

Article

Research on Joint Optimization of Task Offloading and UAV Trajectory in Mobile Edge Computing Considering Communication Cost Based on Safe Reinforcement Learning

Yu Dai ¹, Jiaming Fu ¹, Zhen Gao ² and Lei Yang ^{2,*}

¹ School of Software, Northeastern University, Shenyang 110167, China; daiy@swc.neu.edu.cn (Y.D.); 2171303@stu.neu.edu.cn (J.F.)

² School of Computer Science and Engineering, Northeastern University, Shenyang 110167, China; 2010672@stu.neu.edu.cn

* Correspondence: yanglei@mail.neil.edu.cn; Tel.: +86-13516045851

Abstract: Due to CPU and memory limitations, mobile IoT devices face challenges in handling delay-sensitive and computationally intensive tasks. Mobile edge computing addresses this issue by offloading tasks to the wireless network edge, reducing latency and energy consumption. UAVs serve as auxiliary edge clouds, providing flexible deployment and reliable wireless communication. To minimize latency and energy consumption, considering the limited resources and computing capabilities of UAVs, a multi-UAV and multi-edge cloud system was deployed for task offloading and UAV trajectory optimization. A joint optimization model for computing task offloading and UAV trajectory was proposed. During model training, a UAV communication mechanism was introduced to address potential coverage issues for mobile user devices through multiple UAVs or complete coverage. Considering the fact that decisions made by UAVs during trajectory planning may lead to collisions, a MADDPG algorithm with an integrated safety layer was adopted to obtain the safest actions closest to the joint UAV actions under safety constraints, thereby avoiding collisions between UAVs. Numerical simulation results demonstrate that the optimization method based on safety reinforcement learning considering communication cost outperforms other optimization methods. Communication between UAVs effectively addresses the issue of redundant or incomplete coverage for mobile user devices, reducing computation latency and energy consumption for task offloading. Additionally, the introduction of safety reinforcement learning effectively avoids collisions between UAVs.

Keywords: mobile edge computing; UAV path planning; task offloading; communication; collision avoidance



Citation: Dai, Y.; Fu, J.; Gao, Z.; Yang, L. Research on Joint Optimization of Task Offloading and UAV Trajectory in Mobile Edge Computing Considering Communication Cost Based on Safe Reinforcement Learning. *Appl. Sci.* **2024**, *14*, 2635. <https://doi.org/10.3390/app14062635>

Academic Editor: Douglas O'Shaughnessy

Received: 6 February 2024

Revised: 6 March 2024

Accepted: 14 March 2024

Published: 21 March 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Currently, the continuous development of IoT technology and 5G mobile technology has promoted the widespread application of augmented reality (AR), virtual reality (VR), vehicle networking, smart cities, and other applications, which require low latency, high efficiency, and network reliability. However, smart mobile devices themselves have limited central processing units, memory, storage, and computational resources. When faced with these computationally intensive applications, smart mobile devices are unable to process them promptly and effectively, and the high power consumption of these applications also reduces the quality of service (QoS) and quality of experience (QoE).

Mobile edge computing (MEC) has been proposed as a new computing paradigm. By deploying infrastructure with computing capabilities near smart devices, the tasks generated by smart devices can be transferred to these facilities, shortening the distance between mobile devices and servers, thereby reducing task processing latency. It can be seen that MEC can satisfy the low-latency requirements of a large number of terminal

devices for offloading applications [1]. In the MEC environment, IoT devices or mobile users are allowed to offload their tasks to nearby edge servers to overcome the limitations of local computing resources and battery energy. However, under normal circumstances, the deployment of MEC servers is fixed, which means that these edge servers cannot utilize mobility to approach user devices to further reduce device latency and energy consumption.

UAV-assisted communication, as a supplement to ground communication, has attracted increasing attention from academia and industry. By integrating unmanned aerial vehicles (UAVs) into MEC networks, UAV-supported MEC architectures have been proposed, in which UAVs can be considered as user devices with computation tasks to execute, as relays to help user devices offload computation tasks, or as MEC servers to perform computation tasks [2–4]. In summary, due to their flexible deployment, extensive coverage, and reliable wireless communication, UAVs have been widely applied in MEC systems, and by establishing LoS links with ground terminals, UAVs can serve as “flying MEC servers” to provide a considerable amount of task offloading services.

Previous research focusing on UAV-assisted networks predominantly concentrated on communication aspects [5,6], but some studies have addressed UAV-assisted MEC systems, covering topics such as trajectory design [7–9], resource management [10–12], and computation offloading [13–15]. Due to the limited computing resources and sensitive battery consumption constraints of single UAVs, most research considers scenarios involving multiple UAVs. In practical computations, UAVs need to provide service to mobile user devices within a designated area while constantly moving, and different UAV trajectories may lead to varying channel quality, resulting in different communication delays and energy consumption. With limited computational resources, the allocation of computational tasks to UAVs also affects computational latency and energy consumption. Moreover, the rapid consumption of UAV battery power impacts overall system performance. Therefore, it is necessary to consider scenarios with multi-UAV and multi-edge cloud collaboration and to minimize execution latency and energy consumption in the joint optimization problem of flight trajectory and computational task allocation.

In recent years, some studies have attempted to address the joint optimization problem in UAV-assisted MEC systems through reinforcement learning. In reference [16], research explored UAV-assisted MEC systems by optimizing the trajectories of aerial UAV base stations and access control of mobile user devices. An algorithm called the Aerial-Ground MADDPG (AG-MADDPG) algorithm was proposed to find the optimal joint optimization strategy. However, in this system, UAVs do not communicate with each other but take actions based on their own and other UAV positions, neglecting the mobile devices covered by other UAVs, which may lead to issues. In reference [17], a UAV positioning scheme was proposed to optimize links between UAV nodes and optimize UAV positions using a deep Q-learning-based method, but it disregarded the positions of other UAVs and mobile devices, potentially resulting in device redundancy or omission. Reference [18] established a multi-UAV-supported hierarchical flying self-organizing network for MEC, utilizing a block coordinate descent algorithm for resource allocation and trajectory optimization. However, the UAVs did not communicate with each other, potentially resulting in similar issues.

To address the aforementioned challenges, this paper proposes a task offloading and UAV trajectory joint optimization model based on UAV communication mechanisms. By introducing UAV communication mechanisms, UAVs can receive information about covered mobile devices from other UAVs, make decisions based on their observations, make inferences about other UAVs, and received messages, thereby resolving the issue of redundant or omitted coverage of mobile devices.

In reference [19], to reduce the propulsion energy consumption, operational costs of all UAVs, and weighted sum of all ground user energy consumption, joint optimization of user device wake-up time allocation, transmission power, and UAV trajectories was conducted. Given the non-convex nature of the optimization problem, the literature decomposed it into two sub-problems: UAV path optimization and wake-up time allocation optimization, proposing an iterative algorithm to address this issue. Reference [20] introduced an

algorithm based on differential evolution (DE) to minimize the energy consumption of user devices through optimizing UAV trajectories and device transmission scheduling. Reference [21] studied secure video streaming in UAV-assisted MEC systems, aiming to maximize the ratio of video quality to power consumption while meeting secrecy outage probability requirements based on joint optimization of power allocation and UAV trajectories. The system was modeled as a constrained Markov decision process, and a safety policy set was constructed using Lyapunov functions, introducing a safety deep Q network to search for optimal policies to solve the constrained Markov decision process (CMDP) problem. Reference [22] formulated the UAV trajectory optimization problem considering energy constraints of mobile devices and obstacle positions. The problem was transformed into a UAV-agent-based CMDP problem, and a UAV trajectory optimization algorithm based on a safety deep Q network was proposed to address the challenge of energy constraints on mobile devices.

While the above works improved reinforcement learning models integrating safety, mainly by optimizing UAV trajectories to ensure secure communication or solve obstacle avoidance issues, they did not consider the possibility of collisions between UAVs during movement. Therefore, this chapter proposes a task offloading and UAV trajectory joint optimization model that integrates safety reinforcement learning. This model combines the decision-making ability of reinforcement learning with the constraint capability of safety reinforcement learning, obtaining actions for each intelligent agent through the MADDPG algorithm and obtaining joint actions of intelligent agents through a cascade. By projecting them onto a safety action set using a safety layer, safe joint actions are derived, effectively addressing potential collision issues during movement.

To address these challenges, this paper designs a collaborative MADRL method for multiple UAVs and ECs to jointly offload the computing tasks of user devices, and proposes a safe layer and communication mechanism between UAVs to obtain the UAV's motion trajectory and computing task allocation scheme with minimal execution delay and energy consumption. The main contributions of our work are:

- (1) Introducing communication mechanisms between UAVs to enable them to make final decisions based on their observations, estimations, and received messages, solving the problem of ground user devices not being covered or covered repeatedly by multiple UAVs in the training process;
- (2) Integrating a safety layer into the MADDPG algorithm to constrain the UAV's actions and enable the use of safe reinforcement learning to plan UAV trajectories and avoid UAV collisions, solving the problem of high energy consumption and delay caused by UAV collisions in the training process;
- (3) Conducting numerical simulations and demonstrating that the proposed MADDPG-based optimization algorithm outperforms other methods. The experimental results show that the coverage of UAVs for user devices significantly improved and collisions among UAVs were reduced during the training process.

2. Materials and Methods

2.1. System Model

Figure 1 shows a multi-UAV-assisted MEC system with M mobile user devices, N unmanned aerial vehicles (UAVs), and K edge clouds (ECs). The set of mobile user devices is denoted as $E = \{E_1, \dots, E_m\}$, the set of UAVs is represented by $U = \{U_1, \dots, U_n\}$, and the set of edge servers is denoted as $S = \{S_1, \dots, S_k\}$. The system time is uniformly divided into time slots, specifically represented as $T = \{T_1, \dots, T_t\}$. Each user device needs to periodically process computationally intensive and latency-sensitive tasks. In time slot T_i , the computational task generated by user device m can be expressed as $W_m = \{D_m, C_m, \lambda_m\}$, where D_m denotes the task size, C_m represents the number of CPU cycles required to process the task, and λ_m is the task arrival rate, with one computational task generated by each mobile device every second. Mobile device computational tasks are fine-grained and divisible; therefore, a partial offloading model is adopted. The generated

computational tasks can be offloaded to UAVs, and the UAVs then offload part of the tasks to edge clouds based on their own state, achieving collaborative processing between UAVs and ECs.

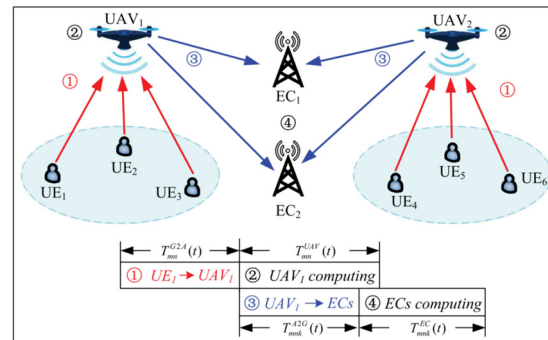


Figure 1. System Model.

2.1.1. UAV Movement Model

In the MEC system, UAVs can move freely. In time slot t , the coordinates of the n th UAV are $\omega_n(t) = [x_n(t), y_n(t), z_n(t)]^T$, where $x_n(t)$, $y_n(t)$ and $z_n(t)$ represent the UAV's coordinates on the X , Y , and Z axes, respectively. Assuming that the flight distance of UAV n in time slot t is $l_n(t)$ and the flight angle is $\theta_n(t) \in [0, 2\pi)$, the position of the UAV in the next time slot using Equations (1) and (2) can be determined.

$$x_n(t+1) = x_n(t) + l_n(t)\cos(\theta_n(t)) \quad (1)$$

$$y_n(t+1) = y_n(t) + l_n(t)\sin(\theta_n(t)) \quad (2)$$

The UAV's altitude can change within a certain range, constrained by Equation (3).

$$Z_{min} \leq z_n(t) \leq Z_{max} \quad (3)$$

where Z_{min} and Z_{max} represent the minimum flight altitude and maximum flight altitude of the UAV, respectively.

Since the horizontal flight speed and vertical flight speed of the UAV are limited, the flight distance within a time slot is also limited. The horizontal and vertical flight distances of the UAV are shown in Equations (4) and (5), respectively.

$$l_n(t) = \|v_n(t+1) - v_n(t)\| \leq L_{max}^h \quad (4)$$

$$\Delta z_n(t) = |z_n(t+1) - z_n(t)| \leq L_{max}^v \quad (5)$$

where L_{max}^h represents the maximum horizontal flight distance of the UAV in a time slot, and L_{max}^v represents the maximum vertical flight distance of the UAV in a time slot.

2.1.2. Communication Model

1. User devices to UAV

Computing tasks generated by mobile user devices cannot be performed on the local device and must be uploaded to the UAV covering the device. The position of user device m is shown in Equation (6). The distance calculation formula between user m and UAV n is shown in Equation (7).

$$\omega_m(t) = [x_m(t), y_m(t), 0]^T \quad (6)$$

$$d_{mn}(t) = \|\omega_n(t) - \omega_m(t)\| \quad (7)$$

The channel gain between mobile device m and UAV n is shown in Equation (8).

$$h_{mn}(t) = \frac{g_0}{[d_{mn}(t)]^2} \quad (8)$$

where g_0 represents the unit power gain. The uplink data transmission rate calculation formula between user m and UAV n is shown in Equation (9).

$$R_{mn}(t) = \frac{B_u}{M_n(t)} \log_2 \left[1 + \frac{h_{mn}(t)P_m}{\sigma_u^2} \right] \quad (9)$$

Here, B_u is the uplink bandwidth, $M_n(t)$ represents the number of user devices covered by UAV n at time slot t , P_m refers to the transmission power of user device m , and σ^2 is the additive white noise power of each UAV.

2. UAV to EC

The fixed coordinates of edge server k are $\omega_k = [x_k, y_k, 0]^T$. The distance formula between UAV n and edge server k can be expressed as:

$$d_{kn}(t) = \|\omega_n(t) - \omega_k\| \quad (10)$$

The channel gain calculation formula between UAV n and edge server k is shown in Equation (11). When transmitting data between UAV n and edge server k , interference caused by other UAVs sending data is not considered. The data transmission rate calculation is shown in Equation (12).

$$h_{kn}(t) = \frac{g_0}{[d_{kn}(t)]^2} \quad (11)$$

$$R_{nk}(t) = B_k \log_2 \left[1 + \frac{h_{nk}(t)P_n^t(t)}{\sigma_e^2} \right] \quad (12)$$

Here, B_k is the bandwidth resource pre-allocated to UAV k by the edge server, $P_n^t(t)$ represents the transmission power of UAV n in time slot t , and $0 \leq P_n^t(t) \leq P_{max}$, where P_{max} is the maximum transmission power of UAV k , and σ_e^2 represents the additive Gaussian noise power at the edge server location.

2.1.3. Computation Model

1. UAV Computation Model

The task is transmitted through the channel between the mobile user device and the UAV, and the transmission delay is as shown in Formula (13). The transmission energy consumption between the mobile device m and the UAV n is as shown in Formula (14).

$$T_{mn}^{G2A}(t) = \frac{D_m}{R_{mn}(t)} \quad (13)$$

$$E_{mn}^{G2A}(t) = P_n^r T_{mn}^{G2A}(t) = \frac{D_m P_n^r}{R_{mn}(t)} \quad (14)$$

Here, P_n^r refers to the receiving power of the UAV.

Upon receiving the complete data of a task from the user device, the UAV makes a task offloading decision and obtains the proportion of tasks calculated locally on the UAV. The task computing delay is as shown in Formula (15).

$$T_{mn}^{UAV}(t) = \frac{\gamma_{m0}^n(t) D_m C_m}{f_{mn}(t)} \quad (15)$$

Here, $f_{mn}(t)$ represents the computing resources allocated by UAV n to user m . Assuming that UAV n evenly allocates computing resources to each user, i.e., $f_{mn}(t) = \frac{F_u}{M_n(t)}$.

$\gamma_{m0}^n(t)$ is the proportion of tasks generated by user device m during time slot t executed on UAV n . If $\gamma_{m0}^n(t) = 0$, it means the UAV offloads all tasks to the edge server, while if $\gamma_{m0}^n(t) = 1$, it means all tasks are executed locally on the UAV. The energy consumption of UAV n processing user device m is as shown in Formula (16).

$$E_{mn}^{UAV}(t) = \kappa [f_{mn}(t)]^3 T_{mn}^{UAV}(t) \quad (16)$$

Here, κ represents the effective switching capacitance.

2. EC Computation Model

If the $\gamma_{m0}^n(t)$ in the previous stage is not 1, it indicates that the UAV will offload part of the tasks to the edge server. The transmission delay in this process is as shown in Formula (17). The energy consumption generated by the UAV uploading task data to the edge server is as shown in Formula (18).

$$T_{mnk}^{A2G}(t) = \frac{\gamma_{mk}^n(t) D_m}{R_{nk}(t)} \quad (17)$$

$$E_{mnk}^{A2G}(t) = P_n^t T_{mnk}^{A2G}(t) = \frac{\gamma_{mk}^n(t) D_m P_n^t}{R_{nk}(t)} \quad (18)$$

Here, P_n^t is the transmission power of UAV n . The task calculation delay on the edge server is as shown in Formula (19).

$$T_{mnk}^{EC}(t) = \frac{\gamma_{mk}^n(t) D_m C_m}{f_{mk}(t)} \quad (19)$$

Here, $f_{mk}(t)$ represents the computing resources allocated by the edge server k to user m . Similar to the UAV calculation model, this study assumes that edge server k evenly allocates computing resources to each user, i.e., $f_{mk}(t) = \frac{F_k^c}{M}$. In summary, when all tasks within the coverage area are completed, the total energy consumption and total delay of the UAV n are as shown in Formulas (20) and (21), respectively.

$$E_n(t) = \sum_{m=1}^M \rho_m^n(t) \lambda_m [E_{mn}^{G2A}(t) + E_{mn}^{UAV}(t) + E_{mnk}^{A2G}(t)] \quad (20)$$

$$T_n(t) = \sum_{m=1}^M \rho_m^n(t) [T_{mn}^{G2A}(t) + \max_k \{T_{mn}^{UAV}(t), T_{mnk}^{A2G}(t) + T_{mnk}^{EC}(t)\}] \quad (21)$$

2.1.4. Problem Modeling

The optimization objective of this paper is to minimize the system's delay and energy consumption. Define $U_n(t)$ as the system utility of UAV n in time slot t , as shown in Formula (22).

$$U_n(t) = w_1 E_n(t) + w_2 T_n(t) \quad (22)$$

Here, w_1 and w_2 are the weight factors indicating energy consumption and delay, both initialized as 1. If $w_1 \geq w_2$ it means more emphasis is placed on the energy-saving situation of the system; otherwise, it means the system is more sensitive to the task delay situation. The objective function of this paper is defined as shown in Formula (23).

$$\min_{w_n(t), \gamma_{m0}^n(t), \gamma_{mk}^n(t), P_n^t(t)} \sum_{n=1}^N U_n(t), \quad (23)$$

$$0 \leq \gamma_{mk}^n(t) \leq 1, \quad (24)$$

$$0 \leq \gamma_{m0}^n(t) \leq 1, \quad (25)$$

$$\gamma_{m0}^n(t) + \sum_k \gamma_{mk}^n(t) = 1, \forall n \quad (26)$$

$$0 \leq P_n^t(t) \leq P_{max} \quad (27)$$

Here, the constraints (24–26) ensure that, in a multi-drone and multi-edge-cloud-assisted MEC system, tasks generated by mobile devices will be partially stored on drones after being collected, with the remaining tasks being sent to other edge cloud base stations. Constraint (27) ensures the maximum transmission power of drones. Limiting the maximum transmission power can effectively address the energy consumption of drones and extend flight time. Reasonable setting of transmission power can protect wireless spectrum resources, avoid spectrum waste and pollution, and promote effective management and utilization of spectrum resources. Equations (3)–(5) are also constraints for the objective function in this paper. Constraints (3)–(5) can restrict the flight height, horizontal, and vertical flight distances of drones. By limiting the flight range of drones, it helps to manage communication resources and energy consumption reasonably. By controlling the flight range, communication resources can be more effectively allocated and the flight time of drones can be extended, enhancing the efficiency and performance of the system. Furthermore, ensuring that drones fly within the appropriate range helps to avoid collisions with other aircraft or obstacles, thus enhancing flight safety.

2.2. MADDPG Solves UAV Path Planning and Task Offloading Problem

Here, we first re-model the aforementioned problem as a Markov decision process (MDP) and then solve it using the MADDPG algorithm.

1. MDP

The joint optimization problem of task offloading and UAV trajectory is modeled as a Markov decision process, denoted as $(N, S, \{A_n\}_{n \in N}, P, \{R_n\}_{n \in N}, \delta)$; the six elements are the set of agents, the set of agent states, the action of the n th agent, the state transition matrix, the reward obtained by the n th agent and the decay factor, whose range is $[0, 1]$, respectively.

2. State

According to the joint optimization problem of path planning and task offloading, the state (State) is composed of task parameters, UAV parameters, and edge server parameters within the time interval t . The state of the t -th time interval is denoted as $s_t = \langle \omega, O_i, W, f, F \rangle$, where ω represents the coordinate matrix of n UAVs, as shown in Formula (28). O_i represents the observation of all user devices by UAV i ; if UAV i can not observe a user device q , then $O_{1,q} = [0, 0, 0, 0]$, otherwise, it is the specific information, as shown in Formula (29). W is the set of tasks in the current time slot. f refers to the computing resources of the UAVs, which can be represented as $f = (f_1, f_2, \dots, f_N)$. F represents the computing resources on the edge servers, denoted by the formula $F = (F_1, F_2, \dots, F_K)$.

$$\omega_1 \begin{bmatrix} x_1 & y_1 & z_1 \\ \omega_2 & x_2 & y_2 & z_2 \\ \dots & \dots & \dots & \dots \\ \omega_n & x_n & y_n & z_n \end{bmatrix} \quad (28)$$

$$O_{i,1} \begin{bmatrix} i & 1 & d_{i,1} & \alpha_{i,1} \\ O_{i,2} & i & 2 & d_{i,2} & \alpha_{i,2} \\ \dots & \dots & \dots & \dots \\ O_{i,m} & i & m & d_{i,m} & \alpha_{i,m} \end{bmatrix} \quad (29)$$

Here, $d_{i,m}$ denotes the distance between UAV i and user device m , $\alpha_{i,m}$ represents the angle between UAV i and user device m .

3. Action

Each UAV needs to determine its motion state (horizontal flight distance l_n , horizontal flight angle θ_n , vertical flight distance Δz_n), transmission power P_n , task allocation ratio γ_{mk}^n and UAV transmission vector $g_{n,s}$. The action of UAV n is represented as:

$$a_n = \{l_n, \theta_n, \Delta z_n, P_n, \gamma_{mk}^n, g_{n,s}\} \quad (30)$$

Here, γ_{mk}^n is the proportion of user devices m in the coverage range of UAV n offloading tasks to edge server k . $g_{n,s}$ is the vector sent by UAV n to other UAVs, with dimensions of $R_{s \times M}$ and specifically the probability inferred by UAV n of all other UAVs covering each mobile user device.

4. Reward Function

In reinforcement learning algorithms, agents need to design appropriate reward functions to obtain the optimal policy. Based on the optimization goals described in formula (23), the reward function in this paper is set as:

$$R_n = -U_n(t) = -w_1 E_n(t) - w_2 T_n(t) \quad (31)$$

This paper aims to train the proposed reinforcement learning algorithm to obtain the optimal strategy and thus minimize the total cost of the system. We set the minimization of the total system cost as the objective because a well-trained reinforcement learning algorithm will generate the best strategy, i.e., the actions of the UAV. The reward values generated by the UAV will gradually increase and converge during training, so we set the negative value of the system total cost as the target reward for reinforcement learning. We hope that the trained algorithm can achieve the maximum reward value, i.e., the minimization of the total system cost.

2.3. Security Reinforcement Learning Algorithm Integrating UAV Communication Mechanism

2.3.1. UAV Communication Mechanism

When using the MADDPG reinforcement learning algorithm to jointly optimize task offloading and UAV path, since there is no communication function between UAVs, each UAV can only make a joint decision on task offloading and UAV path planning based on its observed situation. Therefore, for a UAV, it is not aware of the situation of other UAVs covering mobile user devices. This may result in the problem of multiple UAVs covering the same mobile device or some user devices not being serviced by a UAV. To address this issue, we propose a joint optimization algorithm for task offloading and path planning that incorporates communication mechanisms for UAVs.

The structure of the UAV communication mechanism is as shown in Figure 2, which specifically includes four parts: observation encoding, estimation inference, message sending, and decision-making. The observation encoding part uses the self-attention method to encode the local observations of the UAV because the self-attention method is permutation-invariant and order-invariant, which is beneficial for model expansion. Moreover, due to the weighted sum mechanism, multi-objective information can be encoded into a single feature. Scaled dot-product self-attention is used to transform the local observation O_i into keywords K_i , queries Q_i , and values V_i by three different neural networks. The calculation formula is as shown in (32).

$$E_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) \odot V_i \quad (32)$$

The encoded feature E_i of UAV i and the location information of other UAVs (e.g., ϕ_j, ϕ_k, ϕ_l) are taken as inputs to the estimation and inference part, where $\zeta_{i,j}$ represents the information observed by UAV j as inferred by UAV i . Combining the information observed by UAV i itself with the user device information perceived by drone i to be observed by other drones,

because the structures of the two sets of observation information are both one-dimensional vectors, the way of combination is to concatenate the two sets of one-dimensional vectors. That is, after the combination, the size of the observation space of drone i is expanded to the sum of the lengths of the two sets of vectors. The combined observation information can help drones better assess the situation of the mobile devices covered by other drones.

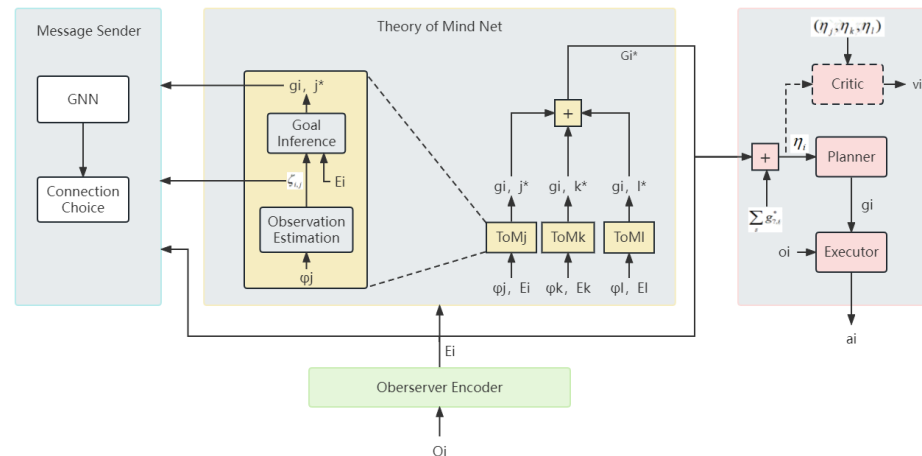


Figure 2. Structure diagram of unmanned aerial vehicle communication mechanism.

Next, $g_{i,j}^*$ is obtained by taking E_i and $\zeta_{i,j}$ as the input of the target inference subpart, which can be represented as $g_{i,j}^* = GI(E_i, \zeta_{i,j})$. $g_{i,j}^*$ is an m -dimensional vector, where m represents the number of user devices, and the vector represents the probabilities of UAV i inferring that each mobile user device is covered by UAV i . The update of the observation estimation subnetwork parameters is achieved through the binary cross-entropy loss function, and the specific calculation is as shown in Formula (33).

$$L^{OE} = -\frac{1}{N} \sum_i \sum_{j \neq i} \sum_q [c_{j,q} \cdot \log(c_{i,j,q}^*) + (1 - c_{j,q}) \cdot \log(1 - c_{i,j,q}^*)] \quad (33)$$

Here, c_{ij}^* is the distance relationship between all user devices and UAV j inferred by UAV i . $c_{j,q}$ represents the real coverage relationship between agent j and device q . If agent j covers device q or UAV j is the closest agent to device q , then $c_{j,q} = 1$; otherwise, $c_{j,q} = 0$.

Similarly, the target inference subnetwork parameters are also updated by the binary cross-entropy loss function, as shown in Formula (33). Here, $g_{i,j,q}^*$ represents the probability that UAV i predicts UAV j chooses target q , and $g_{j,q}$ is the actual situation in which UAV j covers the user device. When $g_{j,q} = 0$, there is only the second term in Formula (34); at this time, the closer the sub-network inference $g_{i,j,q}^*$ is to 0, i.e., the closer it is to the true value, and the smaller the loss value. When $g_{j,q} = 1$, there is only the first term in Formula (34); at this time, the closer the sub-network inference $g_{i,j,q}^*$ is to 1, i.e., the closer it is to the true value, the smaller the loss value. To better represent the gap between the real sample labels and the predicted probabilities, the parameters of the target inference subnetwork are updated by the binary cross-entropy loss function. The loss function of the estimation and inference part is the sum of the loss functions of the two subnetworks, as shown in Formula (35).

$$L^{GI} = -\frac{1}{N} \sum_i \sum_{j \neq i} \sum_q [g_{j,q} \cdot \log(g_{i,j,q}^*) + (1 - g_{j,q}) \cdot \log(1 - g_{i,j,q}^*)] \quad (34)$$

$$L(\theta^{ToM}) = L^{OE} + L^{GI} \quad (35)$$

As graph neural networks (GNNs) have strong learning capabilities, they can capture spatial information hidden in the network topology, and a GNN has the generalization ability to handle unseen topology when the network is dynamic. Message sending is

essentially a communication network, and the basic structure of communication network is a graph. Graph data, a kind of non-Euclidean spatial data, has complex correlations and inter-object dependencies, and a GNN can quickly mine topological information and complex features in graph structure to solve the communication problem between agents. Therefore, the message sending part uses a GNN to implement the communication function between UAVs. This part is modeled end-to-end, with UAVs modeled as graph nodes and the communication relationship between UAVs modeled as edges. First, generate node features $u_{i,j} = (E'_{i,j}, \zeta_{i,j})$, which represent the node features of UAV j from the perspective of UAV i , where $E'_{i,j} = \sum_{q=1}^m (g_{i,j,q}^* > \delta) \cdot E_{i,q}$, and δ represents the probability threshold. Then, the edge features between UAV j and UAV k are generated according to $\sigma(u_{i,j}, u_{i,k})$, where σ is the edge feature encoder. Finally, the feature set $G = (V_i, \xi_i)$ of all nodes and edges in UAV i 's GNN is obtained. In order to reduce communication redundancy, it is necessary to reduce unnecessary edges, and the algorithm determines the validity of each edge through ablation analysis. The specific steps of ablation analysis are: remove a certain edge from $\sum_s g_{s,i}^*$ to obtain the non-connected subgoal g_i^- , where $\sum_s g_{s,i}^*$ represents the message set sent by all other UAVs to UAV i ; then, calculate the divergence value $\chi = D_{KL}(g_i^- || g_i)$ through the KL divergence calculation formula. If $\chi \leq \tau$ (τ is a preset threshold), it means that $\sum_s g_{s,i}^*$ has a small impact on the final decision made by UAV i and belongs to redundant communication, so this edge should be removed; otherwise, keep the edge. The update formula of the message sending part parameters is (36). The reason we chose KL divergence to assess redundant communication is that KL divergence is a measure of the difference between two probability distributions. In reinforcement learning algorithms, communication between agents is achieved by transmitting probability distributions or policies. By comparing the differences in the probability distributions transmitted by different agents, we can determine the existence of redundant communication. The basic principle is that if the probability distributions transmitted by two agents are similar enough, then the communication between them may be redundant because the information from one agent does not provide additional value or help. KL divergence can be used to quantify the difference between two probability distributions, and a smaller KL divergence indicates a higher similarity between the two probability distributions, suggesting the presence of redundant communication.

$$L^{CR} = -\frac{1}{M} \sum_j \sum_i [l_{j,i} \cdot \log(p_{retain}^{j,i}) + (1 - l_{j,i}) \cdot \log(p_{cut}^{j,i})] \quad (36)$$

Here, $l_{j,i}$ represents the edge retention situation between UAV j and UAV i . If the communication between UAV i and j belongs to redundant communication, the edge is removed, and $l_{i,j} = 0$; otherwise, the edge is retained, and $l_{i,j} = 1$. $p_{retain}^{j,i}$ is the retention probability of edge j and i predicted by UAV i , and similarly, $p_{cut}^{j,i}$ is the deletion probability.

The decision part includes three sub-parts. The Critic network can obtain the features of all UAVs $(\eta_1, \dots, \eta_n)$ and calculate the v_i value for evaluating the Planner network, where η_i is specifically shown in Formula (37).

$$\eta_i = (E_i, \max_j g_{i,j}^*, \sum_s g_{s,i}^*) \quad (37)$$

Here, E_i is the encoding value of the UAV i 's own observation situation, $\max_j g_{i,j}^*$ represents the probability of the device most likely covered by UAV j inferred by UAV i , and $\sum_s g_{s,i}^*$ represents the set of messages sent to UAV i from all other UAVs. The input of the Planner network is η_i mentioned above, and the output is the user devices that the UAV should cover. Since η_i includes its own observation situation, inferred coverage targets of other drones, and received coverage situations sent by other UAVs, UAV i possesses relatively comprehensive information and can make decisions on which devices to cover. Finally, the Executor network makes the final action based on the policy $\pi_i^L(a_i | o_i, g_i)$.

2.3.2. Security Reinforcement Learning

(1) Introduction to Security Reinforcement Learning

a. Linear Safety Signal Model

In the absence of prior knowledge of its environment, intelligent agents initialized with random policies cannot guarantee meeting constraints for each state during the initial training phase. In order for agents to learn to avoid unsafe or unexpected actions, they must violate a sufficient number of times for the negative impact to propagate in the algorithm model. Therefore, the linear safety signal model combines some prior knowledge based on basic forms of single-step dynamics, which is implemented as follows in the linearization.

$$\bar{c}(s') = c_i(s, a) \approx \bar{c}_i(s) + g(s; \omega_i)^T a \quad (38)$$

Here, $g(s; \omega_i)$ represents a neural network that takes a state s as input and outputs a vector with the same dimension as action a , which is the extracted feature of the state. ω_i is the weight of the neural network. $\bar{c}(s')$ is the first-order approximation of $c_i(s, a)$ with respect to a , which explicitly represents the sensitivity of actions to changes in the safety signal using state features. Given a policy tuple $D = \{(s_j, a_j, s'_j)\}$, the neural network $g(s; \omega_i)$ is trained using Formula (39).

$$\underset{\omega_i}{\operatorname{argmin}} \sum_{(s, a, s') \in D} (\bar{c}_i(s') - (\bar{c}_i(s) + g(s; \omega_i)^T a))^2 \quad (39)$$

b. Analysis and Optimization of the Safety Layer

Adding a safety layer on top of the deep policy network addresses the action correction optimization issue for each forward pass, as illustrated in the specific model in Figure 3. Let s denote the current environmental state, $\mu_\theta(s)$ denote the intelligent agent's action output by the deterministic policy gradient network. After passing through the safety layer, the resulting safe action $\tilde{\mu}_\theta(s)$ is obtained, which is closest to the original action $\mu_\theta(s)$.

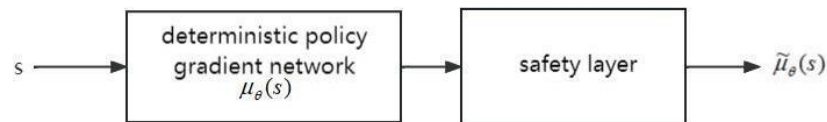


Figure 3. Deep Policy Network with Added Security Layer.

The safety layer aims to minimize interference with the original action in the Euclidean norm to meet necessary constraints. The objective of the safety layer is to find safe intelligent agent actions that satisfy the constraints, as shown in Formula (40).

$$\underset{a}{\operatorname{argmin}} \|a - \mu_\theta(s)\|_2^2 \quad (40)$$

$$\text{s.t. } \bar{c}_i(s) + g(s; \omega_i)^T a \leq C_i \quad \forall i \in [K] \quad (41)$$

To address problem (40), we substitute the safety signal $c_i(s, a)$ with the linear model from Formula (38) to obtain a quadratic programming formulation, as shown in Formula (42). Due to the positive definiteness of the quadratic form and linear constraints, the global solution to this convex problem can be obtained.

$$a^* = \underset{a}{\operatorname{argmin}} \|a - \mu_\theta(s)\|_2^2 \quad (42)$$

$$\text{s.t. } \bar{c}_i(s) + g(s; \omega_i)^T a \leq C_i \quad \forall i \in [K] \quad (43)$$

(2) MADDPG Algorithm with Integrated Security Layer.

The actions obtained by the MADDPG algorithm do not consider the safety of the actions and may cause collisions between agents. Therefore, we integrated a safety layer

into the MADDPG algorithm to constrain the actions of agents, enabling agents to avoid unsafe actions and thereby prevent collisions.

The reinforcement learning we propose with an integrated safety layer is a complete improvement based on the MADDPG algorithm. In the training process, unlike the MADDPG algorithm, which directly outputs the actions of multiple agents, the improved algorithm combines the actions of multiple agents and then imposes safety constraints on the combined action to obtain a safe combined action. The structure of the algorithm is shown in Figure 4. The algorithm combines the actions of all agents into an action vector a , i.e., $a = (a_1, a_2, \dots, a_n)$, and inputs this vector into the safety constraint layer. The safety constraint layer outputs the safety-constrained joint action a^* , where $a^* = (a_1^*, a_2^*, \dots, a_n^*)$, and each component expression is $a_i^* = \{l_i^*, \theta_i^*, \Delta z_i^*, P_i^*, \gamma_{mk}^{i*}, g_{i,s}^*\}$, and $l_i^*, \theta_i^*, \Delta z_i^*$ represent the horizontal flight distance, horizontal flight angle, and vertical flight distance of UAV i after safety constraints. P_i^* is the transmission power of the UAV, and γ_{mk}^{i*} represents the proportion of tasks offloaded to edge servers by the UAV. The first three elements determine the flight trajectory of the UAV, and γ_{mk}^{i*} determines the offloading situation of tasks generated by m mobile devices. By putting these elements in the same action, joint optimization of task offloading and UAV trajectory is achieved. First, the safety constraint layer approximates the constraint function with respect to the action a to the first order, as shown in Formula (44).

$$c_j(x') = \hat{c}_j(x, a) \approx c_j(x) + g(x; \omega_j)^T a \quad (44)$$

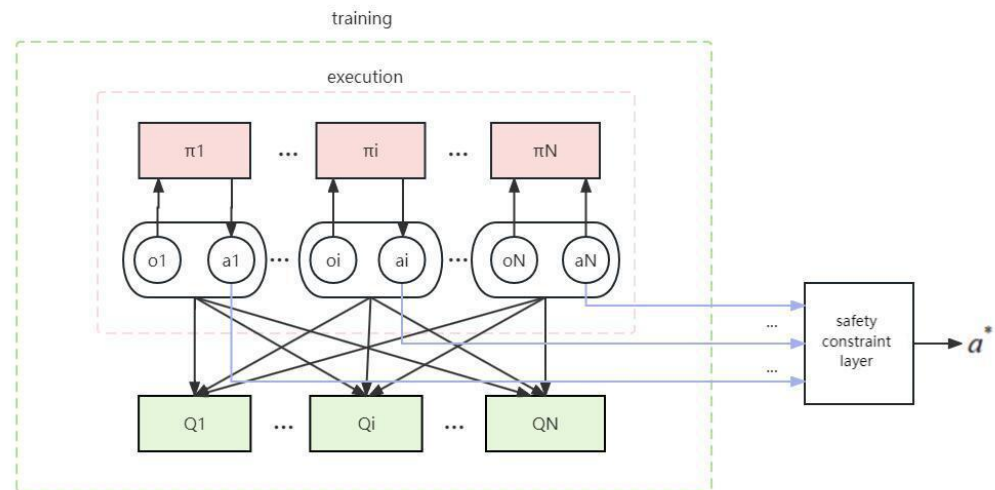


Figure 4. SMADDPG algorithm structure diagram.

Here, g is a neural network, and the single-step transformation data $D = \{(x, a, x')\}$ for training the neural network is generated by randomly initializing the agent's state and selecting actions according to random policies for multiple events. These generated data are used to train each constraint, and the loss function for each constraint is as shown in Formula (45), where each constraint can be trained separately.

$$L(\omega_j) = \sum_{(x,a,x') \in D} (c_j(x') - (c_j(x) + g(x; \omega_j)^T a))^2 \quad (45)$$

The safety signal introduced in Formula (44) adds an additional safety layer to strengthen the policy network. The safety layer constitutes a quadratic programming that computes the minimum distance of the actions taken by each policy network $\pi_{\theta_i}(x_i)$ on the linearized safety set, as shown specifically in Formula (46).

$$\underset{a}{\operatorname{argmin}} \|a - \Pi(x)\|_2^2 \quad (46)$$

$$\text{s.t. } c_j(x) + g(x; \omega_j)^T a \leq C_j \forall j = \{1, \dots, K\} \quad (47)$$

Here, $\Pi(x) = (\pi_{\theta_1}(x_1), \dots, \pi_{\theta_n}(x_n))$ is the cascaded action of all agents without safety constraints. Due to the strong convexity of the optimization problem, a globally unique minimum exists whenever the feasible set is non-empty, which is the safest action closest to the original action. As the formula is generic, even though previous optimization iterations are feasible, there may not exist a recoverable action that can guarantee agents are in a safe state. To solve this problem, we adopt soft constraint formulas, that is, when an action that does not satisfy the safety constraint exists, it can also be executed, but the agent will be given certain penalties. The soft constraint formula is shown in (48).

$$(a^*, \varepsilon^*) = \underset{a, \varepsilon}{\operatorname{argmin}} \|a - \Pi(x)\|_2^2 + \rho \|\varepsilon\|_1 \quad (48)$$

$$\begin{aligned} \text{s.t. } & g(x; \omega_j)^T a \leq C_j - c_j(x) + \varepsilon_i \\ & \varepsilon_j \geq 0 \forall j = \{1, \dots, K\} \end{aligned} \quad (49)$$

Here, $\varepsilon = (\varepsilon_1, \dots, \varepsilon_K)$ is the relaxation variable, and ρ is the penalty weight after violating the constraint. In the algorithm proposed in this chapter, we choose $\rho > \|\lambda^*\|_\infty$, where λ^* is the optimal Lagrange multiplier of the original problem in formula (34). This ensures that the soft constraint problem (48) produces equivalent solutions when the original problem of the formula is feasible. The Algorithm 1 is the SC-MADDPG algorithm process.

Algorithm 1 SC-MADDPG

1. Random initialization the parameter β_i of Online Citric network $Q_i^\pi(x, a; \beta_i)$, The parameter θ_i of the Online Actor network
 2. Random initialization the parameter $\hat{\beta}_i$ of Target Citric network. The parameter $\hat{\theta}_i$ of Target Actor network $\hat{\pi}_i$: $\hat{\beta}_i \leftarrow \beta_i$, $\hat{\theta}_i \leftarrow \theta_i$
 3. Initialize experience replay buffer pool R
 4. **for** episode = 1, M **do**
 5. Random initialization state x^1
 6. **for** $t = 1, T$ **do**
 7. Input the state value x_i^t of each intelligent agent into the Encoder layer and output the state code value E_i
 8. Obtain η_i by solving formula (36) and use it with the agent's state value as input to the policy network
 9. Each agent's action is made through a policy network $\pi_i^L(a_i | o_i, g_i)$ to make the final action a_i^t
 10. Combine the actions of all intelligent agents: $a^t = (a_1^t, \dots, a_n^t)$
 11. Project a^t onto the safe action set by solving the formula
 12. Injecting detection noise n
 13. Execute action a^t , Obtain reward R^t and state x^t for the next time slot
 14. Put $(x^t, a^t, R^t, x^{t'})$ into the experience replay buffer pool R
 15. Randomly select λ samples from the experience replay buffer pool R , with an index of k
 16. Obtained through Target Actor network $\tilde{a} = (\hat{\pi}_1(x_1'; \hat{\theta}_1), \dots, \hat{\pi}_1(x_n'; \hat{\theta}_n))$
 17. Set $z_i^k = R_i^k + \gamma \hat{Q}_i^\pi(x^{k'}, \tilde{a}; \hat{\beta}_i)$
 18. Update the Citric network of all agents by minimizing the loss function:
 19. $L_i = \frac{1}{\lambda} \sum_k (z_i^k - Q_i^\pi(x_k, a; \beta_i))^2$
 20. Sampling strategy gradient update Online Actor network:
 21. $\nabla_{\theta_i} J_i \approx \frac{\sum_k \nabla_{a_i} Q_i^\pi(x^k, a^k; \beta_i) |_{a_i = \pi_i(x_i^k)} \nabla_{\theta_i} \pi_i(x_i^k; \theta_i) |_{x_i^k}}{\lambda}$
 22. Each intelligent agent i updates the Target Citric network and Target Actor network parameters:
 23. $\hat{\beta}_i \leftarrow \tau \beta_i + (1 - \tau) \hat{\beta}_i$
 24. $\hat{\theta}_i \leftarrow \tau \theta_i + (1 - \tau) \hat{\theta}_i$
 25. **end for**
 26. **end for**
-

3. Results

In this section, we conducted numerical experiments to evaluate the performance of our proposed SC-MADDPG. The communication mechanism between UAVs in SC-MADDPG is improved based on the MADDPG algorithm, integrating four parts to achieve UAV communication functions. The observation encoding part consists of two layers of multilayer perceptron (MLP) and a self-attention model; the estimation inference part is composed of a gated recurrent unit (GRU) and two layers of MLP, with the number of hidden layer units in the GRU being 32; the message sending part is implemented using a graph neural network (GNN); the Critic network in the decision part is set to three layers, with the number of neurons being the sum of all UAV state space sizes, 192, and 1, respectively. The Planner network is also set to three layers, with the number of neurons being the state space size of UAV i , 192, and m , respectively. The Executor network has input as UAV i 's observation situation o_i and g_i , and output is the UAV's action. At the same time, a safety layer is added to the MADDPG algorithm, so SC-MADDPG is composed of Actor network, Constraint network, and Critic network. The Constraint network consists of two fully connected layers, with the activation function being the ReLu function, and the initial learning rate is 5×10^{-4} . The Actor network consists of three fully connected layers and two hidden layers, with the number of neurons in the hidden layers being 100. The initial learning rate is 1×10^{-4} . The Critic network also includes three fully connected layers and two hidden layers, with the number of neurons in the hidden layers being 500, and the initial learning rate is 1×10^{-3} . Detailed parameters can be found in Table 1 below.

Table 1. Experiment parameters.

Parameter	Value
Number of hidden layer units	32
Number of critic network layers	3
Number of neurons	The sum of the state space sizes of all drones, 192 and 1
Number of Planner network	3
Number of neurons	State space size, 192, and m of UAV i
Input to the Executor network	Observation of UAV i
Output to the Executor network	Drone actions a_i
Activation function	ReLu
Initial learning rate of constraint network	5×10^{-4}
The number of neurons in the hidden layer of the actor network	100
The initial learning rate of the actor network	1×10^{-4}
The number of neurons in the hidden layer of the critic network	500
The initial learning rate of the critic network	1×10^{-3}
Episode	5000
Step	50
Discount factor	0.99
Batch size	64

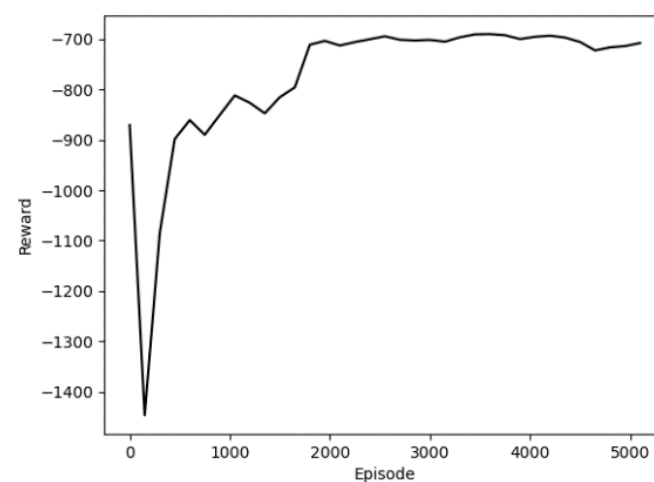
The system model considers a system environment composed of 80 user devices, 4 UAVs, and 2 edge servers. The flight range of UAVs is a rectangular area of $300 \text{ m} \times 300 \text{ m}$. The parameter settings are shown in Table 2.

Table 2. Simulation environment parameters.

Parameter	Value
Number of mobile devices	80
Number of UAVs	4
Number of edge servers	2
Task size	[1, 5] Mbits
Number of CPU cycles required to process 1-bit data	[100, 200] cycles/bit
Task achievement rate	1 task/s
Maximum flying altitude of unmanned aerial vehicle	100 m
Minimum flying altitude of unmanned aerial vehicle	50 m
Maximum horizontal flight distance of unmanned aerial vehicle	49 m
Vertical maximum flight distance of unmanned aerial vehicle	12 m
Unit path loss	−50 dB
Channel bandwidth from device to UAV	10 MHz
Channel bandwidth from UAVs to edge servers	0.5 MHz
Maximum transmission power of UAVs	5 W
Transmission power of mobile devices	0.1 W
Received power of UAVs	0.1 W
Computing resources of UAVs	3 GHz
Computing resources of edge servers	[6, 9] GHz
Effective Switched capacitor	10^{-28}
Gaussian noise at UAV	−100 dBm
Gaussian noise at the edge server	−100 dBm
Weight ω_1, ω_2	1

3.1. Training Performance of SC-MADDPG

In this section, we analyze the training performance of our proposed SC-MADDPG optimization method. In the multi-UAV-assisted MEC system, we propose optimal path planning for UAVs and computation task assignment. The training curve of the SC-MADDPG optimization method is shown in Figure 5. The reward value is unstable during the 0–1000 rounds of training, and it slowly rises during the 1000–2000 training rounds. After 2000 rounds of training, the reward value of the SC-MADDPG algorithm is in a relatively stable state. This demonstrates that the SC-MADDPG algorithm has a certain convergence.

**Figure 5.** Training Set Rewards.

The flight trajectory of the UAVs is shown in Figure 6. The black pentagram represents the mobile user devices, and the four different colored dots represent the four UAVs. As can be seen from the experimental results, the flight trajectories of each UAV are relatively concentrated, with the flight range not exceeding 1/4 or 1/2 of the entire specified range. This indicates that the four UAVs can cover all mobile user devices by communicating with each other, and only one UAV provides service to user devices without any overlapping coverage.

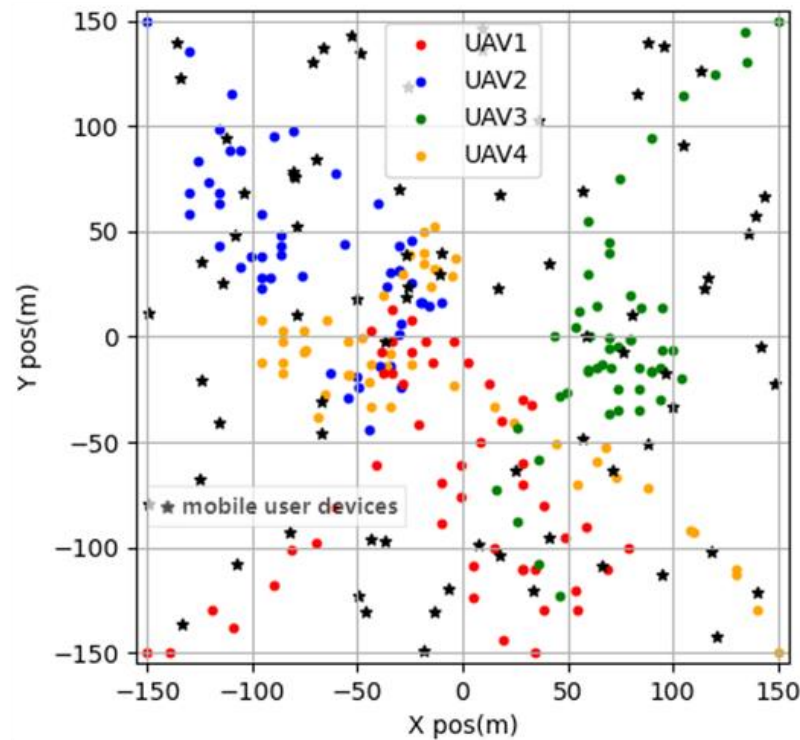


Figure 6. The trajectory of UAV flight.

3.2. The Impact of Communication Mechanisms on Latency and Energy Consumption

Figure 7 shows the average energy consumption performance of the SC-MADDPG algorithm and the MADDPG algorithm during the training process. The energy consumption of UAVs and edge servers for task offloading is proportional to the CPU frequency. As can be seen from the figure, the SC-MADDPG algorithm, which incorporates UAV communication mechanisms, has lower average energy consumption per task than the original MADDPG algorithm under different training rounds.

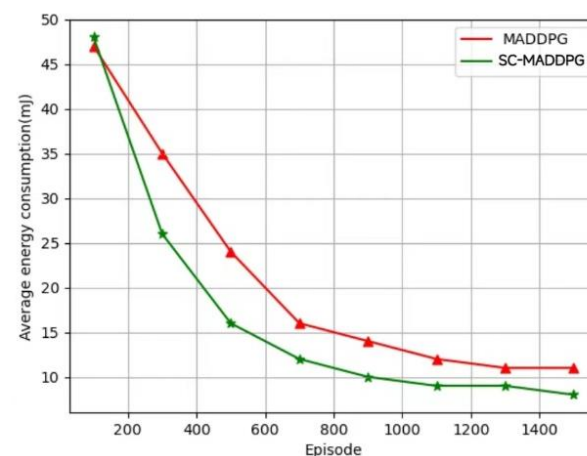


Figure 7. The average energy consumption of MADDPG and SC-MADDPG algorithms.

Figure 8 shows that in the case of different numbers of UAVs, the average task delay of the SC-MADDPG algorithm proposed in this chapter is less than that of the original MADDPG algorithm, indicating that UAV communication has a significant effect on reducing task processing delay. In addition, as the number of UAVs increases, the task processing delay first decreases and then increases. The main reason is that the communication between UAVs can cover mobile devices more evenly, and tasks generated by mobile devices can be promptly processed by UAVs, thus reducing delay. However, when the number of UAVs exceeds a certain range, the increased number of communication instances between UAVs can delay task processing time, resulting in increased delay. As can be seen, when the number of UAVs is 4, the average delay of both algorithms is minimized.

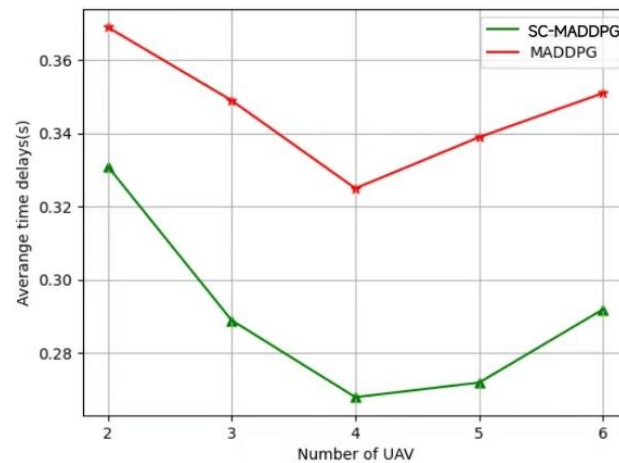


Figure 8. Average latency under different numbers of UAVs.

Figure 9 shows the average delay performance of the SC-MADDPG algorithm and the MADDPG algorithm during the training process. Incorporating UAV communication mechanisms in the MADDPG algorithm allows for more even coverage of tasks generated by devices, thereby reducing the transmission delay. As can be seen from the figure, as the model converges to the optimal policy, the average delay of the SC-MADDPG algorithm decreases to 0.27, saving time cost and improving the QoS of user devices compared to the MADDPG algorithm.

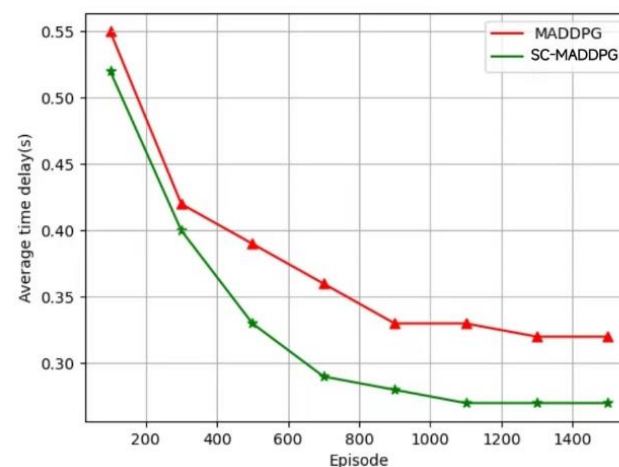


Figure 9. Average latency of different algorithms.

3.3. Performance Comparison of Optimization Algorithms after Integrating Security Layers

In this paper, the SC-MADDPG algorithm proposed (i.e., the MADDPG algorithm that integrates a safety layer based on the incorporation of communication mechanisms)

is compared with the following two joint optimization algorithms for task offloading and UAV trajectory:

- (1) MADDPG algorithm (with UAV communication mechanism): execute joint UAV actions directly obtained from the Actor network without considering safety constraints.
- (2) Hard-MADDPG algorithm (with UAV communication mechanism): consider safety constraints and require the output safe joint action to satisfy all constraints.

The cumulative number of collisions in the test phase after convergence of the three algorithms is shown in Figure 10. From the experimental results, it can be seen that the cumulative number of collisions between UAVs increases gradually with the increase of training rounds. The MADDPG algorithm without safety constraints has the highest number of UAV collisions, followed by the Hard-MADDPG algorithm, and the SC-MADDPG algorithm has the least. Therefore, the SC-MADDPG algorithm with a safety layer has a better effect in avoiding UAV collisions.

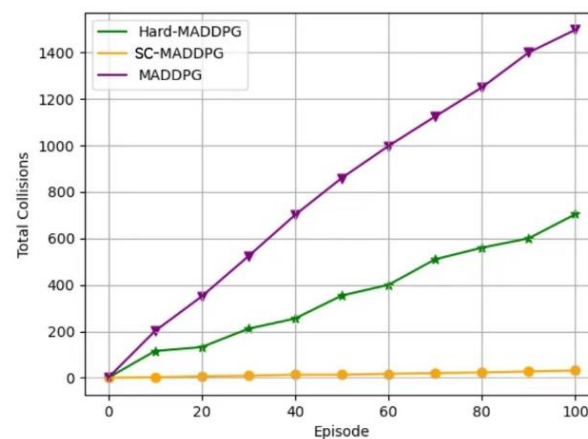


Figure 10. Number of UAV collisions under different algorithms.

Figure 11 shows the average energy consumption of the SC-MADDPG, Hard-MADDPG, and MADDPG algorithms. The experiment iterated 1800 rounds. Experimental results show that the energy consumed by the SC-MADDPG algorithm is less than the energy consumed by the Hard-MADDPG and MADDPG algorithms, as after a UAV collision, the task on the UAV needs to be re-uploaded from the mobile user device to another UAV for task offloading, which increases the transmission energy consumption. The SC-MADDPG algorithm reduces the probability of UAV collisions, resulting in lower energy consumption.

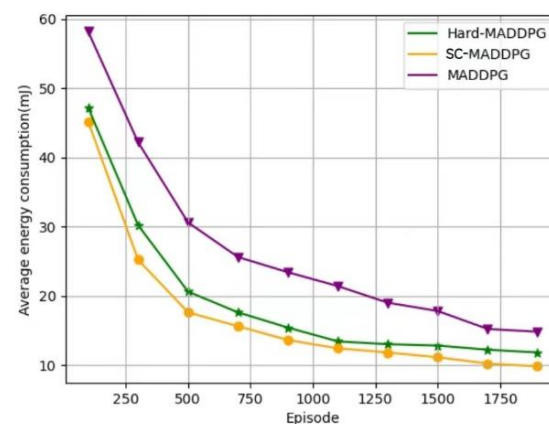


Figure 11. Average energy consumption of different algorithms.

In this section, the average delay of the SC-MADDPG, hard-MADDPG, and MADDPG algorithms are compared in different task data sizes. The experimental results show that

the MADDPG algorithm has the worst performance and the highest average delay. The average task processing delay of the SC-MADDPG algorithm is less than the other two algorithms. The SC-MADDPG algorithm provides safe joint actions, prevents UAVs from colliding, and avoids the delay of task retransmission, resulting in the lowest delay and optimal performance. The specific effect is shown in Figure 12.

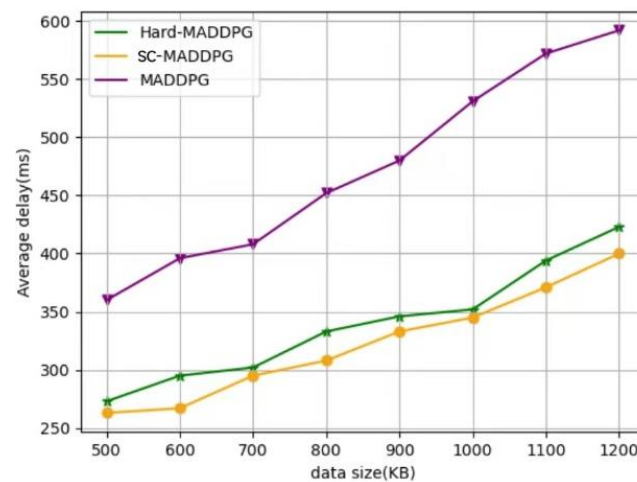


Figure 12. Average latency of algorithms with different data sizes.

4. Discussion

For the MEC scenario of multi-UAV and multi-edge cloud collaboration, this paper jointly designs the optimization problem of trajectory and computation task allocation to obtain the minimum total execution delay and energy consumption. We improve the MADDPG algorithm by adding communication mechanisms between UAVs and incorporating a safety layer to constrain UAV actions. Experimental results show that the proposed SC-MADDPG algorithm can achieve UAV path planning and provide optimal task offloading solutions. Compared with other optimization methods, it reduces the number of UAV collisions in the retraining process and reduces the overall energy consumption and delay.

5. Conclusions

The current MEC system does not consider the mobility of user devices. In practical application scenarios, the positions of user devices are mostly unfixed. Therefore, in the future, we will design joint optimization algorithms for task offloading considering user device mobility and UAV trajectory.

Furthermore, when the MEC system assisted by UAVs contains a large number of UAVs, existing algorithms struggle to obtain good path planning and task offloading strategies. This is because these algorithms must deal with a huge amount of input, making it difficult for the algorithms to converge. Next, we will further explore path planning and task offloading solutions in UAV-assisted large-scale MEC system environments.

Lastly, the latest research on UAV flights in MEC environments reveals the critical impact of factors such as wind speed, precipitation, visibility, terrain, and signal interference on UAV flights. These influencing factors are of great significance in practical scenarios. In our future work, we will delve deeper into more realistic simulation scenarios, including UAV flight models, task offloading modeling, and MDs' motion trajectories.

Author Contributions: Conceptualization, Y.D.; methodology, Y.D.; software, J.F.; validation, J.F. and Z.G.; formal analysis, Z.G.; investigation, J.F. and Z.G.; resources, Y.D.; data curation, J.F.; writing—original draft preparation, J.F.; writing—review and editing, Y.D. and L.Y.; visualization, J.F.; supervision, L.Y.; project administration, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

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

References

1. Liao, H.; Li, X.; Guo, D.; Kang, W.; Li, J. Dependency-Aware Application Assigning and Scheduling in Edge Computing. *IEEE Internet Things J.* **2022**, *9*, 4451–4463. [\[CrossRef\]](#)
2. Xu, Y.; Zhang, T.; Liu, Y.; Yang, D.; Xiao, L.; Tao, M. UAV-Assisted MEC Networks With Aerial and Ground Cooperation. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 7712–7727. [\[CrossRef\]](#)
3. Liu, Q.; Shi, L.; Sun, L.; Li, J.; Ding, M.; Shu, F.S. Path Planning for UAV-Mounted Mobile Edge Computing With Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5723–5728. [\[CrossRef\]](#)
4. Wang, L.; Wang, K.; Pan, C.; Xu, W.; Aslam, N.; Nallanathan, A. Deep Reinforcement Learning Based Dynamic Trajectory Control for UAV-Assisted Mobile Edge Computing. *IEEE Trans. Mob. Comput.* **2022**, *21*, 3536–3550. [\[CrossRef\]](#)
5. Zhao, N.; Liu, Z.; Cheng, Y. Multi-agent deep reinforcement learning for trajectory design and power allocation in multi-UAV networks. *IEEE Access* **2020**, *8*, 139670–139679. [\[CrossRef\]](#)
6. Yang, G.; Dai, R.; Liang, Y. Energy-efficient UAV backscatter communication with joint trajectory design and resource optimization. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 926–941. [\[CrossRef\]](#)
7. Li, M.; Cheng, N.; Gao, J.; Wang, Y.; Zhao, L.; Shen, X. Energyefficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3424–3438. [\[CrossRef\]](#)
8. Wang, Y.; Ru, Z.; Wang, K.; Huang, P. Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing. *IEEE Trans. Cybern.* **2020**, *50*, 3984–3997. [\[CrossRef\]](#) [\[PubMed\]](#)
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.* **2021**, *69*, 573–588. [\[CrossRef\]](#)
10. Yu, Z.; Gong, Y.; Gong, S.; Guo, Y. Joint task offloading and resource allocation in UAV-enabled mobile edge computing. *IEEE Internet Things J.* **2020**, *7*, 3147–3159. [\[CrossRef\]](#)
11. Liu, Y.; Xie, S.; Zhang, Y. Cooperative offloading and resource management for UAV-enabled mobile edge computing in power IoT system. *IEEE Trans. Veh. Technol.* **2020**, *69*, 12229–12239. [\[CrossRef\]](#)
12. Ji, J.; Zhu, K.; Yi, C.; Niyato, D. Energy consumption minimization in UAV-assisted mobile-edge computing systems: Joint resource allocation and trajectory design. *IEEE Internet Things J.* **2021**, *8*, 8570–8584. [\[CrossRef\]](#)
13. Zhang, J.; Zhou, L.; Tang, Q.; Ngai, E.C.-H.; Hu, X.; Zhao, H.; Wei, J. Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing. *IEEE Internet Things J.* **2019**, *6*, 3688–3699. [\[CrossRef\]](#)
14. Sun, C.; Ni, W.; Wang, X. Joint Computation offloading and trajectory planning for UAV-assisted edge computing. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 5343–5358. [\[CrossRef\]](#)
15. Zhan, C.; Hu, H.; Liu, Z.; Wang, Z.; Mao, S. Multi-UAV-enabled mobile-edge computing for time-constrained IoT applications. *IEEE Internet Things J.* **2021**, *8*, 15553–15567. [\[CrossRef\]](#)
16. Ding, R.; Xu, Y.; Gao, F.; Shen, X. Trajectory Design and Access Control for Air-Ground Coordinated Communications System with Multi-Agent Deep Reinforcement Learning. *IEEE Internet Things J.* **2021**, *9*, 5785–5798. [\[CrossRef\]](#)
17. Koushik, A.M.; Hu, F. Deep Q-Learning-Based Node Positioning for Throughput-Optimal Communications in Dynamic UAV Swarm Network. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 554–566. [\[CrossRef\]](#)
18. You, W.; Dong, C.; Wu, Q.; Qu, Y.; Wu, Y.; He, R. Joint task scheduling, resource allocation, and UAV trajectory under clustering for FANETs. *China Commun.* **2022**, *19*, 104–118. [\[CrossRef\]](#)
19. Zhan, C.; Zeng, Y. Aerial–Ground Cost Tradeoff for Multi-UAV-Enabled Data Collection in Wireless Sensor Networks. *IEEE Trans. Commun.* **2020**, *68*, 1937–1950. [\[CrossRef\]](#)
20. Wang, Z.; Liu, R.; Liu, Q.; Thompson, J.S.; Kadoch, M. Energy-Efficient Data Collection and Device Positioning in UAV-Assisted IoT. *IEEE Internet Things J.* **2020**, *7*, 1122–1139. [\[CrossRef\]](#)
21. Zhang, Q.; Miao, J.; Zhang, Z. Energy-Efficient Secure Video Streaming in UAV-Enabled Wireless Networks: A Safe-DQN Approach. *IEEE Trans. Green Commun. Netw.* **2021**, *5*, 1892–1905. [\[CrossRef\]](#)
22. Zhang, T.; Lei, J.; Liu, Y.; Feng, C.; Nallanathan, A. Trajectory Optimization for UAV Emergency Communication With Limited User Equipment Energy: A Safe-DQN Approach. *IEEE Trans. Green Commun. Netw.* **2021**, *5*, 1236–1247. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.