Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Dynamic deployment method based on double deep Q-network in UAV-assisted MEC systems

Suqin Zhang[1], Lin Zhang[2*], Fei Xu[3], Song Cheng[2], Weiya Su[3] and Sen Wang[2]

Abstract

The unmanned aerial vehicle (UAV) assisted mobile edge computing (MEC) system leverages the high maneuverability of UAVs to provide efficient computing services to terminals. A dynamic deployment algorithm based on double deep Q-networks (DDQN) is suggested to address issues with energy limitation and obstacle avoidance when providing edge services to terminals by UAV. First, the energy consumption of the UAV and the fairness of the terminal's geographic location are jointly optimized in the case of multiple obstacles and multiple terminals on the ground. And the UAV can avoid obstacles. Furthermore, a double deep Q-network was introduced to address the slow convergence and risk of falling into local optima during the optimization problem training process. Also included in the learning process was a pseudo count exploration strategy. Finally, the improved DDQN algorithm achieves faster convergence and a higher average system reward, according to experimental results. Regarding the fairness of geographic locations of terminals, the improved DDQN algorithm outperforms Q-learning, DQN, and DDQN algorithms by 50%, 20%, and 15.38%, respectively, and the stability of the improved algorithm is also validated.

Keywords Dynamic deployment, Unmanned aerial vehicle (UAV), Mobile edge computing (MEC), Double deep Q-network

Introduction

Edge computing offers computing, communication resources, network and storage at the edge of the network near the terminal by sinking computing resources to the edge end of the terminal. Terminals can reduce their own energy consumption and task processing delay by transferring their computing tasks to the edge [1]. Mobile edge computing (MEC) is a significant 5G technology that has undergone extensive research. The theory and application of the related research have seen a rapid growth since 2015 [2].

Despite the many benefits of edge computing, traditional base stations are constrained by their fixed locations and high implementation costs. In addition, infrastructure may occasionally be harmed by natural disasters. In the aforementioned scenario, the edge server is unable to completely serve the terminals. Due to its advantages in mobility, flexibility, and cost effectiveness, the unmanned aerial vehicle (UAV) has been widely used in both civil and military contexts for tasks like traffic management, disaster detection, emergency rescue, and target tracking [3]. Some progress has been made in the application research of UAV-assisted MEC taking into account the flexible mobility of UAVs [4, 5]. One of the first to suggest UAV-assisted MEC was Motlagh et al. [6]. By offloading to the edge, mobile terminals in MEC can significantly reduce energy

*Correspondence:
Lin Zhang
1550765068@qq.com
[1] School of Basic, Xi'an Technological University, Xi'An 710021, China
[2] School of Ordnance Science and Technology, Xi'an Technological University, 710021 Xi'An, China
[3] School of Computer Science and Engineering, Xi'an Technological University, 710021 Xi'An, China

consumption and latency. The fusion of an edge computing architecture and a UAV platform is referred to as UAV-assisted MEC. In addition to offloading to the edge computing server as a user node, the UAV can function as an air edge server for the terminal on the ground [7]. The UAV's position can be adjusted to provide better service in accordance with the needs of the terminals. This architecture successfully addresses the drawbacks of fixed base stations. Offloading the workload to the UAV-carried edge server also helps to reduce communication congestion caused by frequent communications between multiple terminals and the cloud.

The limited endurance and storage capacity of the UAV have become the key issues in its application, which promotes the efficient and dynamic deployment of UAV-assisted MEC. In MEC, the energy requirements for flight and hovering propulsion, as well as computing offloading, are the two main determinants of the UAV's energy consumption [8]. To fully utilize UAVs' potential in the MEC system, it is critical to conduct research on their trajectory design, hovering height, and dynamic deployment [9–11]. Under constraints like energy consumption and UAV mobility, the dynamic deployment of UAVs for MEC involves planning the deployment trajectory of UAVs to satisfy the terminals' unique service requirements.

In UAV-assisted MEC, traditional optimization algorithms (such as heuristic algorithms [12], clustering algorithms [13], convex optimization algorithms [14], etc.) always have disadvantages, such as large amount, slow convergence speed, and poor processing effect in dynamic environments. Compared with traditional optimization algorithms, Deep Reinforcement Learning (DRL), which combines deep learning models with reinforcement learning algorithms, has powerful autonomous learning and decision-making capabilities, and can adapt to unstable environments. And through the adaptive adjustment strategy of technology, such as strategy gradient, so as to realize the optimal control in the dynamic environment. The environment is complex and changeable, with high real-time requirements that cannot be met by traditional optimization algorithms. In order to implement the dynamic deployment of the UAV in MEC scenarios, this paper will use the DRL method.

In order to meet the needs of terminals for computing offloading as well as to achieve obstacle avoidance, this paper will propose a DRL algorithm to dynamically deploy the position of the edge server of the UAV in the case where the terminal position is moving and there are obstacles. The following are this paper's main contributions:

- First, it considers the time-varying channel parameters caused by the terminal's movement and the complex scene with obstacles on the ground, which is more realistic. We formulate the optimization problem and create a mathematical model for the optimization objective. In order to realize the training of DRL, the problem is transformed into a Markov decision model.
- Second, on the basis of the double deep Q-network algorithm, a ε-pseudo count based exploration strategy is proposed, which is a hybrid of the ε greedy and pseudo count exploration strategies. The goal is to encourage the agent to investigate more states and actions, maximizing fairness. And the jain fairness factor $f_n(t) \in (0, 1)$ is used to measure the fairness of terminals being provided with computation offloading services by UAVs. When $f_n(t)$ is 1, the fairness is the greatest.
- Third, the simulation experiment verifies the effectiveness of the proposed algorithm. The simulation results demonstrate that the improved algorithm is not only superior to the traditional DDQN, DQN algorithm and Q-learning algorithm in terms of convergence speed and average reward value; under the fairness factor, the improved DDQN algorithm also performs better than the traditional DDQN, DQN algorithm and Q-learning algorithm. In addition, its stability and universality are verified by changing the position, number and size of obstacles.

Related work

The UAV's energy use during flight is a key factor in determining the UAV's flight time, so the flight path of the UAV must be planned. The algorithms for managing the deployment of UAVs dynamically are primarily classified into two types, one is the traditional optimization algorithm including clustering algorithms, successive convex approximation algorithms, greedy algorithms, etc., and the other is the DRL algorithm.

In order to reach the purpose of minimizing the system's energy consumption, Huang et al. planned the trajectory of the UAV through three stages [12]. The three stages used the differential evolution algorithm with variable population size, the mean clustering algorithm and greedy algorithm respectively. The authors implemented 3D UAV localization using the K-means clustering algorithm and grouped terminals into adjacent cluster heads [13]. In order to extend the UAV flight time with charging station, Muhammad et al. proposed a three-stage joint routing and charging strategy, which uses optimization methods to design customer distribution area networks, charging station

distribution area networks and distribution routes [15]. In addition, the deployment of the UAV is generally optimized jointly with other indicators in the case of MEC with UAV assistance. Dai et al. proposed a generalized propulsion energy consumption model for rotary-wing UAVs, and jointly optimize user scheduling and UAVs trajectory to maximize UAVs energy efficiency [16]. An effective approach based on successive convex approximation for concurrently optimizing the UAV's trajectory and the bit allocation was suggested by Jeong et al. so as to reduce the overall energy consumption of the terminals while meeting the QoS criteria [14]. In order to break down the problem with joint optimization issue of terminal scheduling strategy, UAV's trajectory and transmit power into a series of feasible sub-problems to tackle, Qi et al. used successive convex approximation, penalty function, and Dinkelbach approach [17]. On the problem with joint optimization of UAV's trajectory and computing offloading, Hu et al. suggested a low-complexity offloading and trajectory scheduling method relying on Lyapunov optimization theory to minimize long-term energy efficiency [18]. Aiming at the communication security problem of UAV, Xu et al. suggested an approach based on penalized block coordinate descent by concurrently optimizing communication resources, UAV's trajectory, and computing resources to optimize the minimal safe computing capacity [19]. To increase the operating period of the UAV and the life of related network, Wang et al. decomposed region division and trajectory planning of UAV into two independent sub-problems, which were respectively modeled as semi-discrete optimal transportation problem and traveling salesman problem for solving [20].

In dynamic wireless environments, it might not be possible to make quick decisions using traditional optimization algorithms, like [12–20]. The DRL which combines deep neural networks and reinforcement learning, has become a popular research topic in light of the quick development of artificial intelligence. Additionally, articles demonstrate how powerful DRL can be for solving complex control problems [21, 22]. Therefore, DRL method is widely employed to settle the trajectory planning problem of UAV-assisted MEC. In the case of limited UAV's energy and QoS constraints of each terminal, the DRL method is utilized to improve the UAV's trajectory in order to maximize the system's long-term return. In order to maximize the system return and meet the constraint of QoS, Liu Qian et al. suggested QoS behavior selection strategy based on a double deep Q network algorithm to plan the UAV's flight path with limited energy [23]. Wang Liang et al. adopted multi-agent deep reinforcement learning algorithm to realize the dynamic deployment of multi-UAV assisted MEC in the scenario of multi-UAV, so as to meet the load balancing among UAV clusters [24]. Yin et al. used the multi-agent reinforcement learning approach to represent trajectory planning and resource allocation as a decentralized partially observable Markov decision process, with the goal of optimizing overall throughput and fair throughput [25]. UAVs can be utilized as temporary base stations to offer edge services to road vehicles which have heavy traffic. The reason is that the general mobile edge computing scheme with fixed base state cannot sufficiently manage the urgent communication needs in vehicle networks. A UAV-assisted vehicle communication network system was designed, and a traffic situational awareness-based algorithm for the best UAV flight trajectory was put forth to reduce the cost of UAVs [26]. Bor Yalinzi et al. considered both energy efficiency and coverage rate of terminals to optimize the UAV base station layout [27]. Hu et al. investigated task offloading and trajectory design in tandem to minimize the weighted sum of system energy consumption [28].

Furthermore, previous papers on UAV-assisted MEC rarely take collisions into account, which obviously does not conform to the display of real life. Therefore, Chang Huan et al. proposed to adopt a DRL method to realize the dynamic deployment of UAV edge computing platform in a complex environment with obstacles on the ground [29]. However, the position of the terminal will change with time in reality. In order to ensure that the UAV can offer computing services to the mobile terminals while avoiding obstacles, the UAV needs to adjust its trajectory in time. Moreover, in most existing studies, the deployment of UAVs is a one-time deployment. When the location of the terminal on the ground changes, the original deployment location may not provide the terminal with the optimal edge computing service. To sum up, in the case of obstacles on the ground, movement of terminals, and continuous deployment of UAV positions according to the needs of the terminals, it is still challenging to design the deployment trajectory of UAV, which is also the main motivation for the research work in this paper.

System model

For mobile edge computing scenarios, when a single UAV carrying an edge server provides edge computing services for multiple mobile terminals on the ground, it needs to meet obstacle avoidance, UAV's

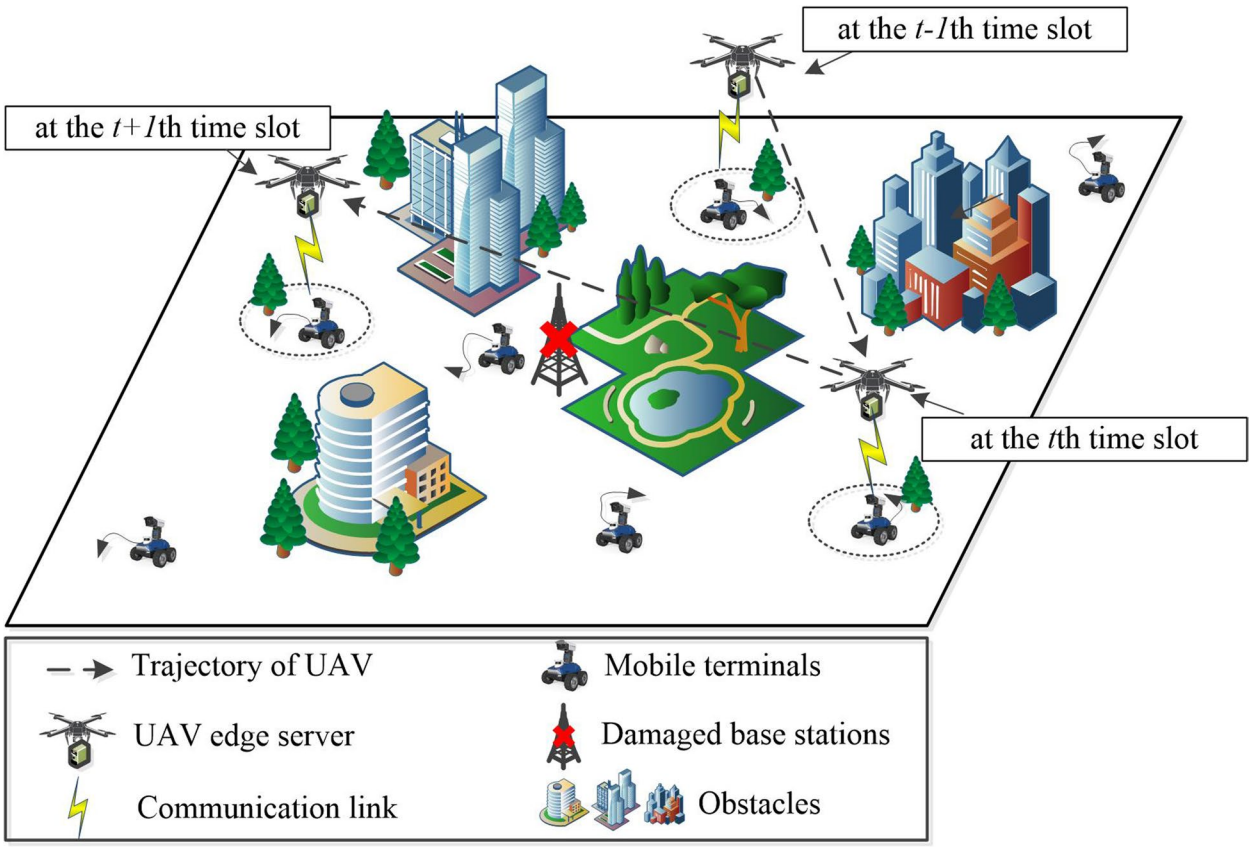

Fig. 1 System model of UAV dynamic deployment

own constraints, and fairness of geographic location of mobile terminals. System model is described in this section.

As shown in Fig. 1, there are a single UAV, N mobile terminals and K obstacles in a rectangular map with a length of ℓ and a width of w. The sets of terminals and obstacles are denoted as $n \in \mathrm{N} = \{1, 2, \cdots, N\}$ and $k \in \mathrm{K} = \{1, 2, \cdots, K\}$ respectively. The geometric center of the obstacle is used to represent the location information, and it is denoted as $b_k = (x_k, y_k, h), \forall k \in \mathrm{K}$, where h is the height of the center point of obstacle. This paper considers a discrete-time system and divides time into T slots, denoted as $t \in \mathrm{T} = \{0, 1, 2, ..., T\}$. The UAV flies over the target area to offer edge computing services for terminals at a fixed height H . The UAV's location is indicated as $u_{uav,t} = (x_t, y_t, H), \forall t \in \mathrm{T}$ at the tth time slot. The initial positions of N terminals are randomly distributed on the ground. Due to the mobility of terminals, the location coordinates of terminal n are $u_{n,t} = (x_{n,t}, y_{n,t}, 0), \forall n \in \mathrm{N}, t \in \mathrm{T}$ at the tth time slot. Assuming that terminals have a task of random size to offload in each time slot, we can derive the following constraints.

The main parameters and parameter meanings of this paper are displayed in the Table 1 below.

Movement model of terminals

Considering the mobility of terminals in the scenario, assuming that the position of the terminals does not change during the duration $\Delta_{t,t-1}$ between the tth and t-1th time slots. The Random Gauss-Markov Mobility (RGMM) model [30] is used to represent the mobility of the terminals. The RGMM is a model based on Gauss-Markov process, which is widely utilized in signal estimation and other fields. In a fixed interval, by changing its speed and direction to establish the correlation of speed and time of a moving terminal. The speed v_t and direction angle θ_t of the moving terminal on the ground at the tth time slot are defined as

$$v_t = \varphi v_{t-1} + (1-\varphi)\mu_v + \sigma_v(\sqrt{1-\varphi^2})\omega_{v_{t-1}}, \forall t \in \mathrm{T} \quad (1)$$

Table 1 Main parameters

Parameters	Implication
b_k	the position coordinates of obstacles
$u_{uav,t}$	the position coordinates of the UAV
$u_{n,t}$	the position coordinates of terminals
$d_{uav,k,t}$	the distance between obstacle k and the UAV
R_k	the radius of obstacle k
$O_{k,t}$	obstacle avoidance variable
$d_{uav,t}$	the flight distance of the UAV
v_{uav}	flight speed of the UAV
$T^{fly}_{uav,t}$	flight time of the UAV
$r_{uav,n,t}$	the uplink data transmission rate
B	channel bandwidth
P_{tr}	transmitted power
$T^{off}_{uav,n,t}$	the time of computation offloading
$D_{n,t}$	the task size needs to offload
$C_{n,t}$	CPU cycle
$E^{fly}_{uav,n,t}$	flight energy consumption of the UAV
P_f	flight power
$E^{off}_{uav,n,t}$	computation offloading energy consumption
f_{uav}	CPU frequency of the UAV
$E^{h}_{uav,n,t}$	hovering energy consumption of the UAV
$E^{total}_{uav,n,t}$	total energy consumption
W	total power of the UAV

$$\theta_t = \varphi\theta_{t-1} + (1-\varphi)\mu_\theta + \sigma_\theta(\sqrt{1-\varphi^2})\omega_{\theta_{t-1}}, \forall t \in \mathrm{T} \tag{2}$$

where $0 < \varphi < 1$ is the memory level. μ_v and μ_θ are the mean values of speed and angle, respectively. σ_v and σ_θ are the standard deviation of speed and angle, respectively. $\omega_{v_{t-1}}$ is an unrelated Gaussian process that is unrelated to v_{t-1} and has zero mean and unit variance. $\omega_{\theta_{t-1}}$ is an unrelated Gaussian process that is unrelated to θ_{t-1} and has zero mean and unit variance. Therefore, the position coordinate of moving terminal n is defined as

$$x_{n,t} = x_{n,t-1} + v_{t-1}cos\theta_{t-1} \cdot \Delta_{t,t-1}, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{3}$$

$$y_{n,t} = y_{n,t-1} + v_{t-1}sin\theta_{t-1} \cdot \Delta_{t,t-1}, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{4}$$

The location information of terminal n is represented by formulas (3), (4). Since the constant movement of terminals will bring about the disaster of action space dimension, the action space needs to be preprocessed. The map is divided into multiple subdomains, and the location coordinates will be updated only when the terminals move outside the subdomain. Otherwise, the location coordinates will not be updated. But the channel transmission parameters are still time-varying.

Obstacle avoidance model

As shown in Fig. 2, it shows the top view of the relationship between the flight path of the UAV and the position of the obstacle k. The outline of obstacles on the ground is simplified into a cylinder with radius R,

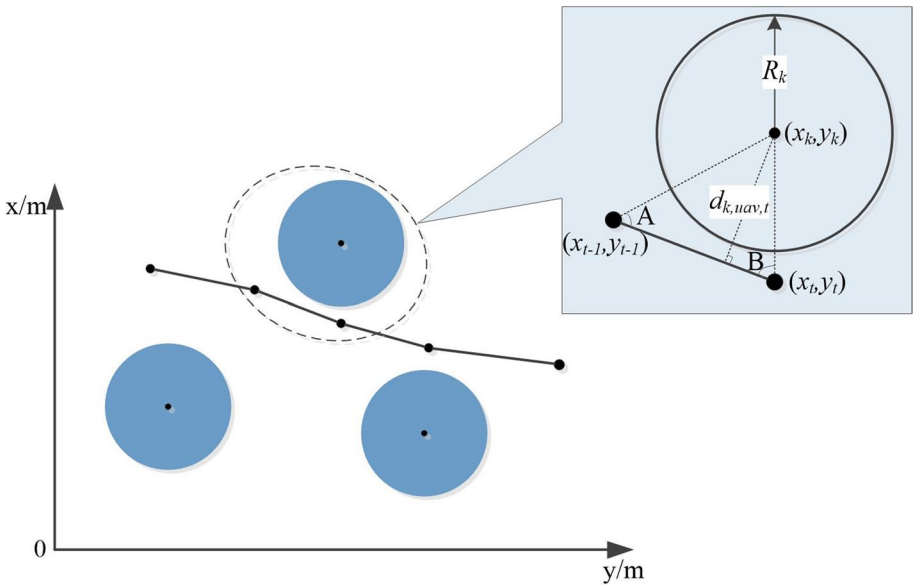

Fig. 2 Diagram of obstacle avoidance

and obstacles are represented by blue circles. Assuming that the flying height H of the UAV is lower than the height h of obstacles, the UAV needs to avoid obstacles in flight to avoid collisions. At the tth time slot, the coordinate of the UAV is (x_t, y_t, H), and the coordinate is (x_{t-1}, y_{t-1}, H) at the *t-1*th time slot. The black dots indicate where the UAV stays at a certain moment, and the straight line between the two points indicates the flight trajectory of the UAV. Only relying on the distance between the current position of UAV and the center of the obstacle cannot accurately determine whether the UAV has collided with the obstacle. Therefore, it is judged whether the UAV collides with the obstacle during the flight according to the distance from the straight line where the flight track of the UAV is located to the center of the obstacle.

At the tth time slot, the equation of the horizontal flight path of the UAV is expressed as

$$(y_t - y_{t-1}) \cdot x + (x_t - x_{t-1}) \cdot y + x_{t-1}y_t - x_t y_{t-1} = 0, \quad \forall t \in \mathrm{T} \tag{5}$$

The distance between the straight line of the flight path of the UAV and the center of obstacle k is

$$d_{uav,k,t} = \frac{\left|(y_t - y_{t-1})x_k + (x_t - x_{t-1})y_k + x_{t-1}y_t - x_t y_{t-1}\right|}{\sqrt{(y_t - y_{t-1})^2 + (x_t - x_{t-1})^2}}, \quad \forall k \in \mathrm{K}, t \in \mathrm{T} \tag{6}$$

If the distance between the straight line where the flight path of the UAV is located and the center of the obstacle k is greater than the radius R_k of the obstacle, the UAV will not hit the obstacle; If the radius R_k is smaller than the distance between the UAV and the obstacle k, it is necessary to judge *cos A* and *cos B*. if $cosA > 0$ and $cosB > 0$ then collide. *cos A* and *cos B* are expressed as

$$cosA = \frac{(x_t - x_{t-1})^2 + (x_{t-1}y_t - x_t y_{t-1})^2 - (y_t - y_{t-1})^2}{2(x_t - x_{t-1})(x_{t-1}y_t - x_t y_{t-1})}, \quad \forall t \in \mathrm{T} \tag{7}$$

$$cosB = \frac{(x_{t-1}y_t - x_t y_{t-1})^2 + (y_t - y_{t-1})^2 - (x_t - x_{t-1})^2}{2(y_t - y_{t-1})(x_{t-1}y_t - x_t y_{t-1})}, \quad \forall t \in \mathrm{T} \tag{8}$$

Define the obstacle avoidance variable as $O_{k,t} = \{0, 1\}$. $O_{k,t} = 0, \forall k \in \mathrm{K}, t \in \mathrm{T}$ indicates that the UAV collides with obstacle k at the tth time slot, and $O_{k,t} = 1, \forall k \in \mathrm{K}, t \in \mathrm{T}$ indicates that UAV successfully avoids obstacle k, specifically it is expressed as

$$O_{k,t} = \begin{cases} 0, & d_{uav,k,t} < R_k \cap (cosA > 0 \cap cosB > 0) \\ 1, & else \end{cases} \tag{9}$$

Energy consumption model

This paper assumes that the entire deployment process is divided into T time slots of unequal length, and each time slot t is divided into three parts. The first part is the scheduling decision. The agent obtains the next deployment position by analyzing the current position of terminals and obstacles and other information; The second part is the flight time, that is, the time required for the UAV to fly to the new deployment location; The third part is the time of computation offloading. After the UAV arrives at the new deployment location, it provides edge services for terminals. The agent interacts with the environment and makes decisions quickly, so the decision time is negligible compared to the flight time and computation offload time, the tth time slot division is shown in Fig. 3 below.

The flying distance of the UAV is $d_{uav,t}$ at the tth time slot, assume that the UAV is flying at a fixed speed v_{uav}

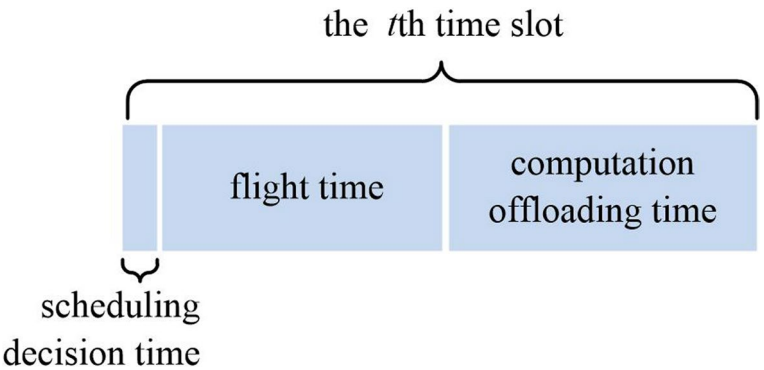

Fig. 3 Diagram of the *t*th time slot division

and the speed is not greater than the maximum flight speed v_{max} of the UAV. Then the flight time of the UAV at the tth time slot is defined as

$$T_{uav,t}^{fly} = \frac{d_{uav,t}}{v_{uav}}, \forall t \in \mathrm{T} \tag{10}$$

where $d_{uav,t}$ represents the horizontal distance of the UAV flying at the tth time slot, it is expressed as

$$d_{uav,t} = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}, \forall t \in \mathrm{T} \tag{11}$$

In the MEC scenario, it is assumed that the flying height of the UAV is always lower than the height of the obstacle, and the UAV needs to avoid the obstacle before it can provide services for the terminals on the ground. Therefore, the wireless communication link between the UAV and the terminals adopts the line-of-sight link communication. At the tth time slot, the uplink data transmission rate is expressed as

$$r_{uav,n,t} = Blog_2(1 + \frac{\rho P_{tr}}{H^2 + d_{uav,n,t}^2}), \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{12}$$

where $d_{uav,n,t}$ is the horizontal distance between the UAV and the terminal n at the tth time slot, and it is expressed as

$$d_{uav,n,t} = \sqrt{(x_t - x_{n,t})^2 + (y_t - y_{n,t})^2}, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{13}$$

B is the bandwidth of the channel and P_{tr} is the transmission power, $\rho = g_0 G_0/\sigma^2$, $G_0 \approx 2.2846$, where g_0 represents the channel power gain at the reference distance of 1m, and σ^2 represents the noise power. At the tth time slot, the time for the UAV to compute unloading for terminal n is defined as

$$T_{uav,n,t}^{off} = \frac{D_{n,t}}{r_{uav,n,t}} + \frac{F_{n,t}}{f_{uav}}, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{14}$$

The time of computation offloading is divided into two parts, one is the task transmission, and the other is the task calculation. Where $D_{n,t}/r_{uav,n,t}$ is the time required for terminal n to transmit mission data to the UAV at the tth time slot, indicates the amount of tasks that terminal n needs to offload at the tth time slot; $D_{n,t}$ indicates the time required for the UAV to calculate tasks offloaded from terminal n at the tth time slot, among them, $F_{n,t} = D_{n,t}C_{n,t}$ is defined as the CPU cycle required to calculate all tasks, $C_{n,t}$ is the CPU cycle required to calculate each bit of data for the UAV, and f_{uav} is the CPU frequency assigned to tasks by the edge server on the UAV.

Then the flight energy consumption of UAV at the tth time slot is defined as

$$E_{uav,n,t}^{fly} = P_f T_{uav,t}^{fly}, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{15}$$

where P_f represents the flight power of the UAV. At the tth time slot, the energy consumption required by the UAV to compute offloading for terminal n is defined as

$$E_{uav,n,t}^{off} = \frac{P_{tr} D_{n,t}}{r_{uav,n,t}} + \kappa D_{n,t} C_{n,t} f_{uav}^2, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{16}$$

The energy consumption of computation offloading is divided into two parts, one is transmission energy consumption, and the other is computing energy consumption. $P_{tr}D_{n,t}/r_{uav,n,t}$ indicates the transmission energy consumption of terminal n offloading the task to the UAV at the tth time slot, where P_{tr} represents the transmission power; $\kappa D_{n,t}C_{n,t}f_{uav}^2$ indicates the computational energy consumption of the UAV at the tth time slot.$\kappa = 10^{-26}$ is a hardware-related constant.

In order to ensure the stability and reliability of the communication link, the UAV needs to hover while providing computation offloading services for terminals. Therefore, the hovering energy consumption of the UAV serving terminal n at the tth time slot is defined as

$$E_{uav,n,t}^{h} = P_h T_{uav,n,t}^{off}, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{17}$$

where P_h is the hovering power of the UAV. Then the total energy consumption of UAV at the tth time slot is

$$E_{uav,n,t}^{total} = E_{uav,n,t}^{fly} + E_{uav,n,t}^{off} + E_{uav,n,t}^{h}, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{18}$$

The UAV needs to serve terminals within the energy consumption range, $\sum_{t=0}^{T} E_{uav,n,t}^{total} \le E_{uav}^{\max}$. The total power of the UAV is W. When the energy of the UAV is exhausted, it will stop serving or return to charge.

Problem formulation

In the case of obstacles on the ground, by dynamically deploying the position of the UAV to meet the computation offloading requirements of the mobile terminals and realize the obstacle avoidance of the UAV, it will lead to some unfair problems. For example, the UAV may only serve nearby terminals in order to save energy and avoid obstacles, and do not consider providing edge computing services for distant terminals. As a result, some terminals may be ignored and have not been served by the UAV. We use jain fairness factor to solve this problem. First, the offloading variable is defined as

$$Z_{n,t} = \{0,1\}, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{19}$$

where $Z_{n,t} = 1, \forall n \in \mathrm{N}, t \in \mathrm{T}$ means that the terminal n is offloaded at the tth time slot, and $Z_{n,t} = 0, \forall n \in \mathrm{N}, t \in \mathrm{T}$ means that the terminal n is not offloaded. Moreover $\sum_{n=1}^{N} Z_{n,t} = 1, \forall n \in \mathrm{N}, t \in \mathrm{T}$, it means that the UAV can only serve one terminal at each time slot. The fairness factor $f_n(t)$ is used to represent the fairness of the geographical location of the terminals (the fairness of the UAV's service between the terminals)

$$f_n(t) = (\sum_{n=1}^{N} \sum_{t=1}^{t} Z_{n,t})^2 / N \sum_{n=1}^{N} (\sum_{t=1}^{t} Z_{n,t})^2 \tag{20}$$

If all terminals are served by UAV for similar times from the initial time slot t=0 to T, then the value of $f_n(t)$ is closer to 1, otherwise it is closer to 0. The optimization problem is formulated as follows:

$$\mathrm{P1}: max \sum_{t=0}^{T} \{O_{k,t} + \frac{f_n(t)}{\sum_{n=1}^{N} E_{uav,n,t}^{total}}\} \tag{21a}$$

$$\sum_{t=0}^{T} E_{uav,n,t}^{total} \leq E, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{21b}$$

$$v_{uav} \leq v_{max} \tag{21c}$$

$$0 \leq x_t \leq \ell_{max}, 0 \leq y_t \leq w_{max}, \forall t \in \mathrm{T} \tag{21d}$$

$$f_{uav} \leq f_{uav}^{max} \tag{21e}$$

$$\sum_{n=1}^{N} Z_{n,t} = 1, \forall n \in \mathrm{N}, t \in \mathrm{T} \tag{21f}$$

The optimization objective is to minimize the UAV's energy consumption while maximizing the obstacle avoidance variable between the UAV and obstacles, and to realize the fairness of the terminals being served by the UAV. (21b) means that from t=1 to T, the UAV's total energy consumption cannot exceed the maximum energy consumption; (21c) means the UAV cannot fly faster than its maximum speed; (21d) means that the UAV's flight range is not allowed to go beyond the fixed area; (21e) means that the CPU frequency that the MEC server installed on the UAV assigns to the task cannot exceed the maximum frequency; (21f) means that the UAV can only serve one terminal during time slot t.

MDP

The DRL takes advantage of the agent's interaction with the environment and allows the agent to learn the optimal action through rewards. In accordance with the system model in the above section, the environment and action of the system are judged to have Markov properties, so the paper utilizes Markov decision process (MDP) to describe the system model. This section introduces the MDP and the DRL model for the above optimization problem. And a DRL algorithm based on an action selection strategy of ε -pseudo count is used to realize the dynamic deployment of UAV edge server.

The MDP mainly consists of state, action, reward and discount factor γ, which is used to describe the interaction process between agent and environment, and the learning process of agent in DRL. Considering the system model in the aforementioned section, the problem we need to solve conforms to the MDP, and the following is to establish the Markov decision model.

1) State Space: first, we describe the state space for each episode as $S = \{s_1, s_2, \cdots, s_i, \cdots, s_t\}, \forall t \in \mathrm{T}$, where s_i is the state at the ith time slot

$$s_i = \{u_{n,i}, u_{uav,i}, b_k, D_{n,i}, O_{k,i}, r_{uav,n,i}, \Omega_i\} \tag{22}$$

where $u_{n,i}$ is the terminals' location information at the ith time slot; $u_{uav,i}$ is the position information of the UAV at the ith time slot; b_k is the location information of obstacles on the ground; $D_{n,i}$ is the amount of tasks that the terminal n needs to offload at the ith time slot; $O_{k,i}$ is the obstacle avoidance variable at the ith time slot; $r_{uav,n,i}$ is the channel transmission rate between the UAV and terminal n at the ith time slot; $\Omega_i = \{u_{n,i} Z_{n,i}\}$ represents the terminal n served by the UAV at the i$text$th time slot.

2) Action Space: the agent selects the action with the greatest reward based on the current state, i.e. the next position of the UAV. Action space is defined as

$$a_i = \{u_{uav,i+1}\} \tag{23}$$

3) Reward Function: when the position of the UAV is deployed once, the reward can be calculated, which is the feedback of the whole system model to the DRL model. Specifically, reward refers to the ratio of fairness factor of edge computing service to total energy consumption, obstacle avoidance variable and additional reward brought by pseudo count exploration for terminals after dynamic deployment of the UAV. Therefore, the reward obtained from the current state and action is defined as

$$R_{i+1} = \omega_1 O_{k,t} + \omega_2 \frac{f_n(t)}{\sum_{n=1}^{N} E_{uav,n,t}^{total}} + \psi(N(s)) \tag{24}$$

among them, ω_1 and ω_2 are the coefficients that adjust the reward ratio. Respectively, $\psi(N(s))$ represents the additional rewards explored by pseudo count strategy.

Deployment algorithm based on DDQN with ε -pseudo count selection strategy of action

The DDQN algorithm in the DRL algorithm is used to eliminate the overestimation problem of the DQN by decoupling the selection of actions in the target Q value and the calculation of the target Q value. The algorithm structure of DDQN and DQN is the similar, but the way of updating in the Q network is different. The DDQN chooses the action with the highest Q value in accordance with the parameters of the main network, whereas the DQN chooses the action in accordance with the parameters of the target network. This solves the problem of overestimation to a certain extent, making the Q value closer to the true value.

The DDQN consists of two neural networks: the target network and the main network. Specifically, Q^{main} is the output of the main network, which is a value function used to evaluate the current state-action. Q^{target} is the output of the target network.

$$Q^{main} = Q(s_i, a_i; \theta_i) \tag{25}$$

$$Q^{target} = Q(s_{i+1}, \underset{a}{argmax}\, Q(s_{i+1}, a_i; \theta_i); \theta_i^-) \tag{26}$$

The special is that it is updated according to the action which maximizes the Q value in the main network. The target Q value Y^{DDQN} is defined as

$$Y^{DDQN} = R_{i+1} + \gamma Q^{target} \tag{27}$$

In order to avoid suboptimal results in DRL algorithms, common DDQN algorithms generally use ε -greedy strategy to explore and utilize. However, ε exploration may select the previously experienced states and actions, which cannot avoid the disadvantages of local optimality.

The core idea of pseudo count exploration is to calculate or estimate the frequency of each state-action by designing density model. And new state-actions are rewarded with higher bonuses, encouraging the agent to try more state-actions. The DRL algorithm is utilized to train the UAV dynamic deployment to meet the fairness of terminals being served by the UAV in this paper. Considering the principle of pseudo count exploration mode, it can be found that the DRL algorithm based on ε-pseudo count exploration mode has better performance to solve this problem.

The probability density of occurrence of s_i is defined as $\rho(s_i) = \rho(s_i|s_{1:t})$, and the probability density is $\rho'(s_i) = \rho(s_i|s_{1:t}s_i)$ when s_i is observed in the next time. In order to better understand the density model, two concepts are introduced, namely the pseudo count function $\hat{N}(s,a)$ and the pseudo count total $\hat{t}$. Then the probability density of the state s_i appearing and the probability density of s_i observed in the next time are $\rho(s_i) = \hat{N}(s_i)/\hat{t}$ and $\rho'(s_i) = (\hat{N}(s_i)+1)/(\hat{t}+1)$, respectively. The relationship between $\rho(s_i)$ and $\rho'(s_i)$ is the learning-positive density model. Then we can get the expression of pseudo count

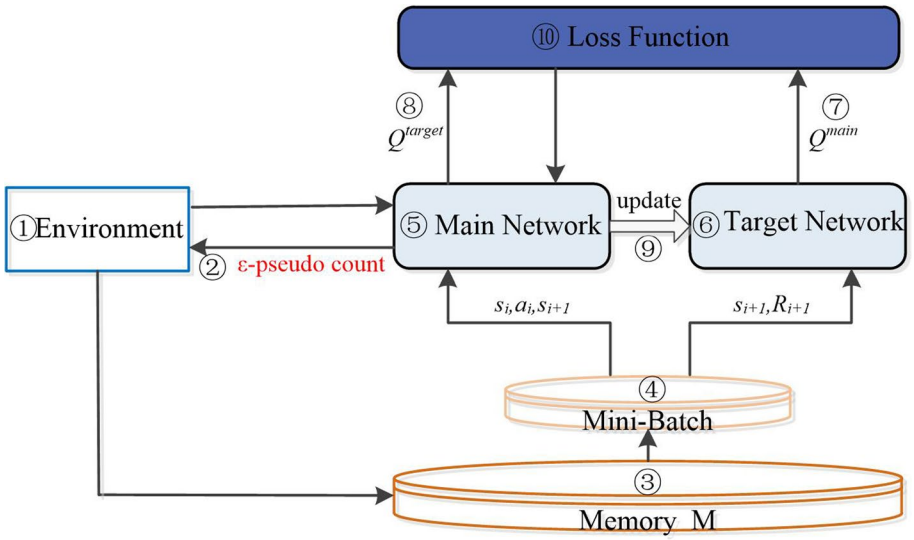

Fig. 4 Schematic diagram of deployment algorithm based on DDQN with ε-pseudo count

$$\hat{N}(s_i) = \hat{t}\rho(s_i) = \frac{\rho(s_i)(1-\rho'(s_i))}{\rho'(s_i)-\rho(s_i)} \tag{28}$$

In addition, the reward defined by pseudo count exploration is $\psi(N(s)) = \hat{N}(s)^{-1/2}$. The pseudocode of ε -pseudo count selection strategy of action is shown in the Algorithm 1.

Input: random numbers δ, $\delta \in (0,1)$
Output: action selection strategy
if $\delta < \varepsilon$ **then**
  $a \leftarrow \underset{a}{argmax}\, Q(s_{i+1}, a_i; \theta_i)$
else
  for i=0,1,2,... **do**
    take action a_i from s_i;
    receive reward R_i and transit to s_{i+1}, and fit new density model $\rho'(s_i)$ to s_i;
    use $\rho(s_i)$ and $\rho'(s_i)$ to estimate $\hat{\mathrm{N}}(s_i)$;
    set $R_{i+1} = \omega_1 O_{k,t} - \omega_2 f_n(t)/\sum_{n=1}^{N} E_{uav,n,t}^{total} + \psi(N(s))$;
  end for
end if

Algorithm 1 The action selection strategy based on ε -pseudo countThe schematic diagram of the DDQN algorithm based on the ε-pseudo count exploration strategy is shown in Fig. 4.

It can be seen from Fig. 4 that ① Environment includes information such as the position of the UAV and the terminals, channel transmission rate, etc.;② Using ε -pseudo count exploration to explore the environment, and make the action a_i according to the current state s_i. Negative rewards will be given when the UAV collides with obstacles or consumes too much energy. The ε-pseudo count strategy based on the combination of the ε-greedy exploration strategy and the pseudo count exploration strategy can speed up the convergence rate and avoid local optimization, so as to achieve better exploration and utilization; Next, the reward R_{i+1} and the next state s_{i+1} are obtained, and then ③ the information obtained by the exploration is stored in the experience pool Memory M as a tuple $(s_i, a_i, R_{i+1}, s_{i+1})$. The experience pool has a fixed size. When the experience pool is full, the previously stored tuples will overflow; ④ During neural network training, a fixed number of small batches of experience are extracted from Memory M for training, which is called Mini-Batch; ⑤ and ⑥ are two neural networks of the DDQN algorithm, which are the main network and the target network, respectively. The purpose of neural network is to obtain the maximum target Q value. In order to ensure the stability of the algorithm, the main network copies the parameter θ to the target network after every fixed interval step C;⑦ and ⑧ are the Q values output by the two neural networks respectively, where ⑨ indicates that the Q value of ⑧ is updated according to the action of the maximum value of ⑦. ⑩Update network parameters by minimizing loss function. The pseudocode of deployment algorithm based on DDQN with ε-pseudo count selection strategy of action is shown in Algorithm 2.

Input: the number and initial position of terminals, the number and position of obstacles, the task size that the terminal n needs to offload, learning rate α, discount factor γ, Mini-Batch
Output: dynamic deployment decision
Initialization: batch gradient descent of mini-batch size m, size of experience pool Memory M, network structure and main network neural network parameters θ, set target network neural network parameters $\theta^- = \theta$, update parameter interval step number C;
for episode = 1,2,3,... **do**
  Initialize the first state S of the current state sequence and obtain its feature vector $\emptyset$(S); update the base exploration factor ε according to the number of iterations;
  for t = 1,2,3,... $(t < T)$ **do**
    choose action according to the action selection strategy based on ε-pseudo count (Algorithm 1);
    execute action a_i in state s_i, get new state s_{i+1}, and get instant reward R_{i+1};
    store the sample tuple $(s_i, a_i, R_{i+1}, s_{i+1})$ in the experience pool Memory M;
    if the total number of sample tuples in the experience pool is less than than M **then**
      continue to store the samples into the experience pool;
    else
      randomly sample m tuples from the experience pool (Mini-Batch);
      for Mini-Batch = 1,2,3,... **do**
        obtain Q^{main} and Q^{target} corresponding to each sample tuple, and calculate the current target Q value Y^{DDQN},$Y^{DDQN} = R_{i+1} + \gamma Q^{target}$;
        using the loss function, the neural network parameter θ is updated by small batch gradient descent, $Loss = E_{(s,a,r,s')\sim u(M)}[(Y^{DDQN} - Q^{main})^2]$;
        copy the target network neural network parameter $\theta^- = \theta$ with fixed update parameter interval step C;
      end for
    end if
  end for
end for

Algorithm 2 Deployment algorithm based on DDQN with ε-pseudo count selection strategy of actionSimulation experiment

In this part, we simulate the performance of the DDQN algorithm with the proposed ε-pseudo count selection strategy of action and analyze the results. In the PC environment of intel Core i7-1165G7, 2.8GHZ CPU, 16GB, we use python3.6 and tensorflow2.0 to simulate. In the algorithm, the main network and target network

Table 2 Parameters of simulation environment

Parameters	Value
Channel bandwidth B	1MHz
Flight speed of UAV	20m/s
Noise power σ^2	-140dB
Flying height of UAV H	50m
The power of the UAV W	200kJ
Amount of tasks to uninstall $D_{n,t}$	[1,10]Kb
CPU cycles of UAV $C_{n,t}$	1000
CPU frequency of UAV f_{uav}	2GHz
Flight energy consumption of UAV P_f	110W
Hovering energy consumption of UAV P_h	80W
Transmission power P_{tr}	0.1W

respectively adopt four fully connected hidden layers, and each layer has 50 neurons.

Create a simulation environment

On a map with a length of 1000m and a width of 600m, the UAV flies at a constant speed at a fixed height, and the obstacles in the training environment are simplified as cylinders, and the radius R of the cylinder, the height h and the coordinates of the obstacles are set; Secondly, set the starting positions of 30 terminals in the environment, and the starting positions of the terminals are randomly set; Finally, add the UAV to the environment, and set the relevant attribute parameters of the UAV. The specific parameter settings are shown in Table 2 below.

Algorithm hyperparameter settings

The hyperparameter setting of DRL algorithm is very important. Only by setting appropriate hyperparameters can the neural network play a better performance and the algorithm converge. A key hyperparameter in the algorithm is the learning rate α. When α is too low, the convergence complexity of the network increases, and it is easy to fall into a local minimum. When α is too high, the gradient oscillates around the minimum.

As shown in Fig. 5, it is the convergence of the proposed algorithm under different learning rates α. It can be seen that the algorithm does not converge when α is 0.1 and 0.01; The algorithm converges when α is 0.001 and 0.0001, but when α=0.001, the algorithm converges in 5000 episodes, and the average reward value of convergence is higher than when α=0.0001. So set the learning rate α equal to 0.001.

Figure 6 shows the convergence of the proposed algorithm under different discount factors γ. It is used to control how future rewards affect the current reward value. A higher value of γ indicates that the agent has more steps to consider but is also more difficult to train, while a lower value of it indicates that the agent pays more attention to immediate interests. Therefore, an appropriate discount factor must be determined. It can be seen from the Fig. 6 that when γ=0.99, the agent is overly focused on potential future rewards, so the convergence speed is slower and more difficult. When γ is equal to 0.95 and 0.9, although the average reward value of the final convergence is similar, it converges at about 7500 episodes when γ=0.95, and it converges at about 5000 episodes when γ=0.9, which is obviously faster than when γ=0.95.

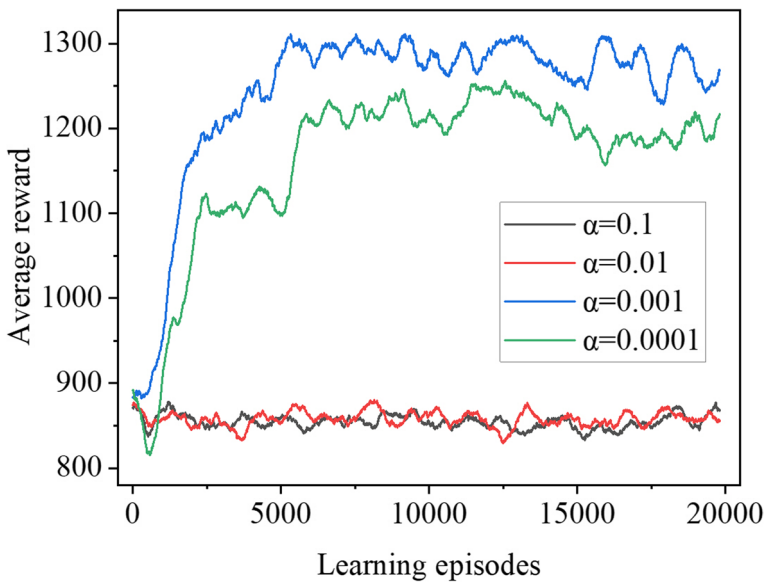

Fig. 5 The convergence of the proposed algorithm under various learning rates

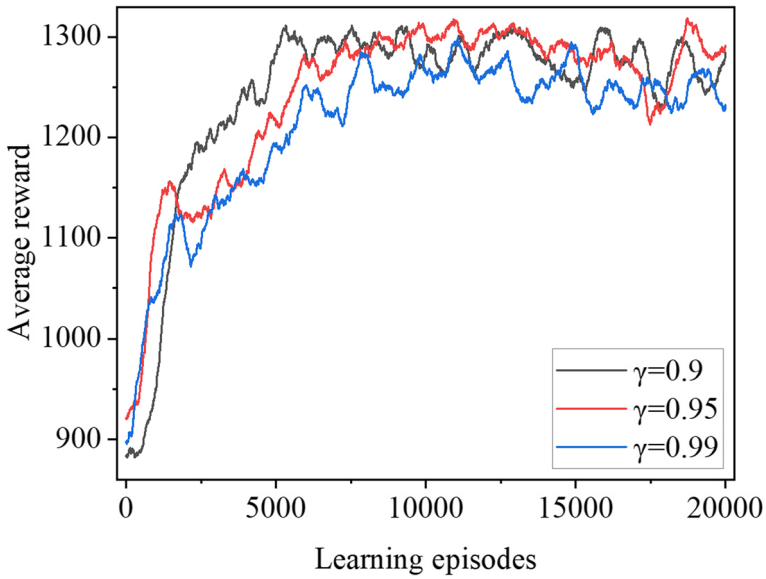

Fig. 6 The convergence of the proposed algorithm under various discount factors

Figure 7 shows the convergence of the proposed algorithm under different Mini-Batch sizes. Mini-Batch refers to the sample size that is subsampled from the experience pool and used for gradient calculation. The size setting should take into account both training stability and training speed.

It can be seen from Fig. 7 that when the Mini-Batch is 32 and 64, effective training samples cannot be extracted because the number of samples is too small, and the convergence trend is not obvious; When the Mini-Batch is 128, it converges at 5000 episodes and converges at about 1300. So set Mini-Batch equal to 128. Due to the mutual influence of the hyperparameters in DRL, after arranging and combining the parameters, a large number of comparative experiments are performed to determine the specific values. The remaining algorithm hyperparameter settings are shown in Table 3 below.

Result analysis

Figure 8 depicts the average rewards of the proposed algorithm, traditional DDQN, DQN algorithm and Q-learning algorithm. First, the algorithm proposed in this paper converges from about 6000 episodes and converges around 1250; The traditional DDQN algorithm converges after about 10,000 episodes, and the final

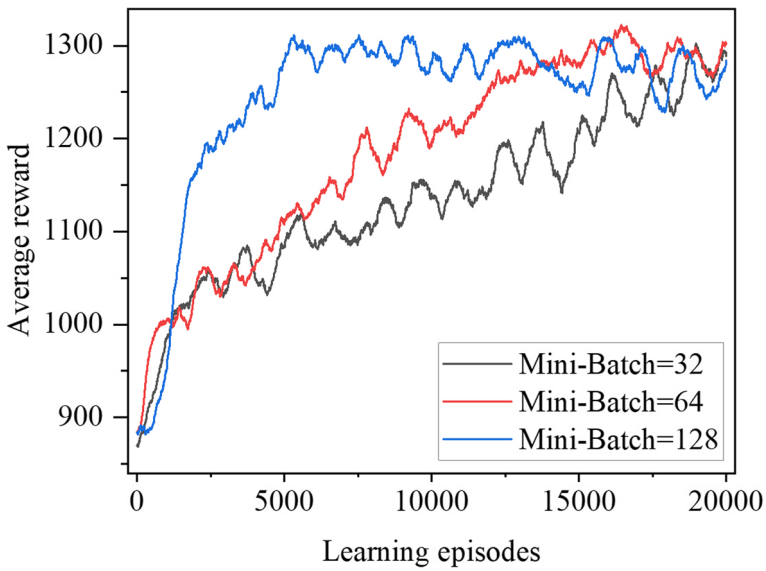

Fig. 7 The convergence of the proposed algorithm under various Mini-Batch sizes

Table 3 The hyperparameters of the algorithm

Parameters	Value
Number of hidden layer	4
Number of nodes	50
Activation function	Relu
Optimizer	Adam
Learning rate	0.001
Discount factor	0.9
Memory M size	20000
Mini-Batch size	128

reward value is 1200; The DQN algorithm converges to 850 around 8000 episodes. However, the Q-learning algorithm cannot converge because it cannot handle high-dimensional action spaces. It can be seen that the algorithm we proposed achieves the fastest convergence speed and the largest average reward value, indicating that the action selection strategy based on ε-pseudo count can more effectively realize the selection and utilization of actions by the agent.

Figure 9 depicts the fairness factor $f_n(t)$ of the proposed algorithm, traditional DDQN and DQN algorithms. It can be seen from the Fig. 9 that under the algorithm we proposed, it is around 0.9, which means that the UAV can satisfy the fairness of the geographical location of most terminals; Under the traditional DDQN algorithm, it is about 0.78, and under the DQN algorithm, it is about 0.75, which means that the UAV can satisfy the fairness of the geographic location of some terminals. The fairness factor $f_n(t)$ under the Q-learning algorithm is always the lowest and decreases with the increase of training times. It can be seen that our proposed algorithm achieves the greatest fairness of geographical location among all compared algorithms.

In order to verify the stability and effectiveness of the proposed algorithm, the fairness factors of the improved DDQN algorithm, DDQN algorithm, DQN algorithm and Q-learning algorithm are compared under different obstacle numbers to verify whether the algorithm can still maintain good effectiveness in different scenarios.

As shown in Fig. 10, when the number of obstacles is 2,4,6,8 or 10, the fairness factor under the improved DDQN algorithm can always be maintained at about 0.9. The fairness factors of DDQN, DQN and Q-learning algorithms are maintained at about 0.78,0.77 and 0.67, respectively. The fairness factor under the improved DDQN algorithm is always stable and superior to other algorithms, indicating that the proposed algorithm has good stability and effectiveness.

Under the above parameter settings, after the agent passes training and learning, the dynamic deployment diagram of the UAV is shown in Fig. 11 below. Blue circles represent obstacles, and black lines represent the dynamic deployment trajectory of the UAV. The gray circles indicate the initial positions of the ground terminals, and the red circles indicate the position of the ground terminals when the UAV is serving the terminals. And the two green circles represent the start and end respectively. It should be noted that the depletion of the UAV's battery indicates the end of the UAV deployment. The end point is the last location deployed before power is exhausted. Therefore, the end point of UAV deployment in this paper is not fixed.

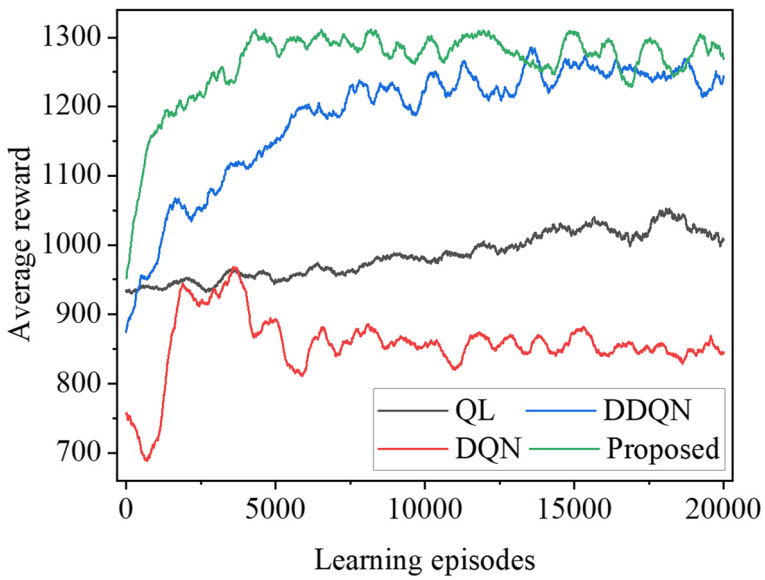

Fig. 8 Average reward under different algorithms

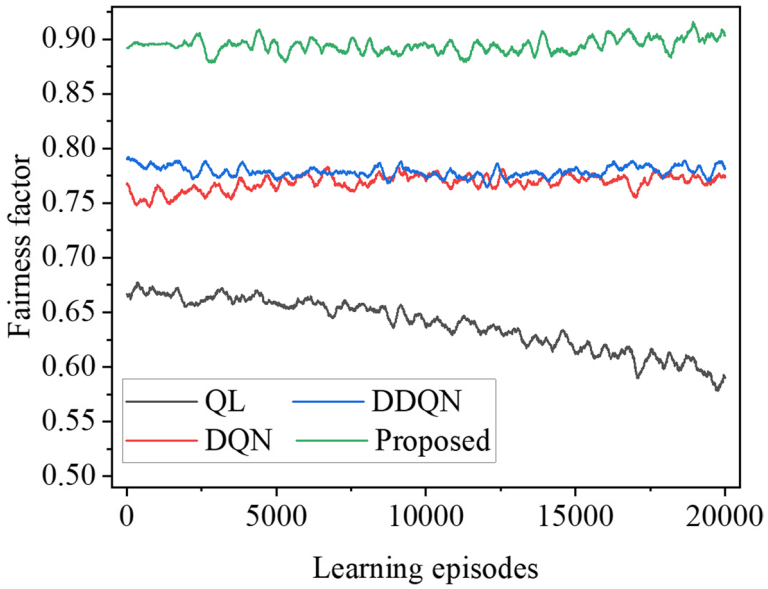

Fig. 9 Fairness factor under different algorithms

From Fig. 11(a) and (b), it can be seen that the UAV avoids obstacles from beginning to end; Secondly, it can be seen that UAVs tend to serve terminals on the ground at a closer distance, thereby saving energy consumption; In addition, it can be seen that the deployment of UAV has basically realized the fairness of offloading service of terminals provided by UAV, and can basically meet the offloading requirements of terminals at various locations. Therefore, deployment algorithm based on DDQN with ε-pseudo count selection strategy of action can still realize the dynamic deployment of a single UAV when the number, size and position of obstacles on the ground change, which has certain stability and effectiveness.

Conclusion

The dynamic deployment of a single UAV is planned track in the scenario where there are multiple mobile terminals and multiple obstacles on the ground in order to achieve the fairness of the geographical location of

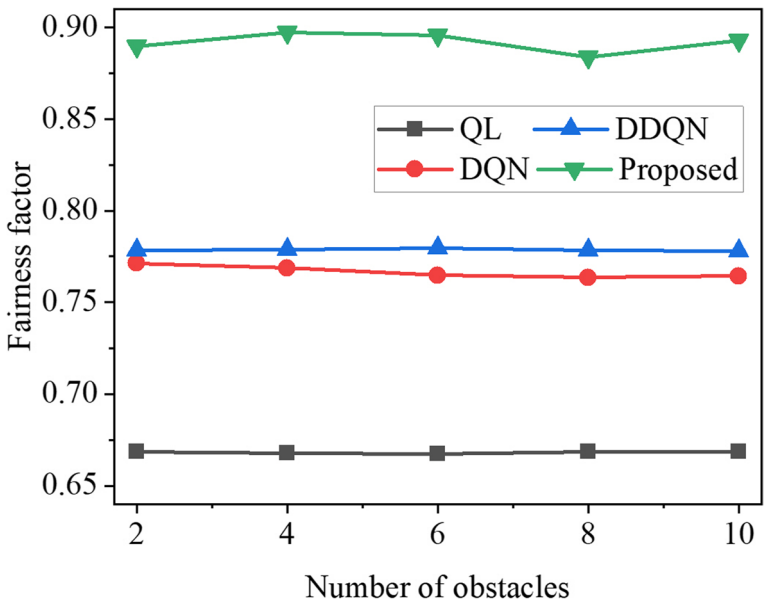

Fig. 10 Fairness factors under different algorithms with different obstacle numbers

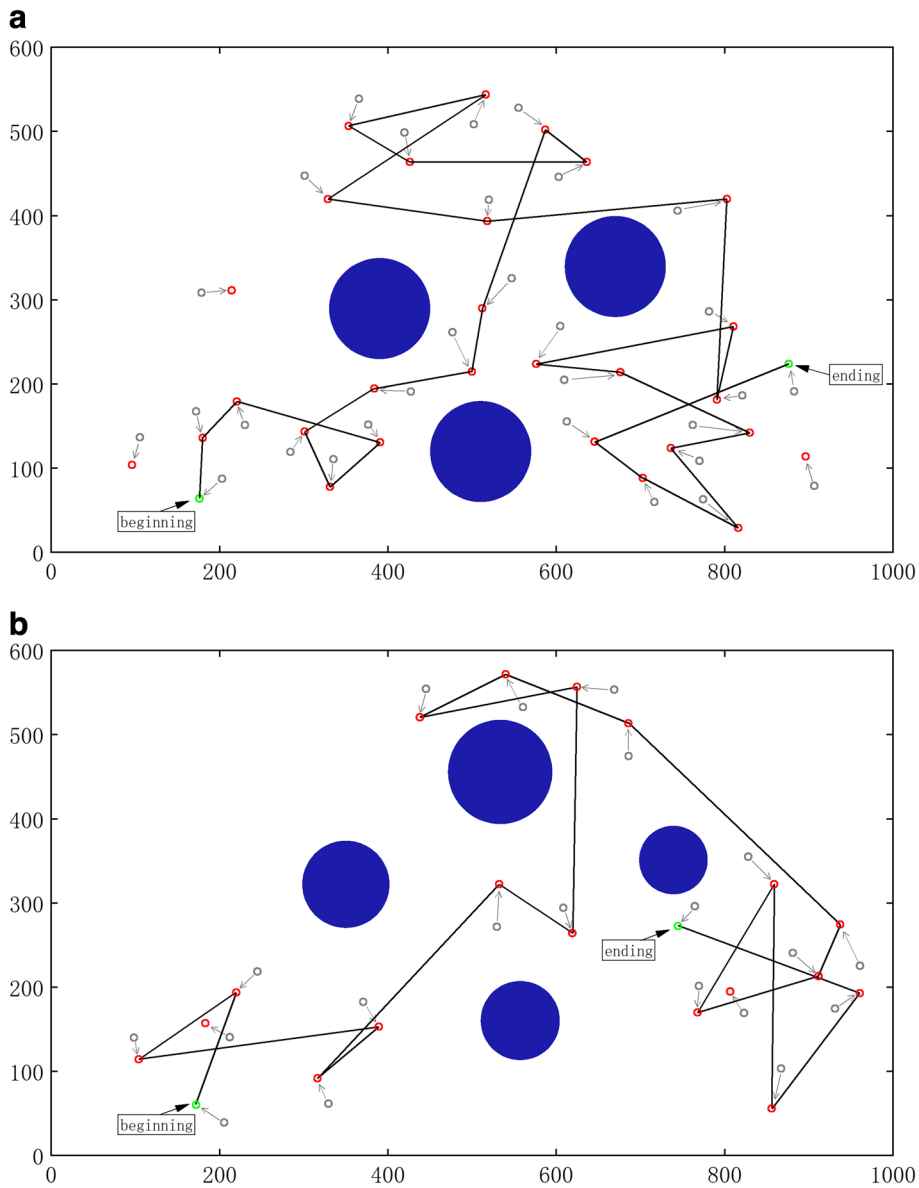

Fig. 11 Deployment trajectory diagram of the UAV

the mobile terminals, the obstacle avoidance of the UAV, and the saving of energy consumption of UAV. To solve this planning problem, a mathematical model is established firstly, including the mobile model of the terminals, the obstacle avoidance model and the energy consumption model of the UAV, and the optimization objectives and constraints are formulated. Second, in order to realize the training and learning of DRL, a MDP is established. In order to improve the training speed and average reward, the ε-greedy exploration strategy is combined with the pseudo count exploration strategy, and the ε-pseudo count selection strategy of action is proposed based on the DDQN algorithm. Finally, the environment for the simulation experiment is established. In order to achieve the optimal training effect, when using the improved DDQN algorithm to train the UAV, the hyperparameters are arranged and combined, and the hyperparameter values are adjusted until the optimal effect is achieved. The average reward value, convergence speed and the fairness of the geographical location of the ground terminals are used as evaluation indicators, and the improved algorithm in this paper is compared with the DDQN algorithm, DQN algorithm and Q-learning algorithm. The simulation results show that the improved DDQN algorithm in this paper has always maintained a good

performance. At the end of the experiment, the stability and universality of the algorithm are verified by changing the number, size and position of obstacles.

However, this paper only studies the single UAV-assisted MEC. In the case of a large number of terminals and delay-sensitive task scenarios, a single UAV obviously cannot meet the needs. Therefore, we will conduct research on multi-UAV assisted edge computing networks in the future.

Abbreviations

UAV	Unmanned Aerial Vehicle
MEC	Mobile Edge Computing
DRL	Deep Reinforcement Learning

Authors' contributions

Zhang Suqin, Zhang Lin, Xu Fei, Cheng Song, Su Weiya and Wang Sen conceived and designed the study. Zhang Suqin and Zhang Lin conceived the original idea and theoretical analysis. Xu Fei, Cheng Song, Su Weiya and Wang Sen performed the simulations. Zhang Lin wrote the paper. All authors reviewed and edited the manuscript. All authors read and approved the final manuscript.

Funding

1.Regional innovation capability guidance plan of Shaanxi Provincial Department of science and Technology (2022QFY01-14);
2.Science and technology planning project of Beilin District, Xi'an City, Shaanxi Province (GX2137);
3.Funded by Lab of High Confidence Embedded Software Engineering Technology; 4.Key R &D projects of Xianyang science and Technology Bureau (2021ZDYF-NY-0019).

Availability of data and materials

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Competing interests

The authors declare no competing interests.

Received: 21 April 2023 Accepted: 14 August 2023
Published online: 01 September 2023

References

1. Shi W, Sun H, Cao J, Zhang Q, Liu W (2017) Edge computing : a new computing model in the era of internet of everything. Comput Res Dev 54(05):907–924
2. Zhao M (2020) Overview of edge computing technology and its application. Comput Sci 47(6A):268–272
3. Zeng Y, Zhang R, Lim T (2016) Wireless communications with unmanned aerial vehicles: Opportunities and challenges. IEEE Commun Mag 54(05):36–42
4. Zhou Y, Pan C, Yeoh P (2019) Secure communications for uav-enabled mobile edge computing systems. IEEE Trans Commun 68(01):376–388
5. Yang Z, Pan C, Wang K (2019) Energy efficient resource allocation in uav-enabled mobile edge computing networks. IEEE Trans Wirel Commun 18(09):4576–4589
6. Motlagh N, Bagga M, Taleb T (2017) Uav-based iot platform: A crowd surveillance use case. IEEE Commun Mag 55(02):128–134
7. Dong C, Shen Y, Qu Y (2020) Survey of edge intelligent computing based on uav. J Intell Sci Technol 2(03):227–239
8. Yuan Y, Lei L, Vu T (2021) Energy minimization in uav-aided networks: Actor-critic learning for constrained scheduling optimization. IEEE Trans Veh Technol 70(05):5028–5042
9. Duong T, Nguyen L, Tuan H (2019) Learning-aided realtime performance optimisation of cognitive uav-assisted disaster communication. Waikoloa, HI, USA.IEEE. In: Proceedings of the 2019 IEEE Global Communications Conference. https://doi.org/10.1109/GLOBECOM38437.2019.9014313
10. Liu X, Liu Y, Chen Y (2019) Trajectory design and power control for multi-uav assisted wireless networks: A machine learning approach. IEEE Trans Veh Technol 68(08):7957–7969
11. Wang J, Jiang C, Wei Z (2018) Joint uav hovering altitude and power control for space-air-ground iot networks. IEEE Internet Things J 6(02):1741–1753
12. Huang P, Wang Y, Wang K (2020) Energy-efficient trajectory planning for a multi-uav-assisted mobile edge computing system. Front Inf Technol Electron Eng 21(12):1713–1725
13. Lai C, Wang L, Han Z (2019) Data-driven 3d placement of uav base stations for arbitrarily distributed crowds. In: Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM)
14. Jeong S, Simeone O, Kang J (2017) Mobile edge computing via a uav-mounted cloudlet: Optimization of bit allocation and path planning. IEEE Trans Veh Technol 67(03):2049–2063
15. Arafat MY, Moh S (2022) Jrcs: Joint routing and charging strategy for logistics drones. IEEE Internet Things J 9(21):21751–21764
16. Dai X, Duo B, Yuan X, Tang W (2022) Energy-efficient uav communications: A generalized propulsion energy consumption model. IEEE Wirel Commun Lett 11(10):2150–2154
17. Qi X, Yuan M, Zhang Q (2022) Joint power - trajectory scheduling optimization in a mobile uav-enabled network via alternating iteration. China Commun 19(01):136–152
18. Hu H, Zhou X, Wang Q (2022) Online computation offloading and trajectory scheduling for uav-enabled wireless powered mobile edge computing. China Commun 19(04):257–273
19. Xu Y, Zhang T, Yang D (2020) Joint resource and trajectory optimization for security in uav-assisted mec systems. IEEE Trans Commun 69(01):573–588
20. Wang D, Tian J, Zhang H (2021) Task offloading and trajectory scheduling for uav-enabled mec networks: an optimal transport theory perspective. IEEE Wirel Commun Lett 11(01):150–154
21. Wang J, Jiang C, Zhang K (2019) Distributed q-learning aided heterogeneous network association for energy-efficient iiot. IEEE Trans Ind Inf 16(04):2756–2764
22. Lillicrap T, Hunt J, Pritzel A (2015) Continuous control with deep reinforcement learning.arXiv:1509.02971 [cs.LG]. https://doi.org/10.48550/arXiv.1509.02971
23. Liu Q, Shi L, Sun L (2020) Path planning for uav-mounted mobile edge computing with deep reinforcement learning. IEEE Trans Veh Technol 69(05):5723–5728
24. Wang L, Wang K, Pan C (2020) Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing. IEEE Trans Cogn Commun Netw 7(01):73–84
25. Yin S, Yu R (2021) Resource allocation and trajectory design in uav-aided cellular networks based on multiagent reinforcement learning. IEEE Internet J 9(04):2933–2943
26. Wu Z, Yang Z, Yang C (2021) Joint deployment and trajectory optimization in uav-assisted vehicular edge computing networks. J Commun Netw 24(01):47–58
27. Bor-Yaliniz R, El-Keyi A, Yanikomeroglu H (2016) Efficient 3-d placement of an aerial base station in next generation cellular networks.Kuala Lumpur, Malaysia. IEEE. In: Proceedings of the 2016 IEEE international conference on communications (ICC). https://doi.org/10.1109/ICC.2016.7510820
28. Hu Q, Cai Y, Yu G (2018) Joint offloading and trajectory design for uav-enabled mobile edge computing systems. IEEE Internet Things J 6(02):1879–1892
29. Chang H, Chen Y, Zhang B (2021) Multi-uav mobile edge computing and path planning platform based on reinforcement learning. IEEE Trans Emerg Top Comput Intell 6(03):489–498
30. Liu Y, Liu D, Yue D (2018) Bgmm: a body gauss-markov based mobility model for body area networks. Tsinghua Sci Technol 23(03):277–287

Publisher's Note