] Z. Tang, J. Lou, and W. Jia, "Layer dependency-aware learning scheduling algorithms for containers in mobile edge computing," *IEEE Transactions on Mobile Computing*, 2022.

[43] T. Saber, C. Cachard, and A. Ventresque, "Ronin: a sumo interoperable mesoscopic urban traffic simulator," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2020, pp. 1104–1111.

[44] Z. Ning, Y. Yang, X. Wang, Q. Song, L. Guo, and A. Jamalipour, "Multi-agent deep reinforcement learning based uav trajectory optimization for differentiated services," *IEEE Transactions on Mobile Computing*, 2023.

[45] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao *et al.*, "Jump-start reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 34 556–34 583.

[46] Y. Qiu, Y. Jin, L. Yu, J. Wang, Y. Wang, and X. Zhang, "Improving sample efficiency of multi-agent reinforcement learning with non-expert policy for flocking control," *IEEE Internet of Things Journal*, 2023.

[47] R. He, B. Ai, G. L. Stüber, and Z. Zhong, "Mobility model-based non-stationary mobile-to-mobile channel modeling," *IEEE Transactions on Wireless Communications*, vol. 17, no. 7, pp. 4388–4400, 2018.

[48] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing uav," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[49] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu, "3-d placement of an unmanned aerial vehicle base station (uav-bs) for energy-efficient maximal coverage," *IEEE Wireless Communications Letters*, vol. 6, no. 4, pp. 434–437, 2017.

[50] F. A. Oliehoek, C. Amato *et al.*, *A concise introduction to decentralized POMDPs*. Springer, 2016, vol. 1.

[51] T. Nguyen, N. Tran, B. M. Nguyen, and G. Nguyen, "A resource usage prediction system using functional-link and genetic algorithm neural network for multivariate cloud metrics," in *2018 IEEE 11th conference on service-oriented computing and applications (SOCA)*. IEEE, 2018, pp. 49–56.

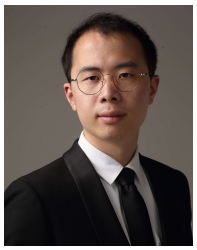
[52] J. Deepa, S. A. Ali, and S. Hemamalini, "Intelligent energy efficient vehicle automation system with sensible edge processing protocol in internet of vehicles using hybrid optimization strategy," *Wireless Networks*, vol. 29, no. 4, pp. 1685–1701, 2023.

[53] M. K. Rajoriya and C. P. Gupta, "Sailfish optimization-based controller selection (sfo-cs) for energy-aware multi-hop routing in software defined wireless sensor network (sdwsn)," *International Journal of Information Technology*, vol. 15, no. 7, pp. 3935–3948, 2023.

[54] E. Longo, A. E. Redondi, M. Cesana, A. Arcia-Moret, and P. Manzoni, "Mqtt-st: a spanning tree protocol for distributed mqtt brokers," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

Jiajie Yin is currently pursuing the B.Sc. degree in data science from Beijing Normal University, Zhuhai, China. His research interests include multi-agent systems, deep learning, reinforcement learning, edge computing, Internet of Things and data mining.

Zhiqing Tang received the B.S. degree from School of Communication and Information Engineering, University of Electronic Science and Technology of China, China, in 2015 and the Ph.D. degree from Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2022. He is currently an Assistant Professor with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University, China. His current research interests include edge computing, resource scheduling, container scheduling, and reinforcement learning.



Jiong Lou received the B.S. degree and Ph.D. degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2016 and 2023. Since 2023, he has held the position of Research Assistant Professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. He has published more than ten papers in leading journals and conferences (e.g., ToN, TMC and TSC). His current research interests include edge computing, task scheduling and container management. He has served as a reviewer for CN, JPDC, IoT-J, and ICDCS.

Jianxiong Guo received his Ph.D. degree from the Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA, in 2021, and his B.E. degree from the School of Chemistry and Chemical Engineering, South China University of Technology, Guangzhou, China, in 2015. He is currently an Associate Professor with the Advanced Institute of Natural Sciences, Beijing Normal University, and also with the Guangdong Key Lab of AI and Multi-Modal Data Processing, BNU-HKBU United International College, Zhuhai, China. He is a member of IEEE/ACM/CCF. His research interests include social networks, wireless sensor networks, combinatorial optimization, and machine learning.

Hui Cai received her Ph.D. degree in Computer Science and Technology from Shanghai Jiao Tong University in 2020. She is currently an Assistant Professor in College of Computer at Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, China. She has authored papers in research related international conferences and journals, such as IEEE INFOCOM, IEEE TPDS, IEEE/ACM IWQoS, Elsevier Computer Networks. Her research interests include data trading, incentive mechanism design, mobile crowd sensing and game theory.

Xiaoming Wu received the M.Eng. degree in computer science and technology from Shandong University, Jinan, China, in 2006, and the Ph.D. degree in Software Engineering from Shandong University of Science and Technology in 2017. Since 2006, he has been with the Shandong Computer Science Center, where he is currently a full professor. He also serves as the director of the Faculty of Computer Science and technology at Qilu University of Technology (Shandong Academy of Sciences), China. His research interests include cyber security, industrial Internet, data security, and privacy protection.

Tian Wang (Senior Member, IEEE) received his BSc and MSc degrees in Computer Science from the Central South University in 2004 and 2007, respectively. He received his PhD degree from the City University of Hong Kong in Computer Science in 2011. Currently, he is a professor with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University. His research interests include the Internet of Things, Edge Computing, and Mobile Computing. He has 27 patents and has published more than 200 papers in high-level journals and conferences. He has more than 14000 citations, according to Google Scholar. His H-index is 68. He has managed 6 national natural science projects (including 2 sub-projects) and 4 provincial-level projects.

Weijia Jia (Fellow, IEEE) is currently a Chair Professor, Director of BNU-UIC Institute of Artificial Intelligence and Future Networks, Beijing Normal University (Zhuhai) and VP for Research of BNU-HKBU United International College (UIC) and has been the Zhiyuan Chair Professor of Shanghai Jiao Tong University, China. He was the Chair Professor and the Deputy Director of State Kay Laboratory of Internet of Things for Smart City at the University of Macau. He received BSc/MSc from Center South University, China, in 82/84 and Master of Applied Sci./PhD from Polytechnic Faculty of Mons, Belgium in 92/93, respectively, all in computer science. From 93-95, he joined German National Research Center for Information Science (GMD) in Bonn (St. Augustine) as a research fellow. From 95-13, he worked at City University of Hong Kong as a professor. His contributions have been recognized as optimal network routing and deployment, anycast and QoS routing, sensors networking, AI (knowledge relation extractions; NLP, etc.), and edge computing. He has over 600 publications in the prestige international journals/conferences and research books, and book chapters. He has received the best product awards from the International Science & Tech. Expo (Shenzhen) in 2011/2012 and the 1st Prize of Scientific Research Awards from the Ministry of Education of China in 2017 (list 2). He has served as area editor for various prestige international journals, chair and PC member/skeynote speaker for many top international conferences. He is the Fellow of IEEE and the Distinguished Member of CCF.