Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.071865

ARTICLE

DRL-Based Task Scheduling and Trajectory Control for UAV-Assisted MEC Systems

Sai Xu[1,*], Jun Liu[1,*], Shengyu Huang[1] and Zhi Li[2]

[1]School of Computer Science and Engineering, Northeastern University, Shenyang, 110169, China
[2]School of Information Science and Engineering, Shenyang Ligong University, Shenyang, 110159, China

*Corresponding Authors: Sai Xu. Email: iexusai@163.com; Jun Liu. Email: liujun@cse.neu.edu.cn

Received: 13 August 2025; Accepted: 28 October 2025; Published: 12 January 2026

ABSTRACT: In scenarios where ground-based cloud computing infrastructure is unavailable, unmanned aerial vehicles (UAVs) act as mobile edge computing (MEC) servers to provide on-demand computation services for ground terminals. To address the challenge of jointly optimizing task scheduling and UAV trajectory under limited resources and high mobility of UAVs, this paper presents PER-MATD3, a multi-agent deep reinforcement learning algorithm with prioritized experience replay (PER) into the Centralized Training with Decentralized Execution (CTDE) framework. Specifically, PER-MATD3 enables each agent to learn a decentralized policy using only local observations during execution, while leveraging a shared replay buffer with prioritized sampling and centralized critic during training to accelerate convergence and improve sample efficiency. Simulation results show that PER-MATD3 reduces average task latency by up to 23%, improves energy efficiency by 21%, and enhances service coverage compared to state-of-the-art baselines, demonstrating its effectiveness and practicality in scenarios without terrestrial networks.

KEYWORDS: Mobile edge computing; deep reinforcement learning; task offloading; resource allocation; trajectory control

1 Introduction

In recent years, frequent natural disasters, emergencies, and regional conflicts have severely damaged terrestrial communication and cloud computing infrastructure, making it difficult for user terminals to obtain timely and effective processing for their computation-intensive and delay-sensitive tasks [1,2]. Air-ground integrated networks leverage the high mobility and rapid deployment capabilities of unmanned aerial vehicles (UAVs), employing UAVs as aerial edge servers to provide ground terminals with low-latency computing and reliable communication services [3–5]. Owing to its practical relevance and technical promise, this paradigm has attracted significant attention from both academia and industry [6,7].

In UAV-assisted mobile edge computing (UAV-assisted MEC) systems, several fundamental challenges remain in achieving efficient and reliable task scheduling. First, UAVs are subject to limited onboard energy, which constrains their flight time and computational capacity, thereby necessitating a balance between energy consumption and processing performance. Second, in multi-UAV scenarios, trajectory planning becomes increasingly complex, as each UAV must coordinate its path while satisfying energy, communication, and collision-avoidance constraints. Collectively, these challenges highlight the need for task scheduling mechanism that can adapt to dynamic environmental conditions while maintaining energy efficiency.

Existing research related to this study primarily focuses on the joint optimization of task latency and energy consumption, along with unmanned aerial vehicle (UAV) trajectory control, which are predominantly addressed using deep reinforcement learning (DRL)-based approaches.

To jointly minimize task latency and energy consumption in dynamic environments. Tang et al. [8] deploy unmanned aerial vehicles (UAVs) as mobile edge computing nodes in an air-ground collaborative network to deliver AI services, enabling rapid response to ground users' task requests and reducing overall system energy consumption. Zhao et al. [9] address dynamic network environments by considering the mobility of both end devices and UAVs, and propose a PER-DDPG-based task offloading method that jointly optimizes task offloading success rate and system energy consumption, thereby improving overall system efficiency. Li et al. [10] present a UAV-assisted MEC model based on MADDPG, which jointly optimizes task offloading ratios and the UAV's 3D trajectory to achieve a favorable trade-off among system latency, energy consumption, and throughput. Paper [11] introduces a fairness-aware optimization framework for a hybrid dual-layer UAV architecture combining fixed-wing and rotary-wing UAVs, and employs the MATD3 algorithm to minimize system latency while ensuring fair service delivery among users. Ma et al. [12] develop a blockchain-assisted edge resource allocation framework that combines DRL-based server bidding with Stackelberg game-driven incentive mechanisms, enabling efficient and cost-aware resource trading in distributed edge environments.

UAV trajectory control is essential for enhancing computation performance and energy efficiency in UAV-assisted MEC systems. Seid et al. [13] jointly optimized trajectories and resource allocation using TD3 to minimize energy and delay, and Yin et al. [14] developed QEMUOT, which leverages MATD3 to co-optimize UAV paths and offloading ratios for improved coverage and efficiency. Gao et al. [15] propose a multi-objective reinforcement learning algorithm that jointly optimizes task latency and system energy consumption by controlling UAV trajectories and task offloading decisions. Wu et al. [16] enhanced coordination efficiency using an attention-based DRL approach for joint offloading and resource allocation. Zhang et al. [17] propose a multi-agent deep reinforcement learning-based strategy for joint task offloading and resource allocation in air-to-ground networks, enabling a UAV swarm to provide computation offloading services for ground IoT devices.

Although significant progress has been made in task scheduling and trajectory control for UAV-assisted MEC systems, most existing approaches focus primarily on minimizing either task latency or energy consumption in isolation. To address this challenge, this paper proposes PER-MATD3, a multi-agent deep reinforcement learning algorithm that jointly optimizes task latency, energy efficiency, and user coverage. PER-MATD3 adopts the centralized training with decentralized execution (CTDE) paradigm, allowing agents to learn coordinated policies using global information during training while executing based only on local observations. It further incorporates prioritized experience replay (PER) to accelerate learning by focusing on high-impact experiences, thereby improving both multi-agent coordination and training efficiency. The main contributions are as follows:

(1) We model a UAV-assisted MEC system with randomly distributed user terminals, multiple UAVs, and a ground base station. Terminals generate delay-sensitive, computation-intensive tasks, which can be processed by UAVs or offloaded via UAVs to the base station.

(2) We propose PER-MATD3, a joint optimization algorithm based on the Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3) with prioritized experience replay, improving stability, sample efficiency.

(3) Simulation results show that PER-MATD3 achieves fast convergence and effectively reduces task latency, energy consumption, and user coverage.

2 System Model and Problem Description

In emergency scenarios such as natural disasters, large-scale accidents, or temporary public events, terrestrial cloud infrastructure may be damaged or unavailable, leaving ground user terminals unable to access reliable computational resources. To address this challenge, this paper investigates a UAV-assisted mobile edge computing (MEC) system, as illustrated in Fig. 1, comprising a set of user terminals $\mathcal{M} = \{1, 2, \cdots, M\}$, a set of UAVs $\mathcal{U} = \{1, 2, \cdots, U\}$, and a ground base station. User terminals are assumed to be stationary with limited computational capabilities, making them unable to independently process all computation-intensive and delay-sensitive tasks. UAVs, deployed at low altitude as edge computing servers, follow designated trajectories to provide computation offloading services for user terminals within their coverage. Tasks exceeding the UAVs' computational capacity are further offloaded to the ground base station via communication links, enabling a hierarchical end-air-ground computing collaboration.

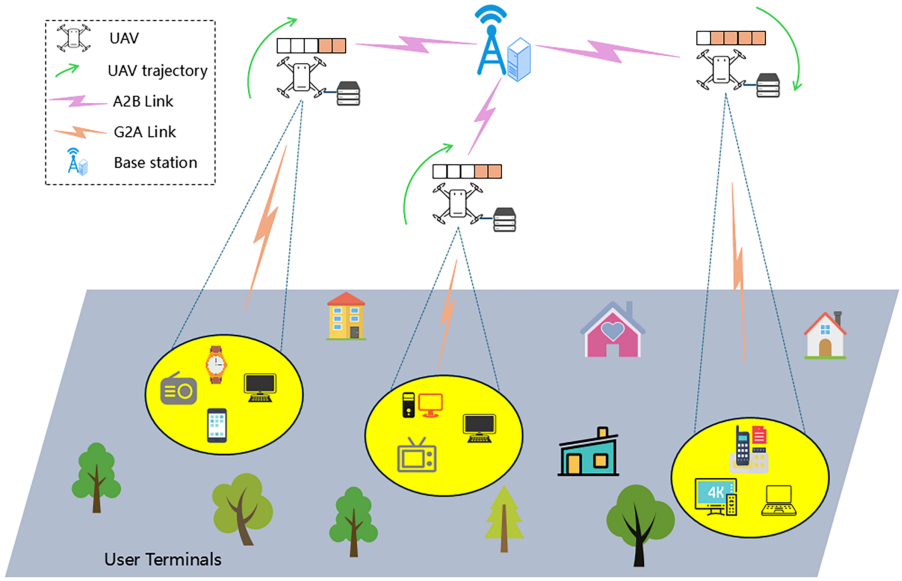


Figure 1: UAV-assisted MEC system architecture

The system operates over discrete time slots $\mathcal{T} = \{1, 2, \ldots, T\}$, each of duration τ, serving as the basic unit for state updates and task scheduling. At each time slot $t \in T$, user terminals generate tasks following a Poisson process. Each task is defined as $l_m(t) = (d_m(t), c_m(t), \zeta_m(t))$, where $d_m(t)$ is the data size (in bits), $c_m(t)$ is the required CPU cycles per bit, and $\zeta_m(t)$ is the delay tolerance. Tasks violating $\zeta_m(t)$ are discarded.

2.1 UAV Motion Model

In this paper, UAVs are assumed to fly at a fixed altitude H, with control focused on horizontal trajectories. The horizontal position of UAV u at time t is denoted as $v_u(t) = [x_u(t), y_u(t)]$. UAV motion in each time slot $t \in T$ is governed by heading angle $\phi_u(t) \in [0, 2\pi]$ and flight speed $\delta_u(t)$, which jointly determine its trajectory. The horizontal position at $t+1$ is updated as follows:

$$\begin{cases} x_u(t+1) = x_u(t) + \delta_u(t) \cdot \tau \cdot \cos(\phi_u(t)) \\ y_u(t+1) = y_u(t) + \delta_u(t) \cdot \tau \cdot \sin(\phi_u(t)) \end{cases}, \forall u \in \mathcal{U}, t \in T \tag{1}$$

To prevent collisions, a minimum safety distance D^{safe} is enforced between UAVs, leading to the following collision avoidance constraints:

$$\|v_u(t) - v_{u'}(t)\| = \sqrt{(x_u(t) - x_{u'}(t))^2 + (y_u(t) - y_{u'}(t))^2} \geq D^{safe}, \forall t \in T, u \neq u' \tag{2}$$

Each UAV is also constrained to operate within a fixed rectangular area, so its position $v_u(t) = [x_u(t), y_u(t)]$ must satisfy the constraints in Eq. (3):

$$\begin{cases} 0 \leq x_u(t) \leq X^{max} \\ 0 \leq y_u(t) \leq Y^{max} \end{cases}, \forall u \in \mathcal{U}, t \in T \tag{3}$$

2.2 Task Queue Model

Limited computational resources at terminals and UAVs require implementing FIFO task queues to buffer tasks orderly and ensure sequential execution under high concurrency.

(1) UT queue model

Each user terminal m generates a set of indivisible computation-intensive tasks $\mathcal{A}_m(t)$ at time slot t, following a Poisson arrival process. A task queue $Q_m(t)$ is maintained with initial state $Q_m(0) = \varnothing$ to store pending tasks. The offloading decision is represented by $a_m^u(t) \in \{0,1\}$, where $a_m^u(t) = 1$ indicates the task is offloaded to UAV u, and otherwise it is processed locally. Each terminal operates in a single-threaded mode, processing one task at a time and connecting to at most one UAV.

Let $T_{m,loc}^{wait}(t)$ denote the waiting time of task $l_m(t)$ in the queue, representing the number of time slots the terminal must wait before processing begins, with $T_{m,loc}^{wait}(1) = 0$. The calculation is as follow:

$$T_{m,loc}^{wait}(t) = \left\{ \left\lceil \max_{t' \in \{1,2,\cdots,t-1\}} T_m^{loc}(t') - t + 1 \right\rceil \right\}^+ \tag{4}$$

where T_m^{loc} denote the completion time of the task being processed. The start time of a new task is determined by the maximum completion time among all its preceding tasks in the queue. The operator $\{e\}^+$ is an indicator function ensuring the waiting time remains non-negative, as detailed in Eq. (5):

$$\{e\}^+ = \begin{cases} e, & \text{if } e > 0 \\ 0, & \text{if } e \leq 0 \end{cases} \tag{5}$$

(2) UAV queue model

When a terminal's computational capacity is insufficient, tasks are offloaded to UAVs. Each UAV u maintains a separate queue $Q_{m,u}^{uav}$ for each associated terminal m, receiving offloading requests within its coverage at time slot t. The waiting time of task $l_m(t)$ in the UAV queue, $T_{m,u}^{wait}(t)$, counts the time slots before processing starts, with $T_{m,u}^{wait}(1) = 0$. It is calculated as follows:

$$T_{m,u}^{wait}(t) = \left\{ \left\lceil \max_{t' \in \{1,2,\cdots,t-1\}} T_{m,u}^{uav}(t') - t + 1 \right\rceil \right\}^+ \tag{6}$$

where $T_{m,u}^{uav}$ is the processing completion time of the task in the UAV u.

2.3 Communication Model

(1) G2A communication model

The Ground-to-Air (G2A) link may be blocked by buildings. Therefore, the channel model accounts for both Line-of-Sight (LoS) and Non-Line-of-Sight (NLoS) conditions to better reflect real-world propagation. The path loss between terminal and UAV is thus given by:

$$\mathrm{Loss}_{m,u}(t) = 20\log_{10}\left(\frac{4\pi f_c\sqrt{d_{m,u}^2 + H^2}}{c}\right) + p_{m,u}^{LoS}(t)\eta_{m,u}^{LoS}(t) + (1 - p_{m,u}^{LoS}(t))\eta_{m,u}^{NLoS}(t) \tag{7}$$

where $d_{m,u}$ is the horizontal distance between the terminal and the UAV, f_c is the carrier frequency, and c denotes the speed of light. $\eta_{m,u}^{LoS}$ and $\eta_{m,u}^{NLoS}$ represent the additional losses for LoS and NLoS links, respectively. The probability of a LoS link, $p_{m,u}^{LoS}$, is determined using an empirical statistical model, as follows:

$$p_{m,u}^{LoS} = \frac{1}{1 + a\exp\left(-b\left[\frac{180}{\pi}\arctan\left(\frac{H}{d_{mu}}\right) - a\right]\right)} \tag{8}$$

where the value of (a, b) is set according to the specific communication environment.

In summary, the data transmission rate between the terminal m and the UAV u can be expressed as:

$$r_{m,u}(t) = B_{m,u}\log_2\left(1 + \frac{P_m}{\sigma^2 10^{\mathrm{Loss}_{m,u}(t)/10}}\right) \tag{9}$$

where P_m is the transmission power of the terminal device and σ^2 is the noise power. $B_{m,u} = \frac{B_u}{N_u(t)}$ denotes the channel bandwidth allocated to the terminal m, where B_u represents the UAV bandwidth, and $N_u(t)$ denotes the number of terminals covered by the UAV u at the time of t.

(2) A2B communication model

The Air-to-Base Station (A2B) link may be obstructed by obstacles like high-rise buildings; thus, both LoS and NLoS conditions are considered to accurately model the channel. The path loss between UAV u and base station bs is expressed as:

$$\mathrm{Loss}_{u,bs}(t) = 20\log_{10}\left(\frac{4\pi f_c\sqrt{d_{u,bs}^2 + H^2}}{c}\right) + p_{u,bs}^{LoS}(t)\eta_{u,bs}^{LoS}(t) + (1 - p_{u,bs}^{LoS}(t))\eta_{u,bs}^{NLoS}(t) \tag{10}$$

The transmission rate is denoted as:

$$r_{u,bs}(t) = B_{u,bs}\log_2\left(1 + \frac{P_u}{\sigma^2 10^{\mathrm{Loss}_{u,bs}(t)/10}}\right) \tag{11}$$

where P_u is the transmission power of the UAV and σ^2 is the noise power. $B_{u,bs} = \frac{B_{bs}}{N_{bs}(t)}$ denotes the channel bandwidth allocated to the UAV u, B_{bs} represents the base station bandwidth, and $N_{bs}(t)$ denotes the number of UAVs served by the base station at the time of t.

2.4 Computing Model

(1) Local computing model

When terminal m processes a task locally, the total response delay consists of the local waiting time and processing time, as calculated by Eq. (12):

$$T_m^{loc}(t) = T_{m,loc}^{wait}(t) + T_{m,loc}^{proc}(t) \tag{12}$$

where $T_{m,loc}^{proc}(t)$ indicates the processing time of task $l_m(t)$, determined by the terminal's computing power and required computation, calculated as:

$$T_{m,loc}^{proc}(t) = \left\lceil \frac{d_m(t) \cdot \zeta_m(t)}{f_m \cdot \tau} \right\rceil \tag{13}$$

f_m is the computing resources of the user terminal.

Local task processing consumes energy dependent on the allocated CPU frequency, calculated as:

$$E_m^{loc}(t) = k_{loc} \cdot T_{m,loc}^{proc}(t) \cdot (f_m)^3 \tag{14}$$

where k_{loc} is the energy consumption factor of the terminal device.

(2) Offloading to the UAV computing model

When a terminal cannot process a task locally, it offloads it to its associated UAV. At time slot t, tthe set of tasks offloaded from terminal m to UAV u is denoted by $A_{m,u}(t)$. The total UAV processing latency includes transmission, queuing, and computation delays. Since UAVs fly at low altitudes near terminals, propagation delay is neglected. The response delay for UAV-offloaded tasks is calculated as follows:

$$T_{m,u}^{uav}(t) = T_{m,u}^{trans}(t) + T_{m,u}^{wait}(t) + T_{m,u}^{proc}(t) \tag{15}$$

where $T_{m,u}^{trans}(t) = \left\lceil \frac{d_m(t)}{r_{m,u}(t) \cdot \tau} \right\rceil$ denotes the number of time slots needed to transmit the task data from the terminal to the UAV. The UAV's task processing time is calculated as:

$$T_{m,u}^{proc}(t) = \left\lceil \frac{d_m(t) \cdot \zeta_m(t)}{f_{m,u}^{uav}(t) \cdot \tau} \right\rceil \tag{16}$$

$f_{m,u}^{uav}(t)$ denotes the computational resources allocated by the UAV u to task $l_m(t)$.

When a task is offloaded to the UAV, the system energy consumption comprises the user terminal's transmission energy $E_{m,u}^{trans}(t)$ and the UAV's computation energy $E_{m,u}^{uav}(t)$, which can be calculated as follows:

$$E_{m,u}^{\text{trans}}(t) = P_m \cdot \frac{d_m(t)}{r_{m,u}(t)} \tag{17}$$

$$E_{m,u}^{\text{proc}}(t) = k_{\text{uav}} \cdot T_{m,u}^{\text{proc}}(t) \cdot (f_{m,u}^{\text{uav}})^3 \tag{18}$$

where k_{uav} denotes the UAV energy consumption coefficient. Then the total system energy consumption in t time slot can be expressed as:

$$E_{m,u}^{uav}(t) = E_{m,u}^{trans}(t) + E_{m,u}^{proc}(t) \tag{19}$$

(3) Offloading to the base station

When a task' computational demand exceeds the UAV's capacity, it is offloaded to the ground base station. Assuming sufficient base station resources and immediate processing, the total delay includes UAV-to-base station transmission and base station processing times.

$$T^{bs}_{m,u,bs}(t) = T^{trans}_{m,u,bs}(t) + T^{proc}_{m,u,bs}(t) \tag{20}$$

where $T^{trans}_{m,u,bs}(t) = \left\lceil \frac{d_m(t)}{r_{u,bs}(t)\cdot\tau} \right\rceil$ represents the number of time slots required for transmitting task data from the UAV to the base station is represented accordingly. The processing time at the base station is calculated as follows:

$$T^{proc}_{m,u,bs}(t) = \left\lceil \frac{d_m(t)\cdot\zeta_m(t)}{f^{bs}_{m,u,bs}(t)\cdot\tau} \right\rceil \tag{21}$$

$f^{bs}_{m,u,bs}(t)$ represents the compute resources assigned by the base station to the task $l_m(t)$.

When a task is offloaded to the base station, the system energy consumption consists of the UAV's transmission energy $E^{trans}_{m,u,bs}(t)$ and the base station's computation energy $E^{bs}_{m,u,bs}(t)$, which are calculated as follows:

$$E^{\text{trans}}_{m,u,bs}(t) = P_u \cdot \frac{d_m(t)}{r_{u,bs}(t)} \tag{22}$$

$$E^{\text{proc}}_{m,u,bs}(t) = k_{\text{bs}} \cdot T^{\text{proc}}_{m,u,bs}(t) \cdot \left(f^{\text{bs}}_{m,u,bs}\right)^3 \tag{23}$$

where k_{bs} denotes the energy consumption coefficient of the base station. Thus, the total system energy consumption in time slot t can be expressed as:

$$E^{bs}_{m,u,bs}(t) = E^{trans}_{m,u,bs}(t) + E^{proc}_{m,u,bs}(t) \tag{24}$$

2.5 Problem Description

To minimize end-to-end task delay and overall energy consumption, this paper proposes a joint optimization framework integrating UAV trajectory planning, task offloading, and resource allocation. Based on the offloading decision $a^u_m(t)$, tasks are processed locally or offloaded to UAVs, which may further offload them to the base station. The total system delay $T_{total}(t)$ and energy consumption $E_{total}(t)$ at time slot t are formulated as follows:

$$T^{total}_m(t) = \sum_{m\in\mathcal{M}} \sum_{u\in\mathcal{U}} \left[(1 - a^u_m(t)) \cdot T^{loc}_m(t) + a^u_m(t) \cdot \min\left(T^{uav}_{m,u}(t), T^{bs}_{m,u,bs}(t) \right) \right] \tag{25}$$

$$E^{total}_m(t) = \sum_{m\in\mathcal{M}} \sum_{u\in\mathcal{U}} \left[(1 - a^u_m(t)) \cdot E^{loc}_m(t) + a^u_m(t) \cdot \min\left(E^{uav}_{m,u}(t), E^{bs}_{m,u,bs}(t) \right) \right] \tag{26}$$

The optimization objective minimizes a weighted sum of delay and energy consumption:

$$
\begin{aligned}
&P_1 : \min_{x,a,f} \sum_{t \in T} \left(W_T T_{total}(t) + W_E E_{total}(t) \right) \\
&\text{s.t.} \\
&C_1 : 0 \le \phi_u(t) \le 2\pi, \forall t \in T, u \in \mathcal{U} \\
&C_2 : 0 \le x_u(t) \le X^{max}, 0 \le y_u(t) \le \Upsilon^{max} \\
&C_3 : a_m^u(t) \in \{0,1\}, a_u^{bs}(t) \in \{0,1\}, \forall m \in \mathcal{M} \\
&C_4 : T_m^{total}(t) \le \zeta_m(t) \\
&C_5 : f_{m,u}^{uav}(t) \le f_u, \ f_{m,u,bs}^{bs}(t) \le f_{bs} \\
&C_6 : \| v_u(t) - v_{u'}(t) \| \ge D^{safe}, u \ne u'
\end{aligned}
\tag{27}
$$

where x denotes the UAV trajectory control decision, a represents the task offloading decision, and f is the resource allocation decision. C_1 and C_2 define the UAVs' flight direction and range; C_3 governs task offloading decisions at each time slot; C_4 ensures task execution time does not exceed its maximum tolerable delay; C_5 restricts allocated resources to not exceed the total available at the UAV and base station, and C_6 enforces a minimum safe distance between UAVs to prevent collisions.

P_1 is non-convex due to the coupling of hybrid discrete-continuous variables and nonlinear relationships among task offloading, resource allocation, and UAV trajectory, making it intractable for conventional optimization methods. To address this challenge, this paper introduces the PER-MATD3 algorithm, which leverages deep reinforcement learning to efficiently learn near-optimal solutions in dynamic environments.

3 Algorithm Design

3.1 Definition of MDP

The optimization problem (P1) is modeled as a Markov Decision Process (MDP) $\langle S, A, P, r, \gamma \rangle$ where S and A denote the system state and action spaces. P the state transition probabilities, r the immediate reward, and γ the discount factor.

(1) State space

User terminals observe local task information and the position of the UAV they are currently associated with, represented as $s_M(t) = \{v_u(t), Q_m(t), A_m(t)\}$, where $v_u(t)$ is the accessed UAV's position, $Q_m(t)$ the terminal's task queue, and $A_m(t)$ newly generated tasks. UAVs observe their own and other UAVs' positions, task queues, and new tasks: $s_{M+U}(t) = \{v_u(t), v_{u'}(t), Q_{m,u}^{uav}, A_{m,u}(t)\}$. The joint state space of the system in time slot can be denoted as $s(t) = s_M(t) \cup s_{M+U}(t)$.

(2) Action space

Terminals decide task offloading $a_M(t) = \{a_m^u(t)\}$. UAV actions include trajectory control, resource allocation, and offloading: $a_{M+U}(t) = \{\phi_u(t), \delta_u(t), a_u^{bs}(t), f_{m,u}^{uav}(t), f_{m,u,bs}^{bs}(t)\}$. Therefore, the joint action of the system in the time slot can be denoted as $a(t) = a_M(t) \cup a_{M+U}(t)$.

(3) Reward function

The reward guides agents to minimize total system cost under constraints. If all constraints are met, the reward equals the negative cost; otherwise, a penalty is applied. Formally:

$$
r(t) = \begin{cases} -\left(W_T T_{total}(t) + W_E E_{total}(t) \right), & \text{if all conditions are met} \\ -\eta_1 - \eta_2, & \text{otherwise} \end{cases}
\tag{28}
$$

When the flight trajectory of the UAV violates C_1~C_3, it will get the penalty η_1; if the task is discarded or exceeds the maximum tolerance time, it will get the penalty η_2.

3.2 PER-MATD3 Algorithm

In multi-agent UAV-assisted MEC networks, the high-dimensional continuous action space challenges traditional RL methods like Q-learning, DQN, and PG. To address this, we propose PER-MATD3, a prioritized experience replay extension of MATD3. Building on TD3's twin critics, target policy smoothing, and delayed updates to reduce overestimation and improve stability, PER-MATD3 prioritizes samples with high TD errors to accelerate convergence and enhance performance in complex multi-agent settings.

Leveraging centralized training with decentralized execution (CTDE), critics use global information to overcome partial observability, while actors act on local observations for scalability. Dual-delay updates and prioritized replay further stabilize training and improve adaptability in high-dimensional, continuous, and collaborative environments.

The PER-MATD3 framework, illustrated in Fig. 2. Each agent i, where $i \in \mathcal{M} \cup \mathcal{U}$, equips with an Actor network $\pi_i(s_i, a_i; \theta_i)$, $Q_i^1(s_i, a_i; \phi_i^1)$ and two Critic networks $Q_i^{1'}(s_i, a_i; \phi_i^{1'})$, $Q_i^{2'}(s_i, a_i; \phi_i^{2'})$, with corresponding target networks π_i', $Q_i^{1'}$, and $Q_i^{2'}$, where θ and ϕ are the parameters of the Actor and Critic networks, respectively. During centralized training, the Critic networks utilize global states $S(t)$ and actions $A(t)$ from all agents to compute rewards, while decentralized execution relies on local observations. The Actor network for each user terminal receives inputs $s_m(t) = \{v_m(t), Q_m(t), A_m(t)\}, \forall m \in M$, representing the accessed UAV location, local task queue, and new task information. For UAV agents, the input is $s_u(t) = \{v_u(t), v_{u'}(t), Q_u^{uav}, A_u(t)\}, \forall u \in U$, including its own and neighboring UAV locations, task queue, and offloaded task information. Each agent outputs an action $a_i(t)$ based on its observation. The Critic evaluates the action $a_i(t)$ using local observations, providing Q-values for policy updates.

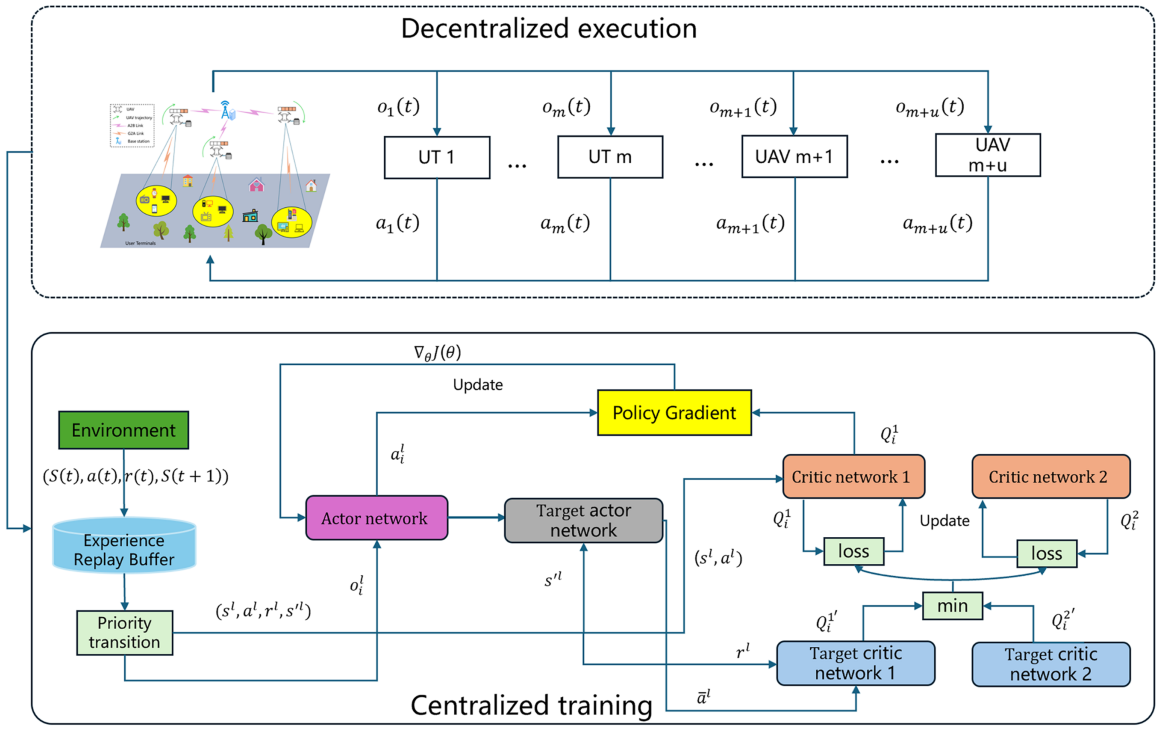


Figure 2: PER-MATD3 algorithm framework

Actor networks are optimized via policy gradient methods during centralized training, with the policy gradient computed as follows:

$$\nabla_\theta J(\theta) = \frac{1}{L}\sum_{l=1}^{L} \nabla_\theta \pi_i(s_i^l) \nabla_{a_i^l} Q_i^1(s^l, a^l)\Big|_{a_i^l=\pi_i(s_i^l))} \tag{29}$$

where L denotes the mini-batch size sampled via prioritized experience replay. Each agent's two Critic networks independently estimate Q-values, and the minimum is used to enhance stability and accuracy. Critic training follows the Temporal-Difference (TD) principle, leveraging the same prioritized replay mechanism as the Actor. The TD target for each sample is computed as:

$$y^l = \gamma \min_{k\in\{1,2\}} \{Q_i^{k'}(s_i^l, a^l), Q_i^{k'}(s_i^l, \bar{a}^l\} + r^l \tag{30}$$

where r^l denotes the immediate reward, γ is the discount factor balancing immediate and long-term rewards, and $Q_i^{k'}(s_i^{'l}, \bar{a}^l)$ the target Q-value, where the minimum of the two Critic outputs is used to reduce overestimation. Prioritized experience replay assigns sampling probabilities based on the TD error magnitude, favoring samples with larger errors to improve learning efficiency and convergence. Each experience is assigned a priority p^l related to its TD error, and importance sampling (IS) is incorporated. The priority p^l is defined as:

$$p^l = |y^l| + \varepsilon \tag{31}$$

where ε prevents zero priority. Sampling probability is given by:

$$P^l = \frac{(p^l)^\partial}{\sum_d^D (p^d)^\partial} \tag{32}$$

where D is the replay buffer size, $\partial \in [0,1]$ controlling sampling randomness and bias. The importance sampling (IS) weight is:

$$c^l = \left(\frac{1}{D}\cdot\frac{1}{P^l}\right)^{\eth} \tag{33}$$

where $\eth \in [0,1]$ adjusts IS correction strength. Finally, weights are normalized as:

$$\hat{c}^l = \frac{c^l}{\max_{l\in\{1,2,\ldots,L\}} c^l} \tag{34}$$

To mitigate value overestimation caused by sharp fluctuations in the Actor's output for state s_i^l, random noise is added to the action to produce a smoother and more stable target, calculated as:

$$\bar{a}^l = \pi_i'(s_i^{'l}) + \epsilon \tag{35}$$

where $\pi_i'(s_i^{'l})$ is the action output by the target Actor network in the next state. $\epsilon \sim clip(N(0,\delta^2), -1, 1)$ is the added smoothing noise obeying Gaussian distribution.

Critic network parameters are updated using Mean Squared Error (MSE) loss, and the loss function is defined as follows:

$$L(\phi_i^k) = \frac{1}{L}\sum_{l=1}^{L} \hat{c}_i \cdot \left[y^l - Q_i^k(s_i^l, a_i^l)\right]^2, k \in \{1,2\} \tag{36}$$

Therefore, the parameters of the Actor and Critic main network of each $agent_i$ are updated in the following way:

$$\begin{cases} \theta_i \leftarrow \theta_i + \lambda \nabla_\theta J(\theta_i) \\ \phi_i^k \leftarrow \phi_i^k - \lambda \nabla_{\phi_i^k} L(\phi_i^k), k \in \{1,2\} \end{cases} \tag{37}$$

where λ denotes the learning rate, controlling the step size of gradient updates to enhance training stability.

PER-MATD3 uses delayed policy updates: Critics update every step for rapid adaptation, while Actors update every d steps to stabilize policy changes from value bias. Target networks are softly updated to enhance convergence and robustness, as follows:

$$\begin{cases} \theta_i' \leftarrow \rho\theta_i + (1-\rho)\theta_i' \\ \phi_i^{k'} \leftarrow \rho\phi_i^k + (1-\rho)\phi_i^{k'}, k \in \{1,2\} \end{cases} \tag{38}$$

where ρ denotes the soft update factor of the target network. The algorithm pseudo-code of PER-MATD3 is shown in Algorithm 1.

Algorithm 1: The training process of PER-MATD3

Initialize actor/critic networks π_i and Q_i with target networks π_i' and Q_i'
Initialize prioritized replay buffer $\mathcal{B}$ with priority parameter α.
for each episode **do**
  Initialize joint state $s(0) = \{s_{ut}, s_{uav}\}$, set $t = 1$.
  while $t < T_p$ **do**
    UTs select actions $a_{ut}(t) = \pi_{ut}(s_{ut}(t)) + \xi$;
    UAVs select $a_{uav}(t) = \pi_{uav}(s_{uav}(t)) + \xi$;
    Execute joint action $\boldsymbol{a}(t)$, get reward $\mathcal{R}(t)$ and next state $s'(t)$.
    Compute TD error $\delta(t)$, store $(s(t), s'(t), \boldsymbol{a}(t), \mathcal{R}(t), \delta(t))$ in $\mathcal{B}$ with priority p.
    $s'(t) \leftarrow s(t)$.
    for agents $i \in \{UT, UAV\}$ **do**
      Sample mini-batch from $\mathcal{B}$ with probability $P \propto p$, compute IS weights w.
      Update critic networks by minimizing $L(\phi_i)$ with weighted MSE Eq. (36).
      if $t \mod d$ **then**
        Update actor networks via policy gradient Eq. (30).
        Soft-update target networks Eq. (38).
      end if
    end for
    $t \leftarrow t + 1$.
  end while
end for

The computational complexity of PER-MATD3 includes two phases: interaction with the environment and training with prioritized experience replay. The interaction phase with $M + N$ agents over T time slots costs $\mathcal{O}((M+N) \cdot T)$. Training involves prioritized sampling with complexity $\mathcal{O}(L \cdot \log|\mathcal{D}|)$ and actor-critic updates across all agents at cost $\mathcal{O}((M+N) \cdot L \cdot N^P)$, where N^P is the neural network size. Compared to standard MATD3, which uses uniform replay with $\mathcal{O}(L)$, PER-MATD3 introduces an additional

$\mathcal{O}(L \cdot \log|\mathcal{D}|)$ overhead due to priority management. Therefore, the total complexity is $\mathcal{O}((M+N) \cdot T + L \cdot \log|\mathcal{D}| + (M+N) \cdot L \cdot N^P)$.

4 Simulation Experiments and Performance Analysis

4.1 Experimental Setup

In order to verify the effectiveness of the algorithm proposed in this paper, we consider a UAV-assisted MEC scenario consisting of three UAVs and multiple ground user terminals and an edge base station. Among them, the UAVs fly in an area of 400 × 400 m^2 to provide computational offloading services for the user terminals in the coverage area, and each user terminal randomly generates delay-sensitive computational tasks at the beginning of each time slot. Each task is associated with a maximum tolerable delay. If a task cannot be scheduled for execution before this deadline due to resource constraints, it is deemed infeasible and discarded. The experimental parameter settings refer to [9,18], and the main experimental parameters are shown in Table 1.

Table 1: Experimental parameter settings

Parameters description	Symbol	Values
Data size of task	$d_m(t)$	[5, 20] MB
Computational resource of task	$c_m(t)$	[100, 500] cycles/bit
Noise power	σ^2	−100 dBm
Transmit power of the terminal and the UAV	P_m, P_u	100 mW, 5 W
UAV flight altitude	H	80 m
UAV safety distance	D^{safe}	15 m
Bandwidth of UAV and Base Station	B_{uav}, B_{bs}	10, 50 MHz
Energy consumption coefficien of UT and UAV	k_{loc}, k_{uav}	$1 \times 10^{-26}, 1 \times 10^{-27}$
Computing capability	f_m, f_u, f_{bs}	0.75, [1, 4], 100 GHz
Weight parameter	W_T, W_E	0.7, 0.3
Soft update coefficient	ρ	0.01
Learning rate	γ	0.96

In this paper, three classical reinforcement learning algorithms—MATD3 [19], MADDPG [20], and PPO [21]—are used as baseline benchmarks. MATD3 extends TD3 to multi-agent settings, improving stability and reducing overestimation in continuous action spaces, making it suitable for coordinated UAV task offloading. MADDPG enables centralized training with decentralized execution, allowing multiple UAVs to learn cooperative strategies. PPO is a single-agent policy gradient method that provides stable and efficient updates, serving as a baseline for independent UAV scenarios. We compare these algorithms under the same simulation settings in terms of convergence speed, training stability, and average system task latency and energy consumption, providing a comprehensive evaluation of the proposed method.

4.2 Result Analysis

Fig. 3 presents the training convergence curves of the compared algorithms in terms of average episode reward. PER-MATD3 achieves rapid improvement in the early training phase and stabilizes after approximately 1000 episodes, demonstrating fast convergence and high policy stability. In contrast, MATD3 converges more slowly, while MADDPG and PPO continue to exhibit significant oscillations beyond 1000 episodes, indicating poorer learning stability. This improved convergence is attributed to the integration

of prioritized experience replay and the dual-Q network architecture, which together enhance learning efficiency and mitigate value overestimation.

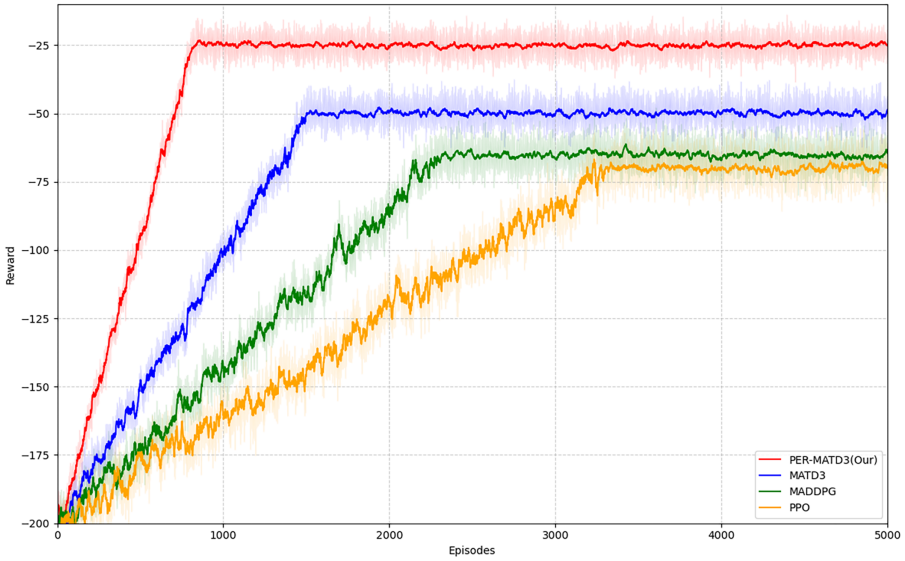
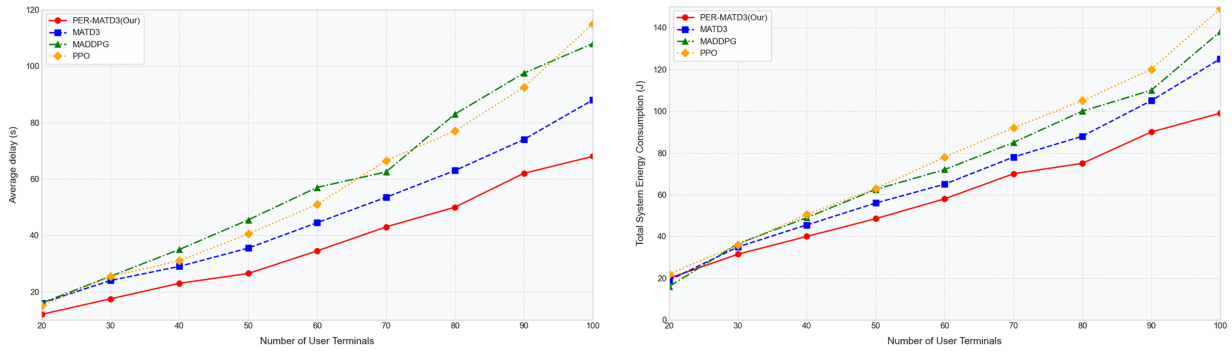


Figure 3: Algorithm convergence performance comparison

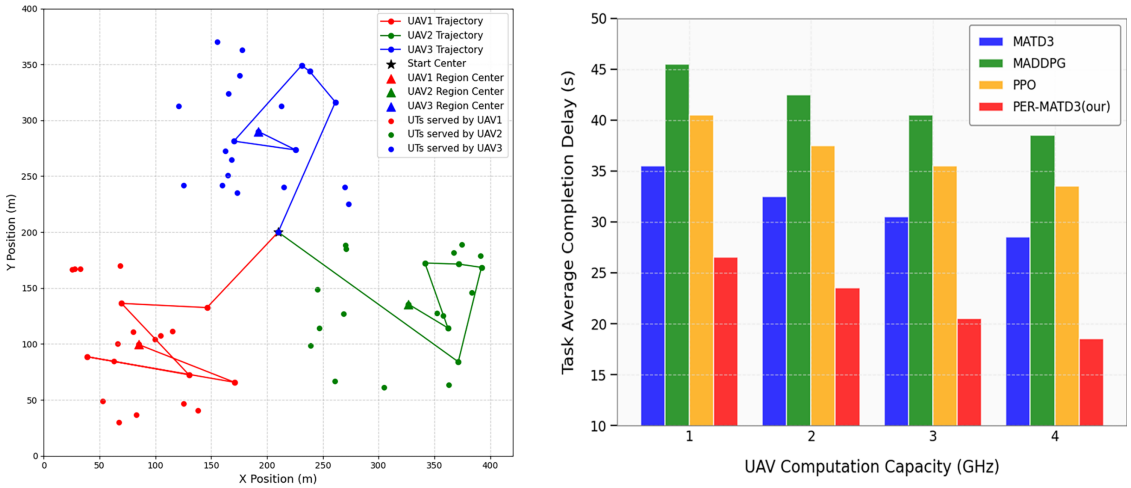
Fig. 4 compares the performance of different algorithms under varying numbers of user terminals. Fig. 4a depicts the average task completion delay of each algorithm as the number of user terminals increases from 20 to 100. With more terminals generating delay-sensitive tasks, task completion delay rises for all methods due to limited computational and communication resources. Nevertheless, the proposed PER-MATD3 consistently achieves the lowest delay, demonstrating superior decision efficiency. Specifically, when the number of terminals reaches 100, PER-MATD3 reduces the average delay by 22.7% compared to MATD3. Notably, beyond 60 terminals, PER-MATD3 maintains low and stable latency, while other algorithms exhibit significant delay spikes, indicating better robustness and scalability under heavy load. Fig. 4b shows the system's total energy consumption with increasing user terminals. As the number of user terminals increases from 20 to 100, the system's total energy consumption gradually rises due to higher task loads and intensified resource contention. It can be observed from the results that PER-MATD3 consistently achieves the lowest energy consumption among all methods, demonstrating its superior resource scheduling and energy efficiency. Specifically, at 100 terminals, PER-MATD3 reduces energy consumption by 20.8% compared to MATD3. This improvement is attributed to the enhanced learning efficiency of prioritized experience replay, which assigns higher sampling priority to more informative transitions, thereby accelerating convergence to energy-efficient offloading and UAV trajectory policies.



(a) average task completation delay **(b) system energy consumption**

Figure 4: Comparison of algorithm performance with different terminal numbers

Fig. 5a depicts the deployment and trajectory planning of three UAVs in a planar coordinate system. All UAVs are launched from the center of the experimental area, and they fly in different directions to avoid inter-UAV collisions. Each UAV determines its destination by optimizing its trajectory to cover the maximum number of user terminals (UTs) within its communication range. Consequently, the UAVs converge to the centroids of three high-density UT clusters and hover there to provide stable computation offloading services. This strategy achieves efficient spatial coverage, reduces transmission delay, and improves overall system performance. Fig. 5b compares the average task delay under varying UAV computing capacities with 50 UTs. As the UAV computation capacity increases from 1 to 4 GHz, tasks are processed more faster, leading to shorter queuing delays and less offloading to the base station, which significantly reduces overall latency. PER-MATD3 consistently achieves the lowest latency. Experimental results show that PER-MATD3 achieves the best performance overall.



(a) UAV Trajectory Control **(b) Comparison of average task latency**

Figure 5: UAV trajectory control and task latency under varying computing power

5 Conclusion

This paper addresses the joint optimization of task offloading, resource allocation, and UAV trajectory control in multi-UAV-assisted MEC networks by proposing PER-MATD3. Leveraging a centralized training and distributed execution framework, it accelerates policy convergence and improves sample efficiency

through prioritized replay. Simulations demonstrate that PER-MATD3 outperforms existing methods in task delay and energy consumption, confirming its robustness in dynamic environments. Future work will incorporate inter-task dependencies and user mobility to enhance scalability and real-world applicability.

Acknowledgement: The authors sincerely thank all those who supported and contributed to this research.

Funding Statement: This work was supported by the National Natural Science Foundation of China under Grant No. 61701100.

Author Contributions: Sai Xu: conceptualization, methodology and writing; Jun Liu: supervision, project administration and funding acquisition; Shengyu Huang: software, validation, visualizatio; Zhi Li: conceptualization, methodology and formal analysis. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The raw data supporting the conclusions of this article will be made available by the authors on request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Li S, Liu G, Li L, Zhang Z, Fei W, Xiang H. A review on air-ground coordination in mobile edge computing: key technologies, applications and future directions. Tsinghua Sci Technol. 2024;30(3):1359–86. doi:10.26599/tst.2024.9010142.
2. Xu X, Han M, Xie N, Li G. Joint resource allocation methodology for space-air-ground collaboration. In: 2025 5th International Conference on Neural Networks, Information and Communication Engineering (NNICE); 2025 Jan 10–12; Guangzhou, China. p. 849–53.
3. Liu C, Zhong Y, Wu R, Ren S, Du S, Guo B. Deep reinforcement learning based 3D-trajectory design and task offloading in UAV-enabled MEC system. IEEE Trans Vehicular Technol. 2025;74(2):3185–95. doi:10.1109/tvt.2024.3469977.
4. Zhao M, Zhang R, He Z, Li K. Joint optimization of trajectory, offloading, caching, and migration for UAV-assisted MEC. IEEE Trans Mob Comput. 2025;24(3):1981–98. doi:10.1109/tmc.2024.3486995.
5. Pervez F, Sultana A, Yang C, Zhao L. Energy and latency efficient joint communication and computation optimization in a multi-UAV-assisted MEC network. IEEE Trans Wirel Commun. 2024;23(3):1728–41. doi:10.1109/twc.2023.3291692.
6. Hui M, Chen J, Yang L, Lv L, Jiang H, Al-Dhahir N. UAV-Assisted mobile edge computing: optimal design of UAV altitude and task offloading. IEEE Trans Wirel Commun. 2024;23(10):13633–47. doi:10.1109/twc.2024.3403536.
7. Ma L, Li N, Guo Y, Wang X, Yang S, Huang M, et al. Learning to optimize: reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system. IEEE Trans Cybern. 2022;52(12):12698–711. doi:10.1109/tcyb.2021.3086501.
8. Tang J, Nie J, Zhang Y, Duan Y, Xiong Z, Niyato D. Air-ground collaborative edge intelligence for future generation networks. IEEE Netw. 2023;37(2):118–25. doi:10.1109/mnet.008.2200287.
9. Zhao N, Ye Z, Pei Y, Liang Y-C, Niyato D. Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing. IEEE Trans Wirel Commun. 2022;21(9):6949–60. doi:10.1109/twc.2022.3153316.
10. Li F, Gu C, Liu DS, Wu YX, Wang HX. DRL-based joint task scheduling and trajectory planning method for UAV-assisted MEC scenarios. IEEE Access. 2024;12:156224–34. doi:10.1109/access.2024.3479312.
11. Li H, Qu L, Chen W, Shao D. Fairness-aware joint optimization of 3D trajectory and task offloading in multi-UAV edge computing systems. In: 2025 21st International Conference on the Design of Reliable Communication Networks (DRCN); 2025 May 12–15; Ningbo, China. p. 1–5.

12. Ma L, Qian Y, Yu G, Li Z, Wang L, Li Q, et al. TBCIM: two-level blockchain-aided edge resource allocation mechanism for federated learning service market. IEEE/ACM Transact Netw. 2025. doi:10.1109/TON.2025.3589017.
13. Seid AM, Boateng GO, Mareri B, Sun G, Jiang W. Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network. IEEE Trans Netw Serv Manag. 2021;18(4):4531–47. doi:10.1109/tnsm.2021.3096673.
14. Yin J, Tang Z, Lou J, Guo J, Cai H, Wu X, et al. QoS-aware energy-efficient multi-UAV offloading ratio and trajectory control algorithm in mobile-edge computing. IEEE Internet Things J. 2024;11(24):40588–602. doi:10.1109/jiot.2024.3452111.
15. Gao Z, Yang L, Dai Y. MO-AVC: deep-reinforcement-learning-based trajectory control and task offloading in multi-UAV-enabled MEC systems. IEEE Internet Things J. 2023;11(7):11395–414. doi:10.1109/jiot.2023.3329869.
16. Wu G, Liu Z, Fan M, Wu K. Joint task offloading and resource allocation in multi-UAV multi-server systems: an attention-based deep reinforcement learning approach. IEEE Trans Vehicular Technol. 2024;73(8):11964–78. doi:10.1109/tvt.2024.3377647.
17. Zhang Q, Gao A, Wang Y, Zhang S, Ng SX. Multiple dual-function UAVs cooperative computation offloading in hybrid mobile edge computing systems. In: ICC 2024-IEEE International Conference on Communications; 2024 Jun 9–13; Denver, CO, USA. p. 1–6.
18. Shao Z, Yang H, Xiao L, Su W, Chen Y, Xiong Z. Deep reinforcement learning-based resource management for UAV-assisted mobile edge computing against jamming. IEEE Trans Mob Comput. 2024;23(12):13358–74. doi:10.1109/tmc.2024.3432491.
19. Ackermann J, Gabler V, Osa T, Sugiyama M. Reducing overestimation bias in multi-agent domains using double centralized critics. arXiv:1910.01465. 2019.
20. Du J, Kong Z, Sun A, Kang J, Niyato D, Chu X, et al. MADDPG-based joint service placement and task offloading in MEC empowered air-ground integrated networks. IEEE Internet Things J. 2023;11(6):10600–15. doi:10.1109/jiot.2023.3326820.
21. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv:1707.06347. 2017.