Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077824

ARTICLE

A Secure Task Offloading Scheme for UAV-Assisted MEC with Dynamic User Clustering and Cooperative Jamming: A Method Combining K-Means and SAC (K-SAC)

Jiajia Liu[1,2], Shuchen Pang[3], Peng Xie[3], Haitao Zhou[3], Chenxi Du[3], Haoran Hu[3], Bo Tang[3], Jianhua Liu[3], Fei Jia[1] and Huibing Zhang[1,*]

[1]School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, China
[2]Faculty Development and Teaching Evaluation Center, Civil Aviation Flight University of China, Guanghan, China
[3]College of Aviation Electronic and Electrical Engineering, Civil Aviation Flight University of China, Guanghan, China

*Corresponding Author: Huibing Zhang. Email: Zhanghuibing@guet.edu.cn

Received: 17 December 2025; Accepted: 03 February 2026; Published: 09 April 2026

ABSTRACT: In the unmanned aerial vehicle (UAV) assisted edge computing system, the broadcast characteristics of the UAV signal, the high mobility of the UAV, and the limited airborne energy make the task offloading strategy face challenges such as increased risk of information disclosure, limited computing resources, and the trade-off between energy consumption and flight time. To address these issues, we propose a K-means in-depth reinforcement learning algorithm based on Soft Actor-Critic (SAC). The proposed method first leverages the K-means clustering algorithm to determine the optimal deployment of ground jammers based on the final distribution of mobile users. Then, building upon the SAC framework, the Cross-Entropy Method (CEM) global sampling strategy is incorporated into the action output phase to form the K-SAC algorithm. This algorithm aims to maximize system rewards, which holistically balance task offloading delay, energy consumption, and secure offloading rate. Consequently, it jointly optimizes the optimal hovering positions of auxiliary UAVs and the task offloading ratio for each user, leading to an overall performance improvement in system security and efficiency. Finally, compared with current schemes, the system benefits achieved by the proposed scheme were 9.83% higher on average in different computing task sizes, 13.67% higher on average in different task complexities, and 14.63% higher on average in different interference powers.

KEYWORDS: Mobile edge computing (MEC); SAC; communication security; K-means; UAV

1 Introduction

The main difference between UAV edge computing [1,2] and ground edge computing is that ground edge computing [3,4] focuses on deploying computing resources on ground devices at the edge of the network to provide users with computing resources. UAV edge computing focuses on integrating computing power into UAV systems. Drone-based computing servers offer greater flexibility than ground deployment [5].

Compared with traditional ground equipment, due to the high altitude of drones, their air to ground or air-to-air communication links are more likely to be line of sight communication links [6]. On the one hand, compared to complex signal propagation methods such as diffraction and reflection, Line of Sight (LoS) transmission has lower channel attenuation, better signal quality, and lower latency [7]. On the other hand, because signals are not easily obstructed by other obstacles, UAVs can act as airborne base stations in line of sight transmission [8], directly communicating with ground users without the need for

additional relay equipment, making deployment more flexible. Although line of sight transmission has many advantages, it also faces some challenges. Due to the broadcasting characteristics of wireless communication between drones and the ground, drones are easily vulnerable to malicious attacks or eavesdropping [9], leading to communication security issues such as data leakage. To solve this problem, traditional security mechanisms focus on encryption, authentication, and key management [10]. For example, Xiao et al. [11], Ghribi et al. [12] proposed communications schemes combined blockchain and encryption techniques to building mechanism to ensure the security of drone network communication. Gai et al. [13] and Usman et al. [14] proposed authentication mechanisms to guarantee the security of unmanned aerial vehicle communication. However, The above methods for upper-layer encryption and secure communication do bolster drone communication security, but at the cost of substantial consumption of the drone's onboard computing and communication resources. With a growing number of communication nodes, securing sufficient resources to handle incoming data offloaded from other users becomes increasingly challenging.

Physical layer secure transmission, which exploits inherent channel properties like fading and noise without relying on complex encryption, offers an efficient solution to enhance security in dynamic UAV networks with limited resources [15,16]. Consequently, it has attracted growing research interest. For instance, Zhou et al. [17] and Lu et al. [18] designed secure communication schemes for UAV mobile edge computing using algorithms such as block coordinate descent and successive convex approximation. However, they did not fully account for the impact of UAV flight energy consumption and environmental obstacles. Similarly, works in [19–21] employed convex optimization and resource allocation techniques to jointly optimize trajectories, power, and task offloading, yet they also overlooked obstacle constraints. To address the vulnerabilities of Line-of-Sight links, researchers in [22,23] proposed iterative optimization and distributed design algorithms, but these often neglected flight energy costs. To address this issue, [24] applied deep reinforcement learning for trajectory and resource optimization, while [25] developed a two-tier robust algorithm to secure transmissions and reduce energy consumption. Nevertheless, these schemes fail to integrate obstacle considerations. Furthermore, Lei et al. [26] leveraged deep reinforcement learning in a NOMA-assisted UAV-MEC system to handle uncertain eavesdroppers and reduce latency and energy consumption. However, this approach relies on an idealized environment without incorporating practical constraints.

Although there has been considerable research on secure offloading in UAV-assisted MEC systems [4,20], effectively balancing security, latency, and energy efficiency in highly dynamic environments remains a challenging issue. These challenges provide research space for further joint optimization. For convenience, we summarize the differences between existing studies and our work in Table 1.

First, most existing risk-aware schemes typically focus on passive risk avoidance—such as selecting communication channels with higher signal-to-noise ratios or relying on fixed ground infrastructure—rather than employing active defense mechanisms. They often overlook the potential of cooperative jamming between auxiliary UAVs and ground jammers, failing to fully utilize interference to degrade the eavesdropper's channel quality. Second, unlike traditional multi-server MEC environments where servers are static, UAV-assisted systems introduce high mobility. Existing methods often fail to address the dynamic risk variations caused by random user movement and UAV flight, leading to suboptimal offloading decisions when the network topology changes rapidly. Third, the tripartite coupling relationship among secure offloading rate, system delay, and complex energy consumption (including UAV hovering, flight, and jamming) has not been thoroughly analyzed in current multi-server risk models. The existing models often aggregate these energy costs rather than parsing their individual impact on security, compromising the fidelity of optimization results. To address these issues, we propose a K-means-enhanced reinforcement learning algorithm based on the SAC framework. Our method first deploys ground jammers optimally by

applying K-means clustering to the final distribution of mobile users. Then, within the SAC framework, we integrate the CEM as a global sampling strategy during the action output phase, forming the so-called K-SAC algorithm.

Table 1: Comparison of the proposed scheme with related works discussed in the third paragraph of the introduction.

Ref.	Mobility Model	Security Strategy	Energy Modeling	Method	Limitation/Gap
[15,16]	Multi-UAV	General PLS	N/A	Survey/Theory	Theoretical foundation; lacks specific joint algorithms.
[17]	Trajectory	Passive (Trajectory)	Flight & Comm.	Convex Opt. (BCD)	High computational complexity; lacks active jamming.
[18]	Trajectory	Secure NOMA	Comm. only	Convex Opt. (SCA)	Computationally expensive; no ground cooperation.
[19–21]	Trajectory	Passive / Power	Flight only	Convex Opt.	Ignores obstacle constraints & jamming energy costs.
[22,23]	Trajectory	Distributed	Comm. only	Iterative Opt.	Neglects flight energy; limited adaptability to dynamics.
[24]	Trajectory	Passive	Flight & Comm.	DRL (Deep RL)	Single-agent limitations; lacks cooperative jamming.
[25]	Trajectory	Robust Jamming	Flight & Comm.	Robust Opt.	Fails to integrate obstacle constraints.
[26]	Trajectory	NOMA-assisted	Flight & Comm.	DRL (Deep RL)	Relies on idealized environment assumptions.
Ours	**User & UAV**	**Active Coop. Jamming**	**Full (Hover+Flight)**	**K-SAC**	–

The main contributions of this paper are as follows:

(1) We construct a novel multi-UAV MEC secure communication framework that integrates ground jammers for active cooperative jamming. Unlike existing models that often treat energy consumption components in isolation, our model establishes a rigorous tripartite coupling relationship among secure offloading rate, system delay, and comprehensive energy costs (including local computation, jamming, edge execution, and UAV hovering/flight). This holistic modeling enables more precise energy-efficiency optimization compared to traditional partial energy models.

(2) To address the challenge of dynamic topology changes caused by random user movement, we propose the K-SAC algorithm, a novel hybrid deep reinforcement learning method. This approach uniquely combines K-means clustering for optimal static jammer deployment with a SAC framework enhanced by CEM global sampling. This hybrid design effectively overcomes the local optima and convergence issues found in traditional DRL algorithms when handling high-dimensional continuous action spaces.

(3) Simulation results confirm the superior performance of the proposed scheme. Compared to existing baselines, it achieves significant and consistent improvements in system utility, with average gains of 9.83% across varying computational task sizes, 13.67% across different task complexities, and 14.63% under diverse interference power levels, demonstrating its comprehensive effectiveness and robustness.

2 Problem Statement

2.1 System Model

The system model, as illustrated in Fig. 1, examines a secure multi-drone MEC communication network comprising neighboring base stations, auxiliary and intrusion drones, ground jammers, and multiple user devices. Servers outfitted on the adjacent base stations offer computational resources to all users within the system. Intrusion drones act as mobile eavesdroppers, attempting to intercept data during the auxiliary drones' operation. To counteract this threat, the auxiliary UAV employs a full-duplex mode with a two-antenna setup: one antenna is dedicated to receiving the user's offloaded signal, while the second simultaneously transmits interference signals toward the invading UAV. Both the user and the invading UAV are equipped with a single antenna for their respective transmission and eavesdropping tasks [22].

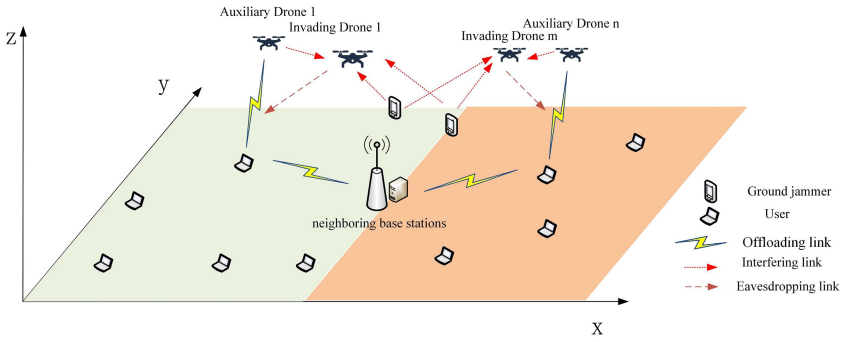

Figure 1: System model diagram.

To further improve security, ground jammers can send interference signals to interfere with invading drones. In addition, in this paper, we consider a multiple access channel where all users can use the same channel to transmit their signals. And assuming the user's location is fixed, the auxiliary UAV can obtain the location information of eavesdropping nodes and user nodes through devices such as cameras or synthetic aperture radar (SAR) [23,27] installed on the auxiliary UAV.

2.2 Communication Model

Following the system model described, we define the collections of entities involved. The system comprises a total of ς auxiliary UAVs, denoted by the set $M_1 = \{1, 2, 3, \ldots, \varsigma\}$, and ξ invading UAVs, represented by $M_2 = \{1, 2, 3, \ldots, \xi\}$. On the ground, there are ζ user devices, which form the set $N_1 = \{1, 2, 3, \ldots, \zeta\}$, and ι ground jammers, belonging to the set $N_2 = \{1, 2, 3, \ldots, \iota\}$. The main symbols used in this paper and their physical interpretations are summarized in Table 2.

Table 2: Important symbols and their significance.

Symbols	Significance
d_{ij}	Distance between user i and auxiliary UAV j.
d_{iu}	Distance between user i and the invading UAV u.
d_{ju}	Distance between the intrusion UAV u and the auxiliary UAV j.
d_{lu}	Distance between the intrusion UAV u and the ground jammer l.
r_{ij}	The uplink transmission rate from user i to auxiliary UAV j.
r_{iu}	The uplink transmission rate from user i to the invading UAV u.

(Continued)

Table 2 (continued)

Symbols	Significance
T_i^{loc}	Local computing latency of user device i.
T_{ij}^{uav}	Transmission delay from user device i to auxiliary UAV j.
T_i^{jzul}	Transmission delay from user device i to base station.
E_{li}^{dm}	Interference energy consumption generated by ground jammer l during user equipment i unloading.
E_{ji}^{jam}	Interference energy consumption caused by electromagnetic waves emitted by auxiliary UAV j on user device i.
E_i^{ul}	Energy consumption of communication transmission from users i to base stations.
E_{ji}^{uav}	Energy consumption of communication transmission from user i to auxiliary UAV j.

In this paper, the environment is modeled as dynamic: user equipment undergo random motion with velocity v_0, and each ground jammer moves with velocity v_1. The planar coordinates of user equipment i and ground jammer l are given by $n_i = (x_i, y_i)$ (for $i \in N_1$) and $w_l = (x_l, y_l)$ (for $l \in N_2$), respectively. For the aerial units, their positions are described in three dimensions. The horizontal position of an auxiliary drone j is $m_j = (x_j, y_j)$ with a vertical altitude of z_j (for $j \in M_1$), while an invading drone u has a horizontal position of $q_u = (x_u, y_u)$ and a vertical altitude of z_u (for $u \in M_2$). The mobility of users and jammers is captured across time slots, while their positions are fixed within each time slot during task processing and transmission. Consequently, the distance between user device i and auxiliary drone j is determined as follows:

$$d_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_j\right)^2}. \tag{1}$$

The distance d_{iu} between user device i and the invading drone u is defined as follows:

$$d_{iu} = \sqrt{\left(x_i - x_u\right)^2 + \left(y_i - y_u\right)^2 + \left(z_u\right)^2}. \tag{2}$$

The distance d_{lu} between the ground jammer l and the invading drone u is expressed as follows:

$$d_{lu} = \sqrt{\left(x_l - x_u\right)^2 + \left(y_l - y_u\right)^2 + \left(z_u\right)^2} \tag{3}$$

The distance d_{ju} between the auxiliary drone j and the invading drone u is given as follows:

$$d_{ju} = \sqrt{\left(x_j - x_u\right)^2 + \left(y_j - y_u\right)^2 + \left(z_j - z_u\right)^2} \tag{4}$$

For the analysis of the air-to-ground link, the connection between the UAV and the user is depicted using a Line of Sight (LoS) model [24]. The gain of the channel is therefore determined via the free space loss

model. It is also assumed that the Doppler shift resulting from movement is fully compensated at the receiver. Considering these conditions, the uplink data rate from user 'i' toward auxiliary UAV 'j' is formulated as:

$$r_{ij} = \log_2\left(1 + \frac{p_i \beta_1}{\left(\chi p_{jam} + \sigma^2\right) {d_{ij}}^2}\right) \tag{5}$$

where the index i ($i \in N_1$) denotes the user device, while index j ($j \in M_1$) represents the auxiliary UAV. The term β_1 signifies the power gain of the channel over a standard distance of $d_0 = 1$ m, determined by the equation $\beta_1 = g_t g_r (\frac{\lambda}{4\pi d_0})^2$. Here, g_t and g_r are the respective gains for the user's transmitting antenna and the UAV's receiving antenna, and λ corresponds to the signal's wavelength. Additionally, p_i indicates the transmission power for user i, while σ^2 represents the additive noise power. For anti-eavesdropping purposes, the auxiliary UAV transmits a jamming signal with a power level of p_{jam}. However, this action leads to self-interference at the UAV's receiver, a phenomenon quantified by the coefficient χ. Consequently, the power of this self-interference noise is given by the product $\chi \times p_{jam}$.

The interference noise σ_{ju} of auxiliary drone j on invading drone u is given as follows:

$$\sigma_{ju} = \frac{\beta_2 p_{\text{jam}}}{{d_{ju}}^2} \tag{6}$$

where $j \in M_1$, $u \in M_2$, $\beta_2 = g_r g_r \left(\frac{\lambda}{4\pi d_0}\right)^2$, and β_2 represents the channel gain between the auxiliary drone and the invading drone at a reference distance of $d_0 = 1$ m. In addition to the noise impact of the interference signal sent by the auxiliary UAV on the invading UAV, since the auxiliary UAV already knows the interference signal sent by the ground jammer [28], it can subtract the interference signal from the received signal. However, the invading UAV is unaware of the presence of ground jammers and considers all received signals as useful signals. Therefore, the interference signals sent by ground jammers also have a noise impact on invading UAVs.

The interference noise σ_{lu} of ground jammers l on invading drones u is expressed as follows:

$$\sigma_{lu} = \frac{\beta_3 p_{\text{dm}}}{{d_{lu}}^2} \tag{7}$$

where $l \in N_2$, $u \in M_2$, and the channel gain is given by $\beta_3 = g_k g_r \left(\frac{\lambda}{4\pi d_0}\right)^2$. β_3 signifies the channel gain between them over a standard reference distance of $d_0 = 1$ m. The power with which the ground jammer transmits these interfering electromagnetic waves is denoted by p_{dm}.

The rate of uplink transmission from user device i to the invading UAV u is given by:

$$r_{iu} = \log_2\left(1 + \frac{p_i \beta_1}{\left(\sum_{l=1}^{N_2} \sigma_{lu} + \sigma_{ju} + \sigma^2\right) {d_{iu}}^2}\right) \tag{8}$$

where $i \in N_1$, $u \in M_2$, $l \in N_2$. We make the simplifying assumption in this paper that g_t denotes the channel gain between user i and the base station, a value that is taken to be fixed for the duration of a time slot. For a given time slot t, the rate of uplink data transfer from user i to the base station is given by:

$$r_i = \text{lb}\left(1 + \frac{p_i g_i}{\sigma^2}\right) \tag{9}$$

For any user $i \in N_1$, this paper operates under the assumption that both local computation and task offloading to nearby base stations are secure channels free from information theft. Consequently, the primary security vulnerability is isolated to the offloading process from a user device to an auxiliary drone. Based on this premise, the secure offloading rate [25,26] from user i to auxiliary drone j is defined as follows:

$$R_{ij}^{\text{sec}} = \left[r_{ij} - r_{iu}\right]^{+} \tag{10}$$

where $i \in N_1$, $j \in M_1$, $u \in M_2$.

2.3 Time Delay Model

Within the MEC framework, each user device i is required to handle computationally intensive tasks, denoted by $W_i = \left(D_i, C_i, T_i^0\right)$, representing its data size, required CPU cycles, and maximum tolerable delay, respectively. For modeling purposes, we simplify this by assuming a uniform maximum delay T for all tasks, thus $T_i^0 = T$. The total delay experienced by user device i is a composite of five distinct components: the local computation latency, the transmission latency to the auxiliary drone, the drone's computation latency, the transmission latency to a neighboring base station, and the base station's computation latency. The distribution of these tasks is governed by the offloading ratios $\varepsilon_i^{\text{loc}}$, $\varepsilon_{ij}^{\text{uav}}$, and $\varepsilon_i^{\text{ul}}$, where $\varepsilon_i^{\text{loc}} \times D_i$ is the portion of the task computed locally, $\varepsilon_{ij}^{\text{uav}} \times D_i$ is the fraction offloaded to auxiliary UAV j, and $\varepsilon_i^{\text{ul}} \times D_i$ corresponds to the share sent to a neighboring base station.

Accordingly, the local computing latency for user device i is formulated as follows:

$$T_i^{\text{loc}} = \frac{\varepsilon_i^{\text{loc}} \times D_i \times C_i}{f_0} \tag{11}$$

Among them, $i \in N_1$. We assume that all users share the same local computation frequency, denoted as f_0. Furthermore, we assume that each user is allocated an identical transmission bandwidth of B. It is reasonable to assume that ground user equipment of the same product model possesses consistent computational frequencies and transmission bandwidths. Based on these assumptions, the transmission delay for offloading a task from user equipment i to auxiliary UAV j is given by:

$$T_{ij}^{\text{uav}} = \frac{\varepsilon_{ij}^{\text{uav}} \times D_i}{B \times r_{ij}} \tag{12}$$

The time delay caused by assisting UAV j to perform user i offloading task can be expressed as:

$$T_{ij}^{c} = \frac{\varepsilon_{ij}^{\text{uav}} \times D_i \times C_i}{f_{ij}} \tag{13}$$

The computational resources of an auxiliary UAV are distributed among users proportionally to the volume of data each user offloads. Consequently, we define f_{ij} as the specific computational frequency that auxiliary UAV j assigns to user $i \in N_1$ under this allocation scheme. When the task data size of each user device is the same, it meets the following requirements:

$$\sum_{i=1}^{N_1} \varepsilon_{ij}^{\text{uav}} \times f_{ij} = f_{\max}^{UAV} \times \varepsilon_{ij}^{\text{uav}} \tag{14}$$

The transmission delay T_i^{jzul} of user i offloading tasks to the base station can be expressed as follows:

$$T_i^{jzul} = \frac{\varepsilon_i^{\mathrm{ul}} \times D_i}{B \times r_i} \tag{15}$$

The delay T_i^{jzc} caused by the base station executing the user i offloading task can be expressed as follows:

$$T_i^{jzc} = \frac{\varepsilon_i^{\mathrm{ul}} \times D_i \times C_i}{f_i^{jz}} \tag{16}$$

where $i \in N_1$. f_i^{jz} represents the calculation frequency allocated to i by neighboring base stations.

Since the allocation of base station computing resources is directly proportional to the size of task data offloaded by users to the base station, when the task data size of each user device is the same, it satisfies:

$$f_i^{jz} = \frac{f_{jz}\varepsilon_i^{\mathrm{ul}}}{\sum_{i=1}^{N_1} \varepsilon_i^{\mathrm{ul}}} \tag{17}$$

where f_{jz} represents the total computing resources of the base station. Here, we consider that tasks can be transmitted and executed simultaneously. Due to the limited latency of user tasks, the allocated computing tasks must be completed within time T. Therefore, the latency constraint can be expressed as:

$$\max\left\{T_i^{\mathrm{loc}}, T_{i1}^{\mathrm{uav}} + T_{i1}^{c}, \cdots, T_{ij}^{\mathrm{uav}} + T_{ij}^{c}, \cdots, T_{i\varsigma}^{\mathrm{uav}} + T_{i\varsigma}^{c}, T_i^{jzul} + T_i^{jzc}\right\} \le T \tag{18}$$

2.4 Energy Consumption Model

Due to the fixed location of the nearby base station, its energy consumption is not considered as it is directly powered by the power supply. At the same time, in order to simplify the model, we consider that the mobile energy consumption of user equipment and ground jammers is provided by additional batteries, and their mobile energy consumption is not considered here. The system energy consumption only considers the user's local computing energy consumption, ground jammer interference energy consumption, UAV edge computing task execution energy consumption and flight energy consumption. The calculation of flight energy primarily accounts for the power used during both hovering and low-speed, constant-velocity flight. As the energy consumption of low-speed constant speed flight is smaller than that of hovering flight, and the model in this paper finds the optimal hovering position, the drone is in a hovering service state. For the sake of model simplicity, our analysis exclusively accounts for the energy consumed during hovering. The energy E_j^U consumed by auxiliary drone j during hovering flight can be expressed as follows [29]:

$$E_j^U = \frac{n_r \times (Mg)^{\frac{3}{2}}}{\sqrt{2L\pi\beta^2}} \times T_j^U \tag{19}$$

The hovering energy model accounts for several key parameters: the total quantity of rotors n_r, the overall weight of the UAV M, the radius of an individual rotor disk β, the gravitational constant g and the fluid density of air L. The total duration of the UAV's hover is represented by T_j^U.

In parallel, we model the computational power (p_{loc}) of a user device using the equation $p_{loc} = \kappa \cdot (f_0)^3$, which shows a cubic relationship with its local operating frequency f_0. In this model, the coefficient κ signifies

the effective switching capacitance. Based on this relationship, we can define the total computational energy that user device i consumes as follows:

$$E_i^{\text{loc}} = p_{\text{loc}} \times T_i^{\text{loc}} \tag{20}$$

The energy consumption due to interference from auxiliary drone j's electromagnetic emissions during the task offloading of user device i is formulated as:

$$E_{ij}^{jam} = p_{jam} \times T_{ij}^{\text{uav}} \tag{21}$$

The energy consumption of the interference generated by the electromagnetic waves emitted by the ground jammer l during the unloading process of user equipment i can be expressed as follows:

$$E_{ij}^{\text{dm}} = p_{\text{dm}} \times T_{ij}^{\text{uav}} \tag{22}$$

Here, p_u is the transmission power of offloading data from the user device to the auxiliary drone and neighboring base stations. The transmission power consumption between the auxiliary drone j and the user device i can be expressed as:

$$E_{ij}^{\text{uav}} = p_u \times T_{ij}^{\text{uav}} \tag{23}$$

The transmission power consumption between neighboring base stations and user equipment i can be expressed as follows:

$$E_i^{\text{ul}} = p_{\text{u}} \times T_i^{\text{jzul}} \tag{24}$$

The computational power of the drone follows the model $p_i^c = \upsilon \times \left(f_{ij}\right)^3$ [30], where υ represents the power consumption coefficient. This coefficient's value is dependent on the CPU architecture of the drone's onboard edge server. Consequently, the energy E_{ij}^c that auxiliary drone j expends to execute user i's task can be expressed as follows:

$$E_{ij}^c = p_i^c \times T_{ij}^c \tag{25}$$

The combined average energy expenditure for both user devices and auxiliary drones is given by the following expression:

$$E_0 = \frac{1}{M_1} \sum_{j=1}^{M_1} \sum_{i=1}^{N_1} \left(E_{ij}^{jam} + E_{ij}^{\text{uav}} + E_{ij}^c\right) + \frac{1}{N_1} \sum_{i=1}^{N_1} \left(E_i^{loc} + E_i^{ul}\right) \tag{26}$$

The overall energy expended by ground jammer l is calculated as follows:

$$E_l^{\text{dm}} = \sum_{i=1}^{N_1} \sum_{j=1}^{M_1} E_{ij}^{\text{dm}} \tag{27}$$

2.5 Problem Formulation

In this paper, we designate E_1, E_2, and E_3 as the total energy for each auxiliary drone, ground jammer, and user device, respectively. The objective is to simultaneously tackle three primary challenges for the system which includes user devices, auxiliary drones, and ground jammers: reducing overall energy expenditure,

minimizing user-experienced delays, and enhancing the security of communication pathways. The system's optimization goal is formulated as a ratio. The numerator represents the UAV's secure data transmission rate. The denominator is the sum of three components: the average energy for UAV edge computation, the average interference energy from ground jammers, and the average user delay. It is important to note that the power consumed by a UAV during hovering flight is not factored into the system's total energy consumption. This is because the energy for hovering depends directly on its duration and does not directly correlate with user delay or the secure data transmission rate, thus it cannot be an active part of the optimization. Consequently, the drone's hovering energy consumption is treated as a constraint within the optimization problem. This allows the drone to operate at a location that maximizes the system's objective, thereby ensuring peak efficiency throughout its operational period. In essence, The optimization problem can be modeled as problem P1:

$$\textbf{P1:} \quad \max \eta = \left(\frac{\sum_{i=1}^{N_1} \sum_{j=1}^{M_1} R_{ij}^{\text{sec}}}{E_0 + \frac{1}{N_2} \sum_{l=1}^{N_2} E_l^{\text{dm}} + \frac{1}{N_1} \sum_{i=1}^{N_1} T_i} \right) \tag{28}$$

$$\text{s.t.} \quad C1 : \varepsilon_i^{\text{loc}}, \varepsilon_{ij}^{\text{uav}}, \varepsilon_i^{\text{ul}} \in [0, 1] \tag{29}$$

$$C2 : \varepsilon_i^{\text{loc}} + \sum_{j \in M_1} \varepsilon_{ij}^{\text{uav}} + \varepsilon_i^{\text{ul}} = 1 \tag{30}$$

$$C3 : x_i, y_i, x_l, y_l, x_u, y_u \in [0, 2P]; x_j, y_j \in [0, P] \tag{31}$$

$$C4 : \sum_{i=1}^{N_1} f_{ij} \leq f_{\max}^{UAV} \tag{32}$$

$$C5 : \sum_{i=1}^{N_1} f_i^{jz} \leq f_{jz} \tag{33}$$

$$C6 : E_l^{\text{dm}} \leq E_2 \tag{34}$$

$$C7 : E_0 \leq E_1 + E_3 \tag{35}$$

$$C8 : \max\left\{ T_i^{\text{loc}}, T_{i1}^{\text{uav}} + T_{i1}^{c}, \cdots, T_{ij}^{\text{uav}} + T_{ij}^{c}, \cdots, T_{i\varsigma}^{\text{uav}} + T_{i\varsigma}^{c}, T_i^{jzul} + T_i^{jzc} \right\} \leq T \tag{36}$$

Constraints $C1$ and $C2$ indicate that the task is collaboratively executed by the user, drone, and neighboring base stations. Constraint $C3$ is to ensure that users, drones, ground jammers, and neighboring base stations are within the designated area. Constraints $C4$ and $C5$ are to ensure that the allocated computing resources are within the range of the total computing resources. Constraining $C6$ and $C7$ is to ensure that the energy consumed is within the range of the total available energy. Constraint $C8$ indicates that the calculation tasks assigned by the user must be completed within a fixed time.

3 K-SAC Algorithm Based on K-Means and SAC

Due to the high-dimensional, dynamic, and non discrete nature of the state space and action space of the optimization objectives in this paper, deep reinforcement learning algorithms are not only more versatile and scalable compared to RL algorithms [31] and traditional objective optimization algorithms, but also perform better in handling complex dynamic systems, long-term planning, and high-dimensional perception tasks. Among numerous deep reinforcement learning algorithms, SAC algorithm not only introduces entropy regularization term to encourage policy exploration of more possibilities, but also allows

the use of experience replay, which has stronger exploration ability and higher sample efficiency, making it a powerful tool for solving high-dimensional continuous control tasks. We choose to use the improved SAC algorithm to solve the objective function and obtain the optimal objective value. The global sampling optimization of CEM [32,33] was added to the action distribution generated by the policy network in the SAC algorithm framework. The user device in this paper is movable and its location changes over time, and the optimal placement of the ground jammer will also change accordingly. However, during the task calculation process, their positions remain unchanged. To obtain the optimal target value by solving the objective function, it is necessary to first determine the positions of the user equipment and ground jammers. To solve this problem and ensure the optimal target value within a certain period of time, we combine the K-means algorithm [34,35] with the SAC algorithm to construct the K-SAC algorithm.

3.1 K-SAC Algorithm Framework

Fig. 2 illustrates the structure of the K-SAC algorithm presented herein, which is composed of seven primary components: a proxy, CEM-based global sampling optimization, an experience buffer, a participant network, a criticism network, a target criticism network, and a K-means algorithm. The procedure commences with the K-means algorithm, which determines the ideal placement for the ground jammer by clustering user device locations, thereby establishing the initial interactive setting. Following this, the K-SAC agent inputs this initial environmental state into the participant network to generate a preliminary action, a_t. This action is then refined using the CEM's distribution-based global sampling before being implemented. This optimization process involves four main steps: first, generating multiple candidate actions by adding random noise to the initial action, second, evaluating each candidate with two Critic networks and selecting the higher Q-value, third, identifying the optimal percent xe of actions as elite samples based on their Q-values, the percentage xe can be adjusted according to the specific situation, and fourth, computing the mean of these elite samples to produce the final optimized action for execution. The agent's action causes the environment to transition to a new state. This entire transition, captured as a tuple (s_t, a_t, r_t, s_{t+1}), is stored in a replay buffer. As this cycle repeats, the buffer populates with experience samples. Specifically, at each decision step, the Actor network first produces a preliminary action based on the current state. This action is then refined using the CEM by generating a set of candidate actions through perturbations around the preliminary action. Each candidate action is evaluated by two independent critic networks, which estimate its action-value function $Q_{w1}(s, a)$ and $Q_{w2}(s, a)$, respectively. To obtain a conservative and robust value estimate, the minimum of the two Q-values is taken for each candidate action, i.e., $Q_{cand}(s, a) = \min(Q_{w1}(s, a), Q_{w2}(s, a))$. This clipped double-Q evaluation prevents overestimation and ensures stable ranking of candidate actions. The candidate actions are then ranked according to their clipped Q-values. The parameter xe denotes the elite ratio, which specifies the proportion of top-ranked candidate actions selected as elite samples. Specifically, if N candidate actions are generated, the top $\lceil xe * N \rceil$ actions with the highest clipped Q-values are selected as elites. The value of xe is a predefined hyperparameter and can be adjusted according to the optimization accuracy and computational complexity requirements. Finally, the optimized action is obtained by computing the mean of these elite actions and is executed in the environment. The resulting transition is stored in the replay buffer and used to update both critic networks and the Actor network following the standard SAC training procedure. Once a minimum storage threshold is met, a batch of G tuples is randomly drawn from the buffer, for which the target network will then perform calculations as follows:

$$y_i = r_i + \gamma \min_{j=1,2} Q_{\omega_j^-}(s_{i+1}, a_{i+1}) - \alpha \log \pi_\theta(a_{i+1}|s_{i+1}) \tag{37}$$

Then calculate the loss function as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q_{\omega_j}(s_i, a_i)\right)^2 \tag{38}$$

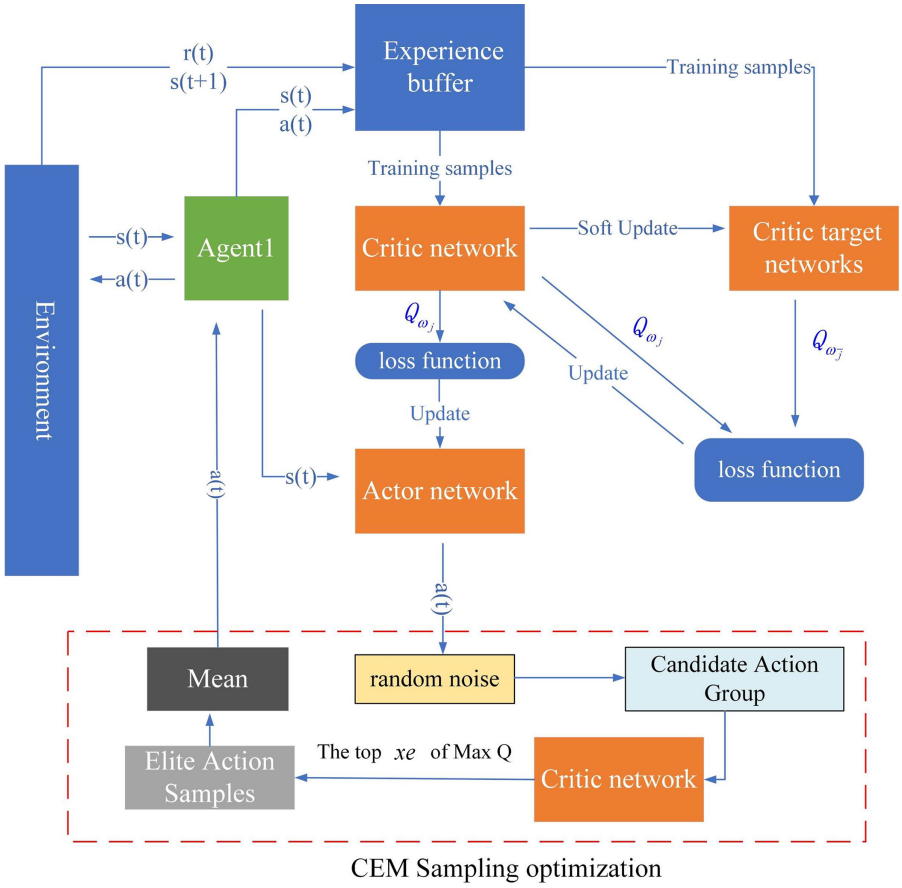

Figure 2: K-SAC algorithm framework.

The continuous nature of the algorithm's action space requires the policy network to output the mean and standard deviation of a Gaussian distribution. A key issue is that action sampling based on this distribution is a non-differentiable operation. Thus, we employ a reparameterization trick to sample actions, which enables the use of the following loss function:

$$L_\pi(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\alpha \log \pi_\theta(\tilde{a}_i|s_i) - \min_{j=1,2} Q_{\omega_j}(s_i, a_i)\right) \tag{39}$$

Update the current Actor network, finally update the target network as follows:

$$\omega_1^- \leftarrow \tau\omega_1 + (1-\tau)\,\omega_1^- \tag{40}$$

$$\omega_2^- \leftarrow \tau\omega_2 + (1-\tau)\,\omega_2^- \tag{41}$$

After updating all networks, the next action is obtained through the updated Actor network, and the above process is repeated until convergence or reaching the predetermined training steps to obtain the optimal target value and the current strategy.

3.2 Markov Decision Process

The deep reinforcement learning algorithm is defined by the following fundamental elements: state, action, and reward. (1) State Space S: The environment state at time step t, denoted as s_t, is given by the vector $s_t = \left\{m_{jt}, w_{lt}, \varepsilon_{it}^{\text{loc}}, \varepsilon_{ijt}^{\text{uav}}, \varepsilon_{it}^{\text{ul}}, E_{jt}, E_{lt}\right\}$. This vector includes the horizontal positions of auxiliary drone j (m_{jt}) and ground jammer l (w_{lt}), the task allocation ratios for user i ($\varepsilon_{it}^{\text{loc}}$, $\varepsilon_{ijt}^{\text{uav}}$, $\varepsilon_{it}^{\text{ul}}$), and the remaining energy of drone j (E_{jt}) and jammer l (E_{lt}) at that time instant. (2) Action Space A: The agent's action at time t is defined as $a_t = \left\{A_{it}^{d\to\text{loc}}, A_{ijt}^{d\to uav}, A_{it}^{d\to ul}, m_{jt}\right\}$, subject to the following constraints:

$$A_{it}^{d\to\text{loc}} + \sum_{j=1}^{M_1} A_{ijt}^{d\to uav} + A_{it}^{d\to ul} = 1 \tag{42}$$

where $A_{it}^{d\to\text{loc}}$ denotes the fraction of computation performed locally by user i, $A_{ijt}^{d\to uav}$ is the portion offloaded to auxiliary UAV j, and $A_{ijt}^{d\to ul}$ is the share sent to adjacent base stations. The term m_{jt} indicates the horizontal coordinates of auxiliary UAV j at time t. (3) Reward R: Following each action, the agent receives a reward from the environment. In order to achieve the objective of maximizing overall system benefits, we set the reward to be equivalent to the system benefit η. The discount factor γ is a hyperparameter defined in the range (0, 1). When γ is close to 0, the agent focuses on short-term gains, while a γ value near 1 prompts it to weigh future outcomes more heavily. The overall computational complexity of the proposed K-SAC algorithm is mainly determined by the policy network updates, the double critic network updates, and the candidate action evaluation based on the CEM. Specifically, the total complexity can be expressed as $O\left(T \cdot K \cdot \left(G \cdot L_1 L_2^2 + N_s \cdot L_1 L_2^2\right)\right)$, where T denotes the total number of training time steps, K is the number of parameter update iterations per time step, G represents the batch size for each update, L_1 and L_2 denote the number of network layers and the number of neurons per layer, respectively, and N_s is the number of candidate actions sampled by the CEM during action selection. In contrast, the standard SAC algorithm evaluates only a single action at each time step, and its computational complexity can be approximated as $O\left(T \cdot K \cdot G \cdot L_1 L_2^2\right)$. Therefore, compared with SAC, K-SAC introduces an additional computational cost that scales linearly with N_s. However, this additional cost does not change the overall polynomial order of the algorithm. From a scalability and deployment perspective, the value of N_s and the network size can be flexibly adjusted according to the system scale and available computational resources. In practice, moderate values of N_s are sufficient to provide effective planning-enhanced action selection, yielding noticeable performance gains while keeping the execution time within acceptable limits. Therefore, the proposed K-SAC framework achieves a favorable trade-off between computational complexity and performance improvement, making it suitable for practical UAV-assisted MEC systems. The pseudocode of the proposed algorithm is illustrated in Algorithm 1.

Algorithm 1: k-SAC algorithm

Require: Location of drones, users, and jammers, task offloading status of each user device, E_1, E_2, E_3, $f_{\max}^{UAV}$, v_0, v_1, f_{jz}.

Ensure: Maximum system benefit and corresponding optimal action vector and auxiliary UAV hovering position.

1: Initialize $Q_{\omega_1}(s,a)$, $Q_{\omega_2}(s,a)$, $\mu_\theta(s)$, $Q_{\omega_1^-}$, $Q_{\omega_2^-}$, R, τ, γ

(Continued)

Algorithm 1 (continued)

for $e = 1 \rightarrow E$ **do**
  Apply K-means clustering on user locations
  Obtain the initial state s_0 of the environment
  for $t = 1 \rightarrow T$ **do**
    Select action $a_t = \mu_\theta(s_t)$ based on the current strategy
    Generate N_s candidate actions: $a_t^i = a_t + \xi_i$, ξ_i is Noise
    Evaluate candidates using critics: $Q_i = min(Q_{\omega_1}(s_t, a_t^i), Q_{\omega_2}(s_t, a_t^i))$
    Select top $xe * N_s$ as elite actions
    Compute optimized action: a_t = mean(elite actions)
    Execute action a_t, receive reward r_t, and the environmental state changes to s_{t+1}
    Store (s_t, a_t, r_t, s_{t+1}) in replay pool R
    for $k = 1 \rightarrow K$ **do**
      Sample G tuples $\{(s_t, a_t, r_t, s_{t+1})\}$ from R, $t = 1, \cdots, G$
      For each tuple, calculate using the target network (37).
      Calculate the minimum target loss using (38) to update the current Critic network.
      Use reparameterization techniques to sample actions, and then update the current Actor network with the following loss function (39)
      Update target network parameters using (40) and (41)
    end for
  end for
end for

4 Simulation Experiments

In this section, the performance of the proposed K-SAC algorithm is evaluated through comprehensive simulation and compared with different baseline schemes. To evaluate the efficiency of task offloading, we compare the performance of our proposed algorithm with DDPG [36], PPO [37], DDQN [38], and DQN [26] algorithms in terms of task volume and complexity.

4.1 Parameter Settings

The simulation experiments are conducted using Python 3.10. Following the parameters outlined in the reference literature [5,34], we establish a scenario in a 600 m × 600 m open square area. This area is populated with 12 users, 2 auxiliary drones, 1 ground jammer, and 1 invading drone, the last of which is located at (400 m, 400 m). The horizontal positions of the users, who regularly process computationally intensive tasks, are depicted in Fig. 3. The auxiliary and invading drones operate at fixed altitudes of $z_j = 110$ m and $z_u = 130$ m, respectively. The horizontal coordinates for the auxiliary drones are set to (20 m, 200 m) and (200 m, 20 m), while the baseline task size for each user is 20 kbit. The optimal percent xe is 20%.

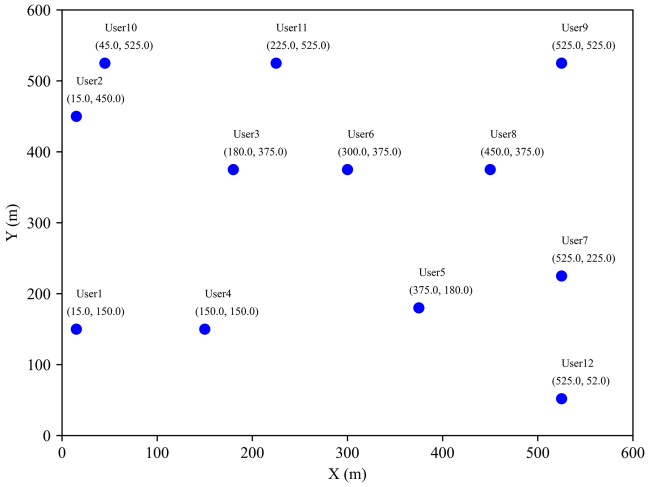

Figure 3: Simulation environment area map.

The K-SAC algorithm utilizes the rectified linear unit (ReLU) activation function for its Actor and Critic networks, whose parameters are specified in Table 3. Given that the two Critic networks share an identical architecture, their parameters are the same. The ReLU function was selected because it is computationally efficient and effectively mitigates the vanishing gradient problem by only propagating the maximum value.

Table 3: Simulation parameters of actor network and critical network.

Network	Number of Layers	Quantity/Piece	Activation Function
Actor network	1	256	Relu
	2	128	Relu
	3	64	Relu
Critic network	1	256	Relu
	2	128	Relu
	3	64	Relu

The remaining simulation parameters are shown in Table 4.

Table 4: Other simulation parameters.

Parameter	Numerical Value	Parameter	Numerical Value
B/MHz	1	Iteration times	1000
β_1	10^{-5}	Time step	400
β_2	10^{-4}	γ	0.99
Task complexity/(cycle/bit)	500	Soft update parameters	0.001
P_{jam}/W	0.2	The learning rate of Critical Network 1	$3e^{-3}$
P_i/W	10^{-2}	The learning rate of Critical Network 2	$3e^{-3}$

(Continued)

Table 4 (continued)

Parameter	Numerical Value	Parameter	Numerical Value
P_u/W	0.2	Playback pool capacity	100,000
σ^2/dbm	−110	Minimum data required for training	1000
χ	10^{-11}	Sampling data volume	128
$f_{\max}^{UAV}$/MHz	1500	$L/\left(\mathrm{kg/m^3}\right)$	1.204
f_0/MHz	300	β/m	0.4
n_r	6	M/Kg	2.5
$g/\left(\mathrm{m/s^2}\right)$	9.8	UAV battery capacity/mAh	10,000
β_3	10^{-5}	Learning rate of actor network	$3e^{-4}$

4.2 Analysis of Simulation Results

In this paper, the model environment considers that users move randomly. In order to provide users with a more secure communication environment, a ground jammer is added to further ensure security. Therefore, a K-SAC proposed scheme based on SAC is proposed. The algorithm first clusters the user's location at a certain moment to obtain the optimal placement position of the ground jammer, forming the initial environment. Then, using the improved SAC algorithm, the optimal offloading strategy and auxiliary drone position corresponding to the optimal system benefits are obtained, and the user's task calculation is completed with the lowest latency and energy consumption in a secure communication environment.

Figs. 4 and 5 present the clustering results under different scenarios. In Fig. 4, subfigures (a–c) illustrate the distribution of clustering centers when the number of clusters is set to 1, 2, and 3, respectively, showing how users are progressively partitioned into finer groups based on spatial proximity. Fig. 5 depicts the variation of a single clustering center after random user movement, where subfigures (a–c) correspond to different stages of user mobility, demonstrating the adaptive capability of the clustering approach to dynamic changes in user distribution. From Fig. 4, it can be seen that as the number of cluster centers increases, the algorithm can find the corresponding number of cluster centers, making the distance between the ground jammer and each user optimal. From Fig. 5, it can be seen that as the user moves randomly, the clustering center position of the algorithm changes accordingly, and an optimal ground jammer position can be found again to achieve the best distance between the ground jammer and each user. This enables the algorithm to find the optimal placement position even when facing the placement of multiple ground jammers and random user movements, resulting in robust performance.

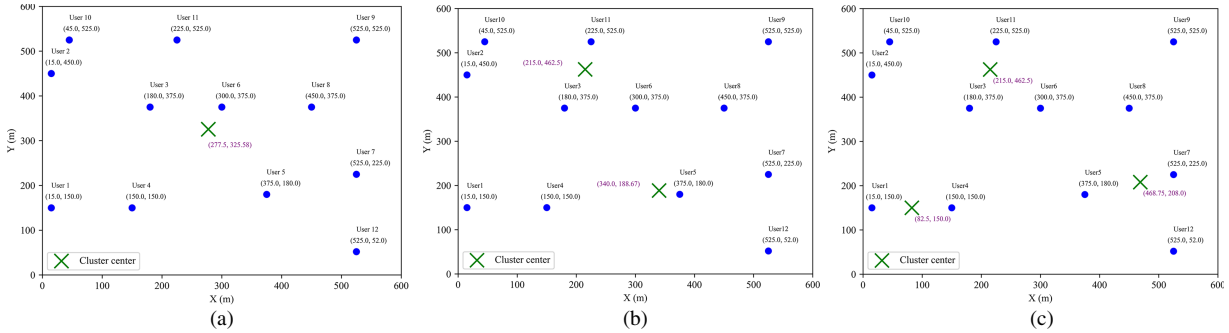

Figure 4: Multicenter clustering graph. (a) Single-center clustering; (b) Dual-center clustering; (c) Three-center clustering.

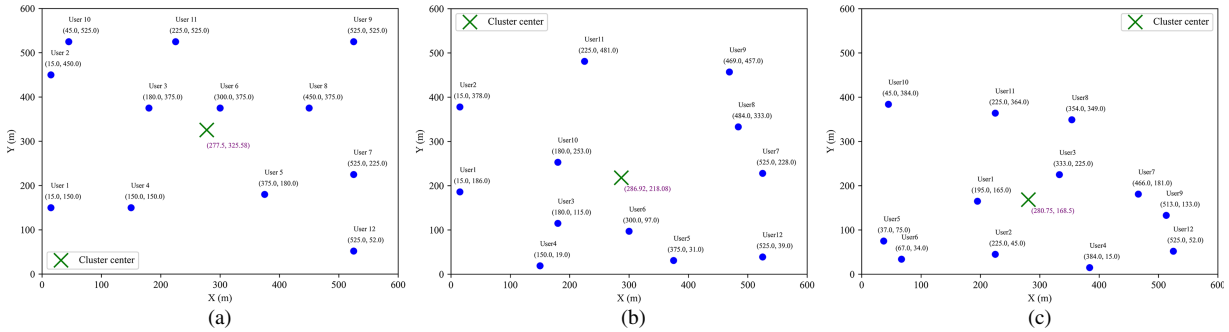

Figure 5: Multicenter clustering graph. (a) Single-center clustering (before user movement); (b) Single-center clustering (after user movement); (c) Single-center clustering (after user moves again).

Fig. 6 shows the user task offloading strategy corresponding to the maximum system benefit when the user task volume in the environment is 6 times the base task volume, the auxiliary drone transmission power is 0.4 w, and the task complexity is 1000 (cycles/bits). Fig. 7 shows the optimal hovering position of the auxiliary drone corresponding to the maximum system benefit. From Fig. 6, it can be observed that when the system efficiency is maximized, users 1 and 11 tend to offload to the base station for execution, while users 2, 6, and 8 tend to offload half to the base station and execute half locally. Users 3, 4, and 10 tend to balance local execution, offloading to base station execution, and offloading to auxiliary drone 1 execution. User 5 tends to balance local execution, offloading to base station execution, and offloading to auxiliary drone 2 execution. Users 7 and 9 tend to balance local execution, offloading to base station execution, offloading to auxiliary drone 1 execution, and offloading to auxiliary drone 2 execution. User 12 tends to unload half to auxiliary drone 2 and execute the other half locally. From Fig. 7, it can be observed that when the system efficiency is at its maximum, the optimal hovering point for auxiliary drone 1 is (10, 600), and the optimal hovering point for auxiliary drone 2 is (600, 10), located at the two diagonals of the ground jammer. These two positions have more users and are far away from invading drones.

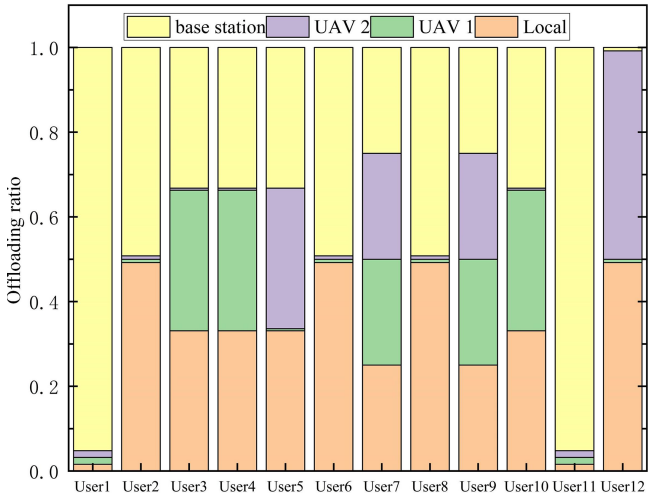

Figure 6: Optimal task offloading ratio of users.

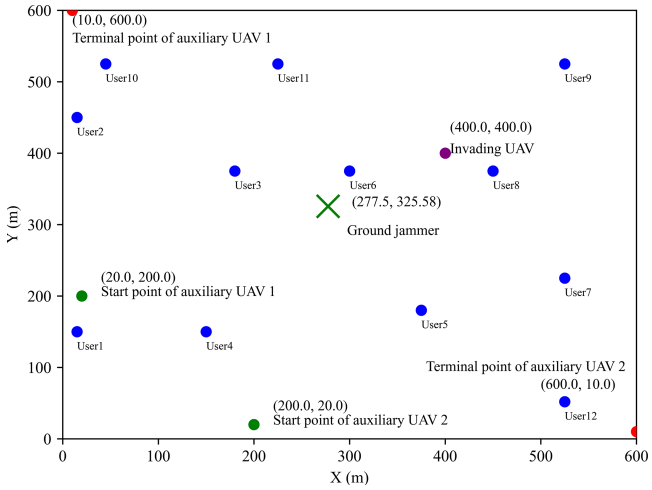

Figure 7: Optimal hovering positions of auxiliary UAVs.

Fig. 8 illustrates the impact of varying auxiliary drone transmission power on the system benefits achieved by different algorithms. The figure reveals a consistent downward trend in system efficiency for all tested algorithms as the drone's transmission power is increased. This trend is primarily attributed to the fact that higher transmission power directly leads to greater transmission energy consumption by the auxiliary drones. While reducing task offloading might seem like a way to conserve energy, it paradoxically leads to longer task completion times. This extension contributes to a higher total of time delay and system energy consumption, ultimately degrading overall system efficiency. The K-SAC algorithm exhibits higher system efficiency under different auxiliary drone transmission powers, mainly due to its superior balancing ability in power control and task offloading decisions. Specifically, as the transmission power of the auxiliary drone increases, the weight of the energy consumption term in the system benefit function gradually increases. At this time, algorithms that rely solely on a single strategy update or single value evaluation (such as PPO, DDPG) are prone to over transmission or conservative offloading, leading to rapid accumulation of energy consumption or time delay. In contrast, K-SAC adopts a dual Critic structure and uses a candidate action

evaluation mechanism to screen actions, enabling the strategy to more accurately evaluate the coupling effect between energy consumption and latency under high-power conditions during the update process, thereby avoiding extreme decisions and maintaining a relatively balanced offloading and power allocation strategy. Therefore, the data in Fig. 8 shows that over the transmission power range of 0.2 to 1, the K-SAC algorithm consistently outperforms the others. It maintains the highest average system efficiency, surpassing PPO by 4.92% and thus showcasing its good performance.

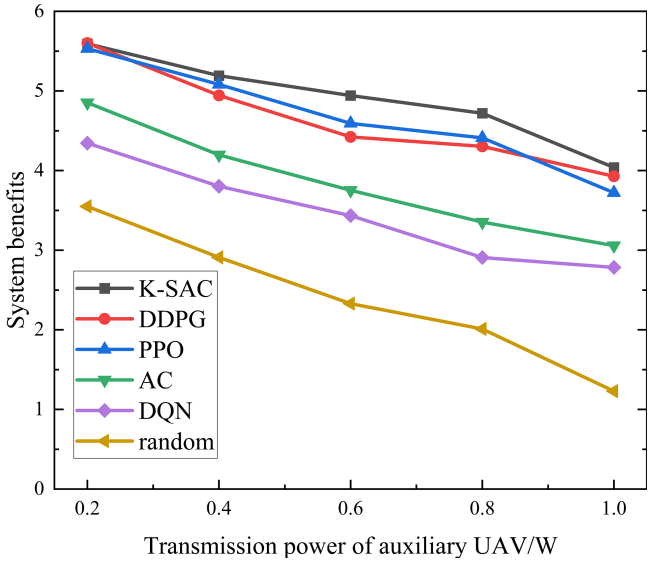

Figure 8: Changes in system benefits with the transmitted power of auxiliary UAV under different algorithms.

Fig. 9 is a simulation diagram of the system benefits under different algorithms as the interference power of auxiliary drones and ground jammers varies. From Fig. 9, it can be seen that the system efficiency of all algorithms decreases with the increase of interference power from auxiliary drones and ground jammers. This is mainly because as the interference power from auxiliary drones and ground jammers increases, the system's interference energy consumption increases. While diminishing the amount of task offloading offers the benefits of shorter transmission times and lower interference energy, this strategy also extends the task completion time. This trade-off ultimately leads to a higher aggregate of time delay and system energy, which in turn degrades overall system efficiency. Furthermore, the performance curves tend to stabilize at high interference power levels. The reason is that high interference causes a substantial rise in the system's energy consumption. To counteract this and maintain efficiency, users strategically reduce their offloading to auxiliary drones. This decision prolongs the task completion time but ultimately results in only a marginal change to the overall system efficiency, explaining the observed stability. Benefiting from its dual-critic structure, K-SAC adopts conservative value estimation to screen candidate actions, enabling more robust offloading decisions under strong interference compared with other algorithms. As a result, K-SAC is more effective in balancing interference energy consumption and task completion delay, leading to a 14.63% improvement in average system benefit when the jamming power increases from 0.1 to 0.5.

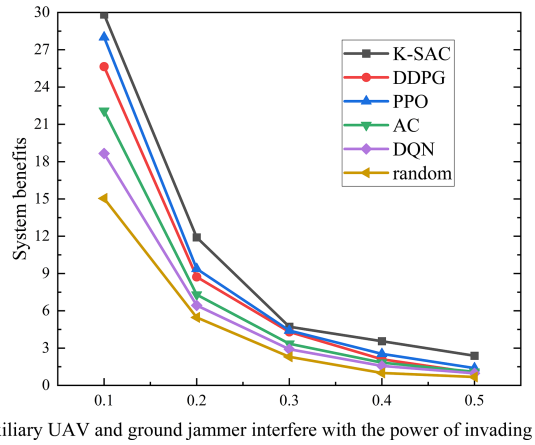

Figure 9: Variation of system benefit with jamming power of auxiliary UAV and ground jammer under different algorithms.

Fig. 10 is a simulation diagram of the system benefits varying with task complexity under different algorithms. Fig. 10 demonstrates a clear trend where the system efficiency of all evaluated algorithms declines as user task complexity grows. The primary reason for this is that more complex tasks necessitate longer computation times. While offloading tasks to auxiliary drones and base stations can shorten computation, this strategy concurrently increases transmission duration, transmission energy, and interference energy. This cumulative increase in both time delay and total energy consumption is what ultimately degrades system efficiency. In this scenario, PPO tends to favor aggressive task offloading to reduce local computation delay, which can significantly increase transmission and interference energy costs. In contrast, K-SAC will adopt a more robust offloading decision. Consequently, K-SAC consistently outperforms PPO when the task complexity increases from 800 to 1200, achieving a 13.67% higher average system benefit.

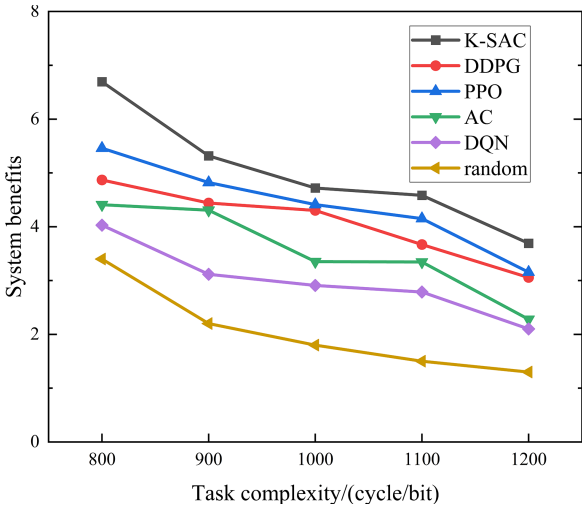

Figure 10: Changes in system benefits with task complexity under different algorithms.

Fig. 11 is a simulation diagram of system benefits varying with the size of user tasks under different algorithms. The results in Fig. 11 indicate a consistent decline in system efficiency for all algorithms as the user task size expands. The primary driver for this trend is the concurrent increase in both computation

and transmission durations required for larger tasks. While offloading to auxiliary drones offers a way to shorten computation, it introduces a penalty in the form of higher transmission and interference energy consumption. This trade-off leads to a higher aggregate of time delay and system energy, thus diminishing overall efficiency. PPO, which updates policies based on local policy gradients, has limited capability to adapt its offloading strategy when task sizes vary significantly. By contrast, K-SAC evaluates multiple candidate actions at each decision step and selects the action with the highest estimated value, enabling it to dynamically adjust offloading ratios according to task scale. This enhanced decision flexibility allows K-SAC to effectively suppress excessive energy consumption under large task sizes, leading to a 9.83% improvement in average system benefit compared with PPO when the task size multiplier increases from 3 to 7.

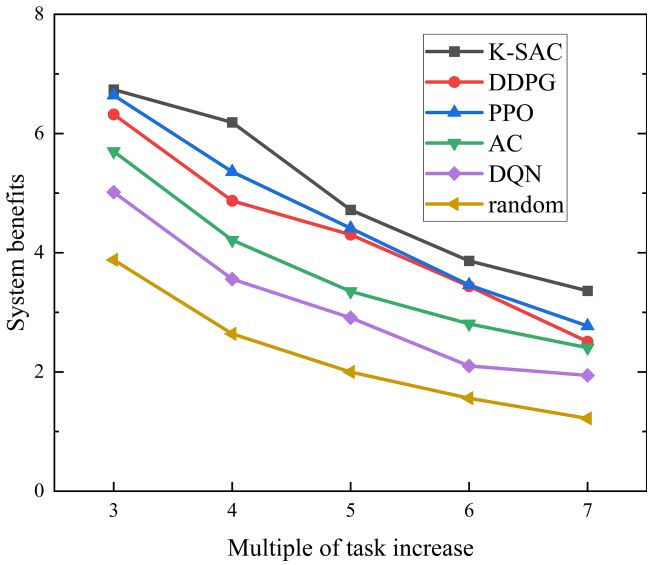

Figure 11: Changes in system benefits with user task size under different algorithms.

5 Conclusion

This paper aims at the problems of system time delay, energy consumption and UAV safe communication in the UAV aided edge computing system with random user movement. First, it constructs the MEC system model of UAV secure communication considering random user movement and adding ground jammers, and then proposes the task offloading scheme of K-SAC algorithm to solve the problem. The simulation results show that the proposed scheme has excellent performance and can reduce system time delay and energy consumption while ensuring communication security in the case of random user movement. Specifically, the algorithm achieves an average efficiency improvement of 9.83% under varying computational task sizes, 13.67% with different task complexities, and 14.63% against various interference powers, demonstrating its capacity to effectively enhance system performance.

However, this study relies on assumptions that may lead to optimistic performance estimates. Specifically, the model presumes precise knowledge of eavesdropper locations, which is challenging to acquire in complex real-world environments where adversaries may be hidden or mobile. In future work, we plan to address this challenge by investigating robust reinforcement learning frameworks that can operate effectively under location uncertainty.

Acknowledgement: Jiajia Liu, Fei Jia and Huibing Zhang acknowledge the support from the Guilin University of Electronic Technology. Shuchen Pang, Peng Xie, Haitao Zhou, Chenxi Du, Haoran Hu, Bo Tang and Jianhua Liu acknowledge the support from Civil Aviation Flight University of China.

Funding Statement: This research was supported by the Laibin City Scientific Research and Technology Development Program Project (No. 241509), Guangxi Key Research and Development Program(AB24010237).

Author Contributions: The individual author contributions are as follows. Jiajia Liu, Peng Xie and Huibing Zhang were responsible for the conceptualization, design, and initial drafting of the manuscript. Data collection, analysis, and interpretation were carried out by Shuchen Pang, Peng Xie, Jiajia Liu, Haitao Zhou, Chenxi Du, Bo Tang, Jianhua Liu, Haoran Hu, Fei Jia. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the Corresponding Author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Nguyen TH, Do VD, Le HL, Nguyen CL, Le DV, Niyato D. Deep reinforcement learning for multi-hop offloading in UAV-assisted edge computing. IEEE Trans Veh Technol. 2023;72(12):16917–22 doi:10.1109/tvt.2023.3292815.
2. Liu J, Tang B, Liu J, Tu X. Dynamic region partitioning and joint optimization for STAR-RIS-assisted UAV edge computing systems via deep reinforcement learning. IEEE Wirel Commun Lett. 2025;15:505–9 doi:10.1109/lwc.2025.3629719.
3. Akter S, Kim DY, Yoon S. Task offloading in multi-access edge computing enabled UAV-aided emergency response operations. IEEE Access. 2023;11:23167–88 doi:10.1109/access.2023.3252575.
4. Kong X, Duan G, Hou M, Shen G, Wang H, Yan X. Deep reinforcement learning-based energy-efficient edge computing for internet of vehicles. IEEE Trans Ind Inform. 2022;18(9):6308–16 doi:10.1109/tii.2022.3155162.
5. Acheampong A, Zhang Y, Xu X. A parallel computing based model for online binary computation offloading in mobile edge computing. Comput Commun. 2023;203:248–61 doi:10.1016/j.comcom.2023.03.004.
6. Liu J, Xie P, Liu J, Tu X. Task offloading and trajectory optimization in UAV networks: a deep reinforcement learning method based on SAC and A-star. Comput Model Eng Sci. 2024;141(2):1243.
7. Shi Z, Wang L, Lin Y, Cai A, Fan J, Liu C. Dynamic offloading strategy in SAGIN-based emergency VEC: a multi-UAV clustering and collaborative computing approach. Veh Commun. 2025;55:100952.
8. Wang L, Zhou Q, Shen Y. Computation efficiency maximization for UAV-assisted relaying and MEC networks in urban environment. IEEE Trans Green Commun Netw. 2022;7(2):565–78 doi:10.1109/tgcn.2022.3222398.
9. Xu Y, Zhang T, Yang D, Liu Y, Tao M. Joint resource and trajectory optimization for security in UAV-assisted MEC systems. IEEE Trans Commun. 2020;69(1):573–88 doi:10.1109/tcomm.2020.3025910.
10. Islam MS, Mahmoud AS, Sheltami TR. AI-enhanced intrusion detection for UAV systems: a taxonomy and comparative review. Drones. 2025;9(10):682.
11. Xiao W, Li M, Alzahrani B, Alotaibi R, Barnawi A, Ai Q. A blockchain-based secure crowd monitoring system using UAV swarm. IEEE Netw. 2021;35(1):108–15 doi:10.1109/mnet.011.2000210.
12. Ghribi E, Khoei TT, Gorji HT, Ranganathan P, Kaabouch N. A secure blockchain-based communication approach for UAV networks. In: Proceedings of the 2020 IEEE International Conference on Electro Information Technology (EIT); 2020 Jul 31–Aug 1; Chicago, IL, USA. p. 411–5.
13. Gai K, Wu Y, Zhu L, Choo KKR, Xiao B. Blockchain-enabled trustworthy group communications in UAV networks. IEEE Trans Intell Transp Syst. 2020;22(7):4118–30 doi:10.1109/tits.2020.3015862.
14. Usman M, Amin R, Aldabbas H, Alouffi B. Lightweight challenge-response authentication in SDN-based UAVs using elliptic curve cryptography. Electronics. 2022;11(7):1026 doi:10.3390/electronics11071026.

15. Devi P, Bharti MR, Gautam D. A survey on physical layer security for 5G/6G communications over different fading channels: approaches, challenges, and future directions. Veh Commun. 2025;53:100891 doi:10.1016/j.vehcom.2025.100891.
16. Cheng Q, Zhou Y, Liu H, Yang L, Ma Z, Fan P. Physical layer authentication in UAV communications with channel randomness and Jamming Uncertainty. IEEE Trans Veh Technol. 2025;74(6):9894–8 doi:10.1109/tvt.2025.3532982.
17. Zhou Y, Pan C, Yeoh PL, Wang K, Elkashlan M, Vucetic B. Secure communications for UAV-enabled mobile edge computing systems. IEEE Trans Commun. 2020;68(1):376–88 doi:10.1109/tcomm.2019.2947921.
18. Lu W, Ding Y, Gao Y, Chen Y, Zhao N, Ding Z. Secure NOMA-based UAV-MEC network towards a flying eavesdropper. IEEE Trans Commun. 2022;70(5):3364–76 doi:10.1109/tcomm.2022.3159703.
19. Zhang Y, Kuang Z, Feng Y, Hou F. Task offloading and trajectory optimization for secure communications in dynamic user Multi-UAV MEC systems. IEEE Trans Mob Comput. 2024;23(12):14427–40 doi:10.1109/tmc.2024.3442909.
20. He Y, Xiang K, Cao X, Guizani M. Task scheduling and trajectory optimization based on fairness and communication security for multi-UAV-MEC system. IEEE Internet Things J. 2024;11(19):30510–23 doi:10.1109/jiot.2024.3412825.
21. Miao J, Chen H, Li H, Bai S. Secrecy energy efficiency enhancement in UAV-assisted MEC system. Sensors. 2023;23(2):723 doi:10.3390/s23020723.
22. Lu F, Liu G, Lu W, Gao Y, Cao J. Resource and trajectory optimization for UAV-relay-assisted secure maritime MEC. IEEE Trans Commun. 2023;72(3):1641–52 doi:10.1109/tcomm.2023.3330884.
23. Zhong L, Liu Y, Deng X, Wu C, Liu S, Yang LT. Distributed optimization of multi-role UAV functionality switching and trajectory for security task offloading in UAV-assisted MEC. IEEE Trans Veh Technol. 2024;73(12):19432–47 doi:10.1109/tvt.2024.3434354.
24. Ding Y, Feng Y, Lu W, Zheng S, Zhao N, Meng L. Deep reinforcement learning-based trajectory optimization and resource allocation for secure UAV-enabled MEC networks. In: Proceedings of the IEEE INFOCOM 2024—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2024 May 20–24; Vancouver, BC, Canada. p. 01–5.
25. Hu H, Hao S, Wang Q, Zhu C, Peng F, Zhou F. Robust trajectory and task allocation in secure UAV-assisted MEC system with cooperative jamming. In: Proceedings of the 2023 IEEE 23rd International Conference on Communication Technology (ICCT); 2023 Oct 20–22; Wuxi, China. p. 1342–7.
26. Lei H, Yang M, Jiang J, Park KH, Pan G. Secure offloading in NOMA-aided aerial MEC systems based on deep reinforcement learning. IEEE J Miniaturization Air Space Syst. 2025;6(2):113–24 doi:10.1109/jmass.2024.3479456.
27. Lu W, Ding Y, Gao Y, Hu S, Wu Y, Zhao N. Resource and trajectory optimization for secure communications in dual unmanned aerial vehicle mobile edge computing systems. IEEE Trans Ind Inform. 2021;18(4):2704–13 doi:10.1109/tii.2021.3087726.
28. Ding Y, Feng Y, Lu W, Zheng S, Zhao N, Meng L. Online edge learning offloading and resource management for UAV-assisted MEC secure communications. IEEE J Sel Top Signal Process. 2022;17(1):54–65 doi:10.1109/jstsp.2022.3222910.
29. Wei X, Cai L, Wei N, Zou P, Zhang J, Subramaniam S. Joint UAV trajectory planning, DAG task scheduling, and service function deployment based on DRL in UAV-empowered edge computing. IEEE Internet Things J. 2023;10(14):12826–38 doi:10.1109/jiot.2023.3257291.
30. Wang Y, Sheng M, Wang X, Wang L, Li J. Mobile-edge computing: partial computation offloading using dynamic voltage scaling. IEEE Trans Commun. 2016;64(10):4268–82.
31. Wang M, Shi S, Gu S, Gu X, Qin X. Q-learning based computation offloading for multi-UAV-enabled cloud-edge computing networks. IET Commun. 2020;14(15):2481–90.
32. Bjerkebæk I, Toftaker H. Reliability assessment combining importance resampling and the cross entropy method. Electr Power Syst Res. 2024;234:110722 doi:10.1016/j.epsr.2024.110722.
33. Nguyen HT, Tran K, Luong NH. Combining soft-actor critic with cross-entropy method for policy search in continuous control. In: Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC); 2022 Jul 18–23; Padua, Italy. p. 1–8.

34. Ahmed M, Seraj R, Islam SMS. The k-means algorithm: a comprehensive survey and performance evaluation. Electronics. 2020;9(8):1295.
35. Ay M, Özbakır L, Kulluk S, Gülmez B, Öztürk G, Özer S. FC-Kmeans: fixed-centered K-means algorithm. Expert Syst Appl. 2023;211:118656.
36. Kumar AS, Zhao L, Fernando X. Task Offloading and resource allocation in vehicular networks: a lyapunov-based deep reinforcement learning approach. IEEE Trans Veh Technol. 2023;72(10):13360–73 doi:10.1109/tvt.2023.3271613.
37. Yang S, Liu J, Zhang F, Li F, Chen X, Fu X. Caching-enabled computation offloading in multi-region MEC network via deep reinforcement learning. IEEE Int Things J. 2022;9(21):21086–98 doi:10.1109/jiot.2022.3176289.
38. Peng Y, Liu Y, Li D, Zhang H. Deep reinforcement learning based freshness-aware path planning for UAV-assisted edge computing networks with device mobility. Remote Sens. 2022;14(16):4016–24. doi:10.3390/rs14164016.