Task Offloading in UAV-to-Cell MEC Networks: Cell Clustering and Path Planning

Mingchu Li(✉), Wanying Qi, and Shuai Li

School of Software Technology, Dalian University of Technology, Dalian 116620, China
mingchul@dlut.edu.cn

Abstract. When a natural disaster occurs, ground base stations (BSs) are destroyed and cannot provide communication services. Rapid restoration of communication is of great significance to the lives of trapped persons. This paper studies the problem of unmanned aerial vehicle (UAV) equipped with mobile edge computing (MEC) servers to provide communication and computing services for ground users in the scenario where the ground infrastructure is destroyed. We designed a UAV-to-Cell offloading system, which provides services in units of cells. By determining the hover locations (HLs) and trajectories, the UAV can handle more tasks with limited battery energy. Since tasks have time limit requirements, the order of processing will affect the task data size of the system. We solve this problem by joint cell clustering and path planning. Among them, elliptic clustering is used to divide the cells, the 3D position of the UAV is determined according to the quality of user service, and the double deep Q-network (DDQN) algorithm is used to determine the trajectory of the UAV. Simulation experiments demonstrate the effectiveness and efficiency of our proposed strategy by comparing it with the baselines.

Keywords: Unmanned aerial vehicle · Mobile edge computing · Task offloading · UAV trajectory · Double deep Q network

1 Introduction

With the rapid development of mobile Internet, smartphones and IoT devices are increasing day by day. An estimated 4.9 billion people are using the Internet in 2021 [1]. In the 5G era, emerging applications are flourishing, such as mobile payment, smart medical care, and virtual reality. To solve the problems in mobile communication, a new concept, Mobile Edge Computing (MEC) [2] is introduced. MEC deploys computing and storage resources at the edge of the mobile network to provide users with low-latency, high-bandwidth services, with more flexible deployment methods and wider application scenarios.

The number of natural disasters in the world has risen alarmingly in the first two decades of the 21st century, with a total of 7,348 disaster events recorded,

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 562, pp. 3–19, 2024.
https://doi.org/10.1007/978-3-031-54528-3_1

resulting in 1.23 million deaths [3]. How restore the communication facilities as soon as possible after the disaster is of great significance to the rescue of the trapped people. Compared with a static edge server, an unmanned aerial vehicle (UAV) has high mobility and can be flexibly deployed in most scenarios. In addition, the wireless communication link between UAV and ground equipment (UE) generally uses line-of-sight (LoS) wireless transmission, which can improve computing performance. Therefore, the UAV equipped with a MEC server can solve the above problems very well. How to provide communication services to more users is a key problem that needs to be solved.

This paper considers that the UAV provides services in units of cells. On the one hand, users with similar geographical locations may have the same type of tasks, and clusters can be naturally formed according to task types and geographical locations. On the other hand, because the purpose is to provide services to more users and handle more tasks, it can encourage users to form clusters and form greater competitiveness [4]. When the UAV server provides services, it must determine a suitable hovering location (HL) to ensure that all users in the cell can be covered and guarantee the quality of service (QoS). Due to technical limitations, the number of UAV servers that can be deployed may be very limited, especially in the chaotic hours after a natural disaster. In this paper, it is assumed that there is only one UAV server in the area to provide services for cell users. Since the task data size of each UE is different, it is more scientific to examine the energy efficiency of the UAV by taking the number of CPU cycles required for the total tasks as a measure.

In this paper, a UAV-to-Cell network is designed. We propose the problem of maximizing the CPU cycles required for the total tasks handled by the UAV, which are constrained by battery capacity. The main contributions are summarized as follows:

1. We build a UAV-to-Cell offloading system without ground infrastructure. The UAV hovers over the HL above the cell to provide services for users. To solve this problem, the optimization problem is decomposed into three independent sub-problems, namely, the ellipse clustering problem, the UAV 3D position problem, and the UAV trajectory planning problem.
2. We fit the users in the cell into an ellipse, and the optimization problem is to minimize the area of the ellipse. We calculate the number of clusters by hierarchical clustering based on the contour index and then calculate the ellipse by using the Lagrange multiplier method.
3. The 3D position of the UAV is constrained by the range of altitude and QoS for the user. Without loss of generality, we only need to consider the QoS of the most marginal users to ensure the QoS of all users in the cell.
4. Since each task has a delay requirement, the order of UAV HL selection has a great influence on the optimization problem. We adopt the path planning algorithm based on DDQN to solve the optimal path.

The rest of the paper is structured as follows. Section 2 presents related work. Section 3 illustrates the system model. Section 4 addresses the problem to be solved by dividing it into three subproblems. Section 5 illustrates performance evaluation. Section 6 presents conclusions.

2 Related Work

2.1 UAV Coverage

At present, there have been many studies on the problem of UAV 3D deployment and coverage. In [5], by studying the effects of UAV-UE antenna beam width, movement speed, and cell association on coverage probability, a framework for evaluating UAV network coverage performance is proposed. In [6], the provided mathematical model aims to determine the optimal low-altitude aerial platform (LAP) height that maximizes ground coverage while adhering to the maximum allowable path loss defined by the International Telecommunication Union (ITU). This optimal altitude is determined as a function of various urban environment statistics parameters. In [7], the author of the research proposed a path loss modeling method specifically tailored for the air-to-ground communication scenario involving LAPs in urban environments. In [8], the approach considers the placement of UAVs to achieve the best possible coverage in a given area, taking into account factors such as the number of UAVs, antenna gain, and beam width. In [9], the author's proposal introduces an "ellipse clustering algorithm" for optimizing the deployment of UAVs in a wireless communication network. This algorithm leverages the concept of ellipses to represent the radiation beam pattern of the UAVs' antennas, ensuring that QoS requirements are met while optimizing various parameters, including UAV positions, transmission power levels, and deployment density. The ultimate goal is to minimize the total energy consumption of the UAVs. In [10], the author's proposal introduces an "ellipse clustering algorithm" for optimizing the deployment of UAVs in a wireless communication network. This algorithm leverages the concept of ellipses to represent the radiation beam pattern of the UAVs' antennas, ensuring that QoS requirements are met while optimizing various parameters, including UAV positions, transmission power levels, and deployment density. The ultimate goal is to minimize the total energy consumption of the UAVs.

Most of the UAV coverage issues focus on dividing an area into multiple subareas, and each subarea has a UAV to provide services. The main difference is the clustering method or the method of calculating the optimal height.

2.2 Path Planning

Path planning is an important issue in UAV-MEC networks. In recent years, many studies have focused on path planning. Most of the solutions to path planning use traditional optimization algorithms. In [11], the path planning is transformed into a multi-objective optimization task, and the goals of the total

path length and terrain threat level are optimized, and the UAV precise path planning is based on the enhanced multi-objective intelligent algorithm. In [4,12], the author employs an auction algorithm and a gap-adjusted branch-and-bound algorithm for determining the trajectory of a UAV or vehicle server in a communication or service delivery context with a focus on maximizing average throughput. In [13], the author's approach involves modeling the sub-region division problem as a semi-discrete optimal transport problem and solving it iteratively. Additionally, the UAV trajectory optimization problem is modeled as a traveling salesman problem (TSP) to find the shortest route. Some studies also explore the use of reinforcement learning algorithms for solving path-planning problems. In [14], through the application of DRL, the author achieves optimal path planning and task assignment for UAVs while minimizing task completion time and maximizing system efficiency. This is accomplished by constructing both a Q-network and a policy network. In [15], the proposed framework addresses trajectory planning for multiple UAVs within a MEC network. It leverages DRL techniques to determine optimal trajectories for these UAVs with the specific objectives of minimizing energy consumption and reducing latency for MEC tasks.

These papers do not take into account the limitations of the flying height and coverage of UAVs. They usually assume that the flying height of UAVs is a fixed value, which does not conform to the actual use of UAVs.

3 System Model

We consider a UAV-to-Cell MEC network serving a finite area $\mathcal{D}$ as shown in Fig. 1, where ground BSs are destroyed due to the natural disaster. There are a UAV server and $k \in [K]$ cells in the area $\mathcal{D}$ following random distribution, and the user equipments (UEs) in the cell obey the normal distribution. There are

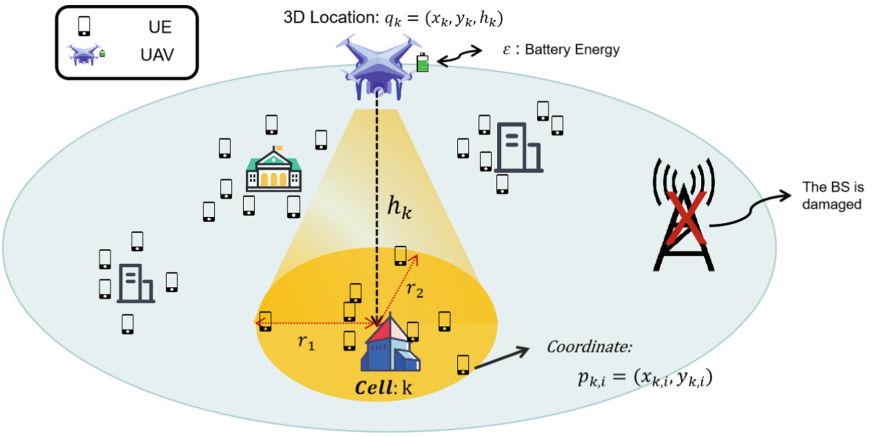


Fig. 1. Illusion of a UAV-to-Cell MEC network system.

N UEs in area $\mathcal{D}$, and UE $i \in [N_k]$ in the cell k, it is obvious that $\sum_{i=1}^{k} N_i = N$. The UAV server stays at the hover locations (HLs) above each cell, and the UEs in this cell can offload tasks to the UAV. $q_k = (x_k, y_k, h_k)$ is a three-dimensional coordinate that represents the HL coordinate of the UAV server at the cell s_k. $p_{k,i} = (x_{k,i}, y_{k,i})$ represents the coordinates of UE i in the cell k. In the cell k, the communication distance between the UAV server and UE i is $d_{k,i} = \|q_k - p_{k,i}\|$. Each UE has a task $A_{k,i} = \{I_{k,i}, \beta_{k,i}, \tau_{k,i}\}$, where $I_{k,i}$ indicate the data size of the task, $\beta_{k,i}$ indicate the number of CPU cycles required per bit of the task, i.e., $I_{k,i} \cdot \beta_{k,i}$ indicate the total number of CPU cycles required to complete the task, $\tau_{k,i}$ indicate the task deadline. Beyond the deadline, the task can not be offloaded. We assume that the battery energy of the UAV server is not enough to handle the tasks of all cells, and when the remaining energy is not enough to handle the tasks of any cell, the UAV server will end the flight.

3.1 Communication Model

Non-orthogonal multiple access (NOMA) can improve spectral efficiency and is considered to be a promising multiple access technology for 5G networks. When the UAV server hovers over the cell k, the UEs in the cell k offload their tasks to the UAV server for processing through NOMA technology. For the uplink link, we employ a probabilistic path loss model. The path loss between UE i in the cell k and the UAV server hovering over this cell is as follows:

$$PL_{k,i} = \begin{cases} FSPL_{k,i} + \varepsilon_{LoS}, & LoSLink \\ FSPL_{k,i} + \varepsilon_{NLoS}, & NLoSLink \end{cases} \tag{1}$$

where $FSPL_{k,i}$ is the free-space path loss between the ith UE in the cell k and the UAV server [7]. $FSPL_{k,i} = 20logd_{k,i} + 20\log(f_{MHz}) - 27.55$, where f_{MHz} is the system center frequency in MHz. Variables ε_{LoS} and ε_{NLoS} are the additional path losses in dB. For the UAV-UE link, the LoS probability is $P_{k,i}^{LoS} = 1/(1 + a \cdot exp(-b[\theta_{k,i} - a]))$, where $\theta_{k,i}$ is the elevation angle between the UAV server and UE i in the cell k, a and b are parameters reflecting the environmental impact. The NLOS probability is $P_{k,i}^{NLoS} = 1 - P_{k,i}^{LoS}$. The average path loss of the UAV-UE link:

$$\overline{PL}_{k,i} = FSPL_{k,i} + P_{k,i}^{LoS} \cdot \varepsilon_{LoS} \ + P_{k,i}^{NLoS} \cdot \varepsilon_{NLoS} \tag{2}$$

According to NOMA, sequential interference cancellation (SIC) is used by the user according to the channel condition. In the cell k, the channel between the UAV server and UE i is represented by $h_{k,i} = \frac{g_{k,i}}{\overline{PL}_{k,i}}$, where $g_{k,i}$ denotes Rayleigh fading channel gain [16]. The received signal-to-interference-plus-noise ratio (SINR) of UE i can be expressed as

$$SINR_{k,i} = \frac{p_{k,i} \cdot |h_{k,i}|^2}{\sum_{j \neq i} p_{k,j} \cdot |h_{k,j}|^2 + \sigma^2} \tag{3}$$

where $p_{k,i}$ is the upload power of the UE, and σ^2 is the noise power, which represents the variance generated by all other signals and interference received during the receiving process. In this case, to guarantee QoS, $SINR_{k,i}$ of UE i covered by UAV must be greater than the minimum SINR threshold δ_{th}. Assume that the binary variable $a_{k,i}$ represents the permission control of UE i task uploading in the cell k, if the processing completion time of UE i does not exceed the task deadline, then $a_{k,i} = 1$, otherwise, $a_{k,i} = 0$. The achievable upload rate between UE i and the UAV can be calculated:

$$\begin{aligned} R_{k,i}^{u} &= a_{k,i} \cdot B \cdot log(1 + SINR_{k,i}) \\ &= a_{k,i} \cdot B \cdot log(1 + \frac{p_{k,i} \cdot |h_{k,i}|^2}{\sum_{j \neq i} p_{k,j} \cdot |h_{k,j}|^2 + \sigma^2}) \end{aligned} \tag{4}$$

where B is the wireless channel bandwidth. In the cell k, the transmission delay between UE i and the UAV server is:

$$t_{k,i}^{u} = \frac{a_{k,i} \cdot I_{k,i}}{R_{k,i}^{u}} \tag{5}$$

3.2 Computation Model

UAV energy consumption is composed of computing energy consumption, hovering energy consumption, flight energy consumption, and communication energy consumption. Since the proportion of communication energy consumption is very small compared with the other three parts [17], it is ignored in this paper.

Computing Energy Consumption Model. The computing energy consumption of the UAV server is determined by the task CPU cycles of the UE and the processing capacity of the UAV server. The CPU frequency of the UAV server is represented by F_{UAV} (CPU cycles per second). The processing delay when the UAV server calculates the task $A_{k,i}$ is: $t_{k,i}^{com} = \frac{w_{k,i}}{F_{UAV}}$ where $w_{k,i} = a_{k,i} \cdot I_{k,i} \cdot \beta_{k,i}$ is the number of CPU cycles required to complete the task $A_{k,i}$.

In the cell k, the energy consumed by the UAV server to process the tasks of the UE i is:

$$E_{k,i}^{com} = P_{k,i}^{com} \cdot t_{k,i}^{com} = c \cdot w_{k,i} \cdot F_{UAV}^2 \tag{6}$$

where c is a constant that depends on the chip architecture of the UAV server [18]. The energy consumed by the UAV to process all UE tasks in the cell k is:

$$E_{k}^{com} = c \cdot \sum_{i=1}^{N_k} w_{k,i} \cdot F_{UAV}^2 \tag{7}$$

Hovering Energy Consumption Model. When the UAV server hovers over the cell k, all qualified UEs will offload the task at the same time. Due to the different task data sizes and transmission rates of each UE, the arrival time of

each task is different, and it needs to be queued for processing. First-come-first-serve (FCFS) queue schedule is adopted to ensure that tasks can be processed in a certain order. The time when the UE offloaded the last task is: $t^u_{k,last} = \max_i t^u_{k,i}$, the number of task CPU cycles completed by the UAV server is $w_{k,last} = t^u_{k,last} \cdot F_{UAV}$. Then the remaining task CPU cycles are $w_{k,res} = \sum_{i=1}^{N_k} w_{k,i} - w_{k,last}$, The hover delay of the UAV server at the cell k is: $t^{hover}_k = t^u_{k,last} + w_{k,res}/F_{UAV}$. The hover energy consumption is:

$$\begin{aligned} E^{hover}_k &= P_h \cdot t^{hover}_k \\ &= P_h \cdot (\max_i t^u_{k,i} + \frac{\sum_{i=1}^{N_k} w_{k,i} - w_{k,last}}{F_{UAV}}) \end{aligned} \tag{8}$$

where $P_h = \frac{F\sqrt{F}}{\eta\sqrt{\frac{1}{2}\pi c r^2 \rho}}$ is the hover power of the UAV [13].

Flight Energy Consumption Model. When the air resistance experienced by the UAV is equal to the weight of the UAV itself, the UAV's flight efficiency is the highest. This paper does not consider the acceleration changes when the UAV starts and stops, and the optimal flight speed of the UAV is: $v_{UAV} = \frac{2 \cdot o \cdot g}{\rho \cdot s \cdot \mu_f}$, where o represents the weight of the UAV, s represents the wing area, and μ_f represents the drag coefficient. The flight energy consumption of the UAV from the cell k to $k+1$ is:

$$E^{fly}_{k,k+1} = P_m \cdot \frac{\|q_k - q_{k+1}\|}{v_{UAV}} \tag{9}$$

where $P_m = \frac{1}{2} \cdot \rho \cdot {v_{UAV}}^3 \cdot s \cdot \mu_f + v_{UAV} \cdot F_T$ is the flying power of the UAV. F_T is the thrust of the UAV.

The total energy consumption of the UAV in area $\mathcal{D}$ is related to the path passed by the UAV. The path of the UAV is represented by the sequence of processed cells: $q_m, m \in [M], [M] \subseteq [K]$. The total energy consumption can be expressed as:

$$E_{total} = \sum_{m=1}^{M} (E^{com}_m + E^{hover}_m) + \sum_{m=1}^{M-1} E^{fly}_{m,m+1} \tag{10}$$

3.3 Problem Formulation

In the proposed UAV-to-Cell MEC network, processing as many task CPU cycles as possible is our goal. We maximize the total task CPU cycles handled by joint cell clustering and UAV path planning. Therefore, the optimization problem denoted by $P1$ can be formulated as:

$$\begin{aligned} \text{P1}: \quad & \max_{k,m,q_k} \sum_{m=1}^{M} \sum_{i=1}^{N_m} w_{m,i} \\ s.t. \quad & C1: \quad \sum_{k=1}^{K} |N_k| = N, \end{aligned}$$

$$
\begin{aligned}
C2: &\quad \mathrm{SINR}_{k,i} \geq \delta_{\mathrm{th}}, \quad (k \in K, i \in N_k) \\
C3: &\quad h_{\min} \leq h_k \leq h_{\max}, \quad (k \in K) \\
C4: &\quad E_{\mathrm{total}} \leq \epsilon, \\
C5: &\quad a_{k,i} \in \{0,1\}, \quad (k \in K, i \in N_k) \\
C6: &\quad (2)-(3), (7)-(10)
\end{aligned}
$$

Constraint C1 ensures that the area division covers all UEs, and constraint C2 guarantees the QoS of UEs. Since the flying height of the UAV is limited, constraint C3 ensures that the flying height of the UAV is within the allowable range. Constraint C4 means that the total energy consumption of the UAV cannot exceed the total battery capacity of the UAV. Through the above formula, we can observe that the $SINR_{k,i}$ in C2 and the energy consumption in C4 are related to the UAV coordinates, so the constraints C2, C4, and C3 are coupled. Constraint C5 indicates whether the task $A_{k,i}$ can be offloaded to the UAV for processing.

4 Problem Analysis and Algorithm

Problem P1 is a MINLP problem, and it is very difficult to solve directly. The optimization variables k and m are independent of each other. We can use the coordinates of the UAV as an entry point, and we can convert problem $P1$ into three sub-problems, namely the ellipse clustering problem, the UAV 3D location problem, and the UAV path planning problem.

4.1 Ellipse Clustering Problem

Due to the practical radiation beam pattern being an ellipse, we consider the UAV server to cover the UEs in an ellipse by adjusting the orientation of the UAV and the HPBWs of the antenna [9]. In the process of rescuing trapped people, each user must be clustered into a cell. Firstly, the UEs on the ground are processed by hierarchical clustering based on the contour index, and the UEs can be merged into K clusters according to the similarity of the user coordinates. After determining the number of clusters K, the users are fitted into an ellipse with the smallest area. For the kth cluster, the problem $P2$ can be expressed as:

$$
\begin{aligned}
P2: &\quad \min_{a_k, b_k, x_k, y_k} \pi \cdot a_k \cdot b_k \\
\text{s.t.} &\quad \frac{(x_i - x_k)^2}{a_k^2} + \frac{(y_i - y_k)^2}{b_k^2} \leq 1 \quad i \in N_k
\end{aligned}
$$

where a_k, b_k is the semi-major axis and the semi-minor axis of the ellipse respectively, (x_k, y_k) are the center coordinates of the ellipse, and (x_i, y_i) are the coordinates of the UE i in cluster k. This is a nonlinear optimization problem with constraints. The following Lagrangian function can be obtained by converting the constraints into the form of Lagrange multipliers: $L(a_k, b_k, x_k, y_k, \lambda_i) =$

$\pi a_k b_k + \sum_{i=1}^{N_k} \lambda_i \left(1 - \frac{(x_i - x_k)^2}{a_k^2} - \frac{(y_i - y_k)^2}{b_k^2}\right)$ where λ_i is the Lagrange multiplier. Let $\frac{\partial L}{\partial a_k} = \frac{\partial L}{\partial b_k} = \frac{\partial L}{\partial x_k} = \frac{\partial L}{\partial y_k} = \frac{\partial L}{\partial \lambda_i} = 0$. This system of equations can be solved using the gradient descent method. The specific implementation process is shown in Algorithm 1.

Algorithm 1. Ellipse Clustering algorithm

Input: U : Coordinates of UEs; K_{max} : Maximum clusters; α : Learning rate; I_{max} : Maximum iterations

Output: Number of clusters K; Ellipse fit parameters (a_k, b_k, x_k, y_k)

Calculate the distance matrix $M_d = Dist_{Matrix}(U)$
Initialize the list of silhouette coefficients $Silh_{scores} \leftarrow \{\}$
for $k = 2$ to K_{max} **do**
    M_{link}=$Hierarchy_{Linkage}(M_d)$
    $clusters = Hierarchy_{Fcluster}(M_{link}, k)$
    $Silh_{scores} \leftarrow Silhouette_{Score}(M_d, clusters)$
end for
if $np.argmax(Silh_{scores}) \neq 0$ **then**
    $K = np.argmax(Silh_{scores}) + 2$
else
    $K = 2$
end if
$U_k = AgglomerativeClustering(K)$
for $k = 1$ to K **do**
    for $i = 0$ to I_{max} **do**
        $\lambda_{i+1} = \lambda_i - \alpha \frac{\partial L}{\partial \lambda}$,
        $a_{i+1} = a_i - \alpha \frac{\partial L}{\partial a}$, $b_{i+1} = b_i - \alpha \frac{\partial L}{\partial b}$, $x_{i+1} = x_i - \alpha \frac{\partial L}{\partial x}$, $y_{i+1} = y_i - \alpha \frac{\partial L}{\partial y}$
    end for
    $a_k = a_i, b_k = b_i, x_k = x_i, y_k = y_i$
end for

4.2 The UAV 3D Location Problem

The UEs of the cell are divided into an ellipse cluster according to the coordinates. We take the center of the ellipse cluster as the plane coordinates (x_k, y_k) of the HL. We assume that the azimuth and elevation HPBWs of the UAV directional antenna are not equal, and they are expressed as $\theta_1, \theta_2 \in (0^o, 90^o)$, as shown in the Fig. 2.

Our goal is to determine a suitable UAV height and HPBWs angle, which minimizes the UAV's communication energy consumption and hovering energy consumption. We achieve this by minimizing the path loss between the UAV and the UEs. To guarantee the QoS of all UEs, we only need to guarantee the QoS between the farthest edge UE and the UAV of the cell. In the cell k, the UE farthest from the UAV is N_e, and its coordinates are $p_{k,e} = (x_{k,e}, y_{k,e})^T$.

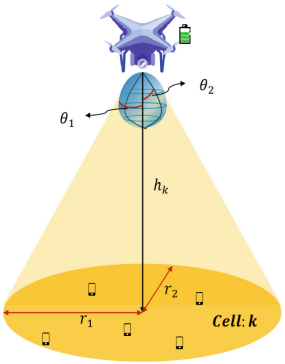


Fig. 2. Downlink between the UAV and UEs in the cell k.

$\overline{PL_e}$ is the average path loss of the cell edge UE which is farthest from the UAV. Problem $P3$ can be formulated as:

$$\begin{aligned} P3: \quad & \min_{h_k} \overline{PL_e} \\ \text{s.t.} \quad & h_{\min} \le h_k \le h_{\max} \quad (k \in K) \\ & \text{SINR}_e \ge \delta_{\text{th}} \end{aligned}$$

Figure 3 shows the average path loss for UAV heights for different environments and horizontal distances. From the Fig. 3, we can observe that the average path loss of UEs increases with the horizontal distance. But when the UAV height is too low, the influence of NLoS links dominates the path loss, which leads to a sharp increase in the average path loss. According to the quasi-convexity of $\overline{PL_e}$, we have to find h^* that minimizes $\overline{PL_e}$ under $h_{min} \le h^* \le h_{max}$. Finally, using this result, we can also finalize the corresponding HPBWs for the UAV: $\theta_i = tan^{-1}(\frac{h^*}{r_i})$ $(i = 1, 2)$, where r_i represents the major/minor axis of the ellipse.

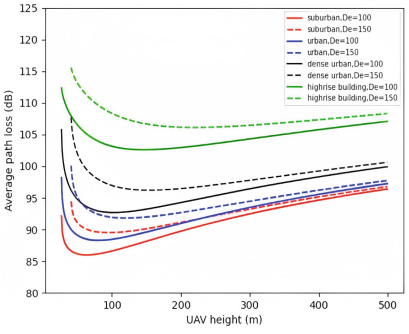


Fig. 3. The average path loss for UAV heights.

4.3 The UAV Path Planning Problem

Algorithm 2. DDQN-based Path Planning algorithm

Input: M : Training rounds; N : Training times; **batch_size** : Sample size
Output: DDQN model

Initialize the weight parameters of the main network $Q(\mathcal{s}, a)$ and the target network $Q'(\mathcal{s}, a)$
Initialize experience replay buffer D
for $eposide = 0$ to M **do**
    Initialize the environment
    Initialize state $\mathcal{s}_0$
    for $t = 0$ to N **do**
        Observe the current state of the environment $\mathcal{s}_t$,and choosing an action based on the main network a_t.
        Execute action a_t, observing reward $\mathcal{r}_t$, the next state $\mathcal{s}_{t+1}$ and end condition *done*.
        if done **then**
            break
        end if
        $D \leftarrow (\mathcal{s}_t, a_t, \mathcal{r}_t, \mathcal{s}_{t+1})$
    end for
    if $len(D) >$ **batch_size then**
        Randomly sample a batch of data from the experience replay buffer for training the main network $Q(\mathcal{s}, a)$.
        Update the weight parameters of the main network according to the difference between the Q value and the target Q value.
    end if
    if $epsoide\%10 = 0$ **then**
        Copy the weight parameters of the main network to the target network
    end if
end for

The coordinates of the HL in each cell have been determined, and the method of deep reinforcement learning can be used for path planning. In our DDQN model, the UAV acts as an agent that feeds back the current state in real time and determines the best action by interacting with the environment, to maximize the observed reward $\mathcal{r}_t$ in the decision step. At each step t, the UAV observes the state $\mathcal{s}_t$ from the state space, then selects an action a_t based on the Q network. DDQN-based path planning algorithm is proposed as Algorithm 2. First, we define the state, action, and reward of the UAV environment.

- State $\mathcal{s}$: The state represents a set of information about the current situation of the environment, including the current remaining battery energy of the UAV (e^{res}), the location of the UAV (p^{UAV}), the remaining task CPU cycles of each cell (w_k), and the positions of all HLs (q_k). In each episode, the initial state of the UAV is randomly determined.

- Action a: The action refers to the UAV choosing the next HL according to the current state $\mathcal{s}$. We use the ε-greedy algorithm to select an action, and ε will continue to decrease each round until it reaches a threshold.
- Reward $\mathcal{r}$: The reward is a scalar indicating how good or bad the action a is for the agent in its current state $\mathcal{s}$. In our DDQN model, after the UAV determines the action a according to the state $\mathcal{s}$, we take the task CPU cycles in the selected cell k as the reward. If the selected HL has been processed before, a penalty is given.

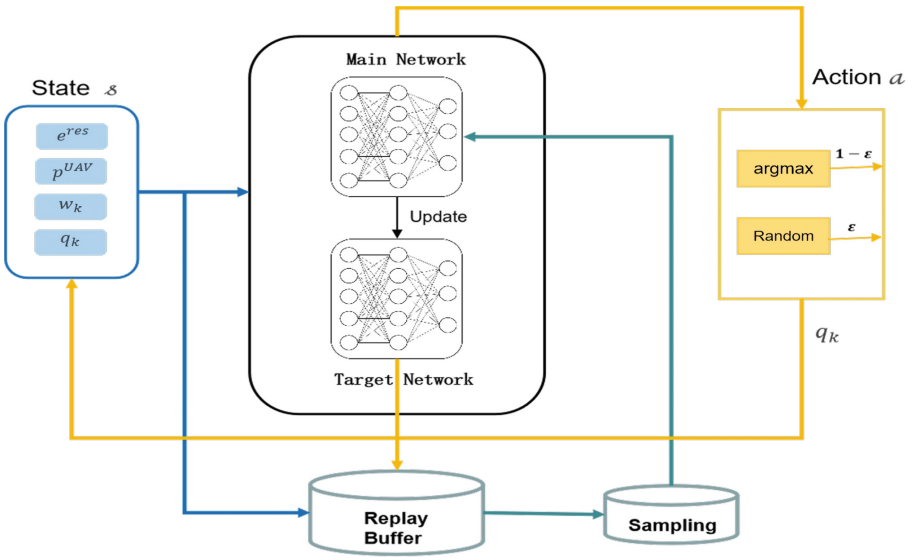


Fig. 4. Schematic of the proposed DDQN-based path planning algorithm.

The schematic of the proposed DDQN-based path planning algorithm is shown in Fig. 4. Then, preprocess the initial state $\mathcal{s}_0$ to normalize the data. Then use a Deep-Learning Neural Network (DNN) to fit a function for generating the Q value. The input of the DNN is the state, and the output is an estimate of the value function for each possible action. During training, the reward is updated based on the current state, the action chosen, the reward obtained, and the next state. The updated formula is:

$$Q(\mathcal{s}_t, a_t) \leftrightarrow \mathcal{r}_t + \gamma Q'(\mathcal{s}_{t+1}, \arg\max_{a} Q(\mathcal{s}_{t+1}, a)) \tag{11}$$

where $\mathcal{s}_t, a_t$ indicates the current state and action at step t. $\mathcal{r}_t$ is the reward at the t step. γ is the discount factor ($0 \leq \gamma < 1$) which determines the importance of future rewards. It controls how the value function of the current state will be affected by future rewards.

We have introduced a penalty mechanism for some bad actions, for example, when the UAV is not moving or the UAV is flying to a HL that has been processed, we set the reward to a negative value. During the learning and updating process, we use the experience replay buffer to update and store the collected environment samples. At each step, the observed state, action, reward, and next state are stored in it. The sequence of states is then randomly sampled in mini-batches from the buffer to update the DDQN network.

5 Simulation and Experiment

Table 1. Simulation parameters

Simulation parameters	Value
Bandwidth B	10 MHz
Noise power σ^2	-100 dBm
CPU frequency F_{UAV}	1.2 GHz
Battery capacity ϵ	30 kJ, 50 kJ
SINR threshold δ_{th}	0 dB
Additional path losses ε_{LoS}, ε_{NLoS}	3 dB, 34 dB
Air density ρ	1.225 kg/m^3
Weight o [21]	2 kg
Wing area s	0.5 m^2
Drag coefficient u_f	0.05

In this section, we conduct simulation experiments to verify the effectiveness and efficiency of our solution, and numerical results are given. We assume that the UAV-to-Cell MEC network includes a UAV, and the number of UEs is 600, where these UEs are distributed in an area of 2×2 km^2 [4]. Each UE has one task waiting to be processed, the task data size varies between $[10, 30]$ MB, and the upload power of each UE varies between $[0.1, 0.6]$ W. Assuming that the computing resources of the UAV are sufficient, the CPU frequency of the UAV is $F_{UAV} = 1.2$ GHz [18]. The power of the UAV is insufficient to provide services for the UEs in all cells, and the battery capacity $\epsilon = 30$ kJ, the bandwidth $B = 10$ MHz [19], the noise power $\sigma^2 = -100$ dBm [18,20]. In the free-space path loss model, the additional path losses for LoS and NLoS are $\varepsilon_{LoS} = 3$ dB and $\varepsilon_{NLoS} = 34$ dB [9], respectively. The simulation environment parameters are shown in Table 1.

To verify the high efficiency of our proposed strategy (DDQN), we compare it with the following baselines.

- DDQN-FH: The HL height of the UAV is fixed at $h = 200$, and the DDQN algorithm is used for path planning. It should be noted that the flight power of horizontal flight is slightly less than that of oblique flight.

- Greedy-R: Determine the height of the HL of the UAV according to the SINR, and use the ratio of task CPU cycles to energy consumption as the optimal choice in the current state.
- Greedy-MD: The HL height of the UAV is determined according to the SINR, and the nearest HL is taken as the next choice.

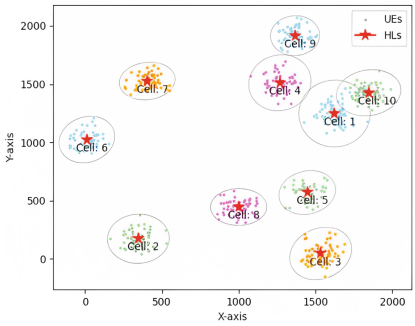


Fig. 5. UEs-UAV associations based on ellipse clustering.

Figure 5 shows the results of our proposed ellipse clustering. The UEs of each cell are fitted into an ellipse, and the center of the ellipse is the horizontal coordinate of the HL of the UAV. We assume that UAV only provides services for the currently hovering cells, even if the results of ellipse fitting may overlap and do not generate inter-cell interference.

The trajectory of the UAV is shown in the Fig. 6, which is a path calculated according to the algorithm proposed in this paper. The red point indicates the 3D HL of the UAV in different cells, and the blue line indicates the path of the UAV. The starting cell is 4 and the end cell is 7. Since the initial HL is randomly generated, we calculate the reward of each path, choose a path with the largest reward, and draw it, as shown in Fig. 7. The size of the HL in the figure indicates the relative number of task CPU cycles of the cell. It can be observed that the number of cells selected by the Greedy-MD and Greed-R algorithms is smaller than that of the reinforcement learning algorithm. The Greed-R algorithm tends to select cells with smaller energy consumption and larger task CPU cycles. The Greedy-MD algorithm selects the nearest HL as the next HL, without considering the task CPU cycle size of the hovering cell, resulting in performance loss. Both DDQN and DDQN-FH algorithms select cell 1 and cell 10, which are relatively closer and have larger task CPU cycles, which are relatively better.

Energy efficiency refers to the total task CPU cycles processed by the UAV divided by the energy consumption of the UAV, which can reflect the energy consumed by the UAV to process each task CPU cycle. Figure 8 shows the training results of average reward over different algorithms. Because the setting of the reward is the ratio of the task amount to the energy consumption, the Energy

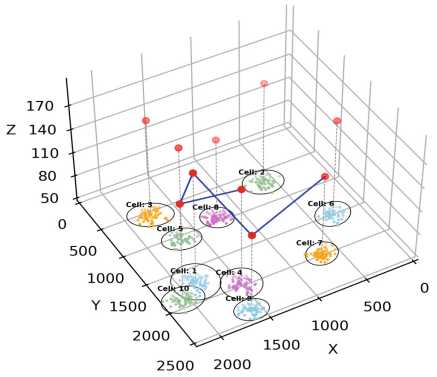


Fig. 6. The UAV 3D trajectory for DDQN.

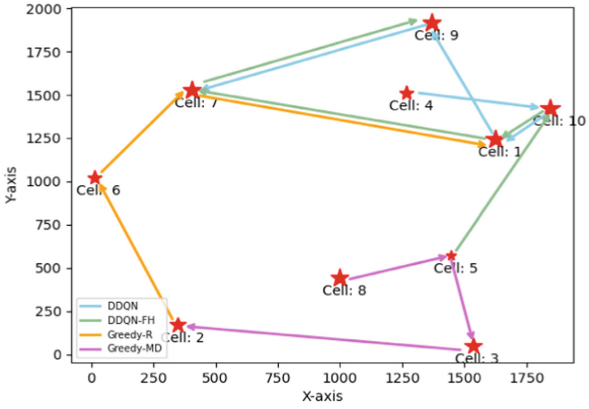
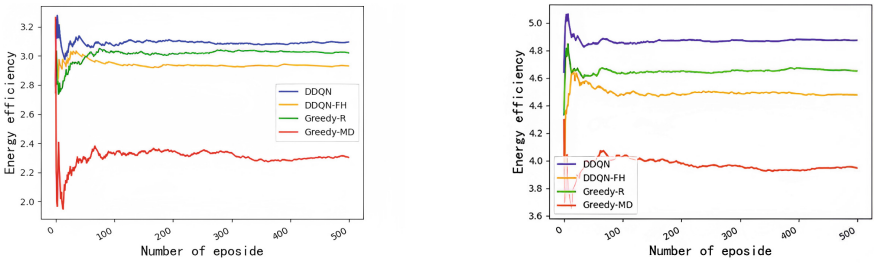


Fig. 7. Illustration of the UAV trajectories based on four algorithms.



(a) $\epsilon = 30$ kJ (b) $\epsilon = 50$ kJ

Fig. 8. Comparison of the energy efficiency with four algorithms

efficiency is a value between [1.8, 3.2]. It can be observed from the figure that our proposed algorithm has a higher energy efficiency than the other three algorithms. In the DDQN-FH algorithm, the height of the UAV is fixed at 200 m, and the path loss is greater than that of DDQN. When the signal drops to a certain level, reliable communication cannot be carried out, and problems such as packet loss occur. The Energy efficiency is slightly lower than the DDQN algorithm. In order to meet the limitation of the limited power of the UAV in this paper, we set the battery capacity of the UAV to be small. Figure 8a is the case where the UAV battery capacity is 30 kJ, and Fig. 8b is the case where the UAV battery capacity is 50 kJ. We can observe that when the battery capacity is 50 kJ, the energy efficiency of the system is higher than 30 kJ. Because of the increased battery capacity, the UAV can serve more cell users.

6 Conclusion

In this paper, we study the UAV-to-Cell MEC network in situations where the ground infrastructure is unavailable, with the goal of maximizing the number of task CPU cycles to complete the task. We combine cell clustering and path planning to achieve task offloading in UAV networks. Considering the radiation pattern of the antenna, we adopt elliptic clustering to realize unit clustering. A DDQN strategy is proposed to compute UAV trajectories. Simulation results demonstrate the efficiency and effectiveness of our strategy.

Acknowledgments. This paper is supported by the National Nature Science Foundation of China under grant number: T2350710232.

References

1. ITU, DBM.: Measuring digital development - Facts and figures 2021. ITU Publication (2021)
2. Luo, Q., Hu, S., Li, C., Li, G., Shi, W.: Resource scheduling in edge computing: a survey. IEEE Commun. Surv. Tutor. **23**(4), 2131–2165 (2021)
3. UNDRR: Human cost of disaster - An overview of the last 20 years (2020)
4. Ning, Z., et al.: 5G-enabled UAV-to-community offloading: joint trajectory design and task scheduling. IEEE J. Sel. Areas Commun. **39**(11), 3306–3320 (2021)
5. Sun, H., et al.: Coverage analysis for cellular-connected random 3D mobile UAVs with directional antennas. IEEE Wirel. Commun. Lett. **12**(3), 550–554 (2023)
6. Al-Hourani, A., Kandeepan, S., Lardner, S.: Optimal LAP altitude for maximum coverage. IEEE Wirel. Commun. Lett. **3**(6), 569–572 (2014)
7. Al-Hourani, A., Kandeepan, S., Jamalipour, A.: Modeling air-to-ground path loss for low altitude platforms in urban environments. In: 2014 IEEE Global Communications Conference (GLOBECOM 2014), pp. 2898–2904. IEEE (2014)
8. Mozaffari, M., Saad, W., Bennis, M., Debbah, M.: Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage. IEEE Commun. Lett. **20**(8), 1647–1650 (2016)

9. Noh, S.C., Jeon, H.B., Chae, C.B.: Energy-efficient deployment of multiple UAVs using ellipse clustering to establish base stations. IEEE Wirel. Commun. Lett. **9**(8), 1155–1159 (2020)
10. Babu, N., Virgili, M., Papadias, C.B., Popovski, P., Forsyth, A.J.: Cost- and energy-efficient aerial communication networks with interleaved hovering and flying. IEEE Trans. Veh. Technol. **70**(9), 9077–9087 (2021)
11. Wan, Y., Zhong, Y., Ma, A., Zhang, L.: An accurate UAV 3-D path planning method for disaster emergency response based on an improved multiobjective swarm intelligence algorithm. IEEE Trans. Cybern. **53**(4), 2658–2671 (2023)
12. Liu, Y., Li, Y., Niu, Y., Jin, D.: Joint optimization of path planning and resource allocation in mobile edge computing. IEEE Trans. Mob. Comput. **19**(9), 2129–2144 (2020)
13. Wang, D., Tian, J., Zhang, H., Wu, D.: Task offloading and trajectory scheduling for UAV-enabled MEC networks: an optimal transport theory perspective. IEEE Wirel. Commun. Lett. **11**(1), 150–154 (2022)
14. Chang, H., Chen, Y., Zhang, B., Doermann, D.: Multi-UAV mobile edge computing and path planning platform based on reinforcement learning. IEEE Trans. Emerg. Top. Comput. Intell. **6**(3), 489–498 (2021)
15. Wang, L., Wang, K., Pan, C., Xu, W., Aslam, N., Hanzo, L.: Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing. IEEE Trans. Cogn. Commun. Netw. **7**(1), 73–84 (2021)
16. Zhang, N., Wang, J., Kang, G., Liu, Y.: Uplink nonorthogonal multiple access in 5G systems. IEEE Commun. Lett. **20**(3), 458–461 (2016)
17. Zeng, Y., Zhang, R.: Energy-efficient UAV communication with trajectory optimization. IEEE Trans. Wirel. Commun. **16**(6), 3747–3760 (2017)
18. Hu, Q., Cai, Y., Yu, G., Qin, Z., Zhao, M., Li, G.Y.: Joint offloading and trajectory design for UAV-enabled mobile edge computing systems. IEEE Internet Things J. **6**(2), 1879–1892 (2019)
19. Zhang, T., Xu, Y., Loo, J., Yang, D., Xiao, L.: Joint computation and communication design for UAV-assisted mobile edge computing in IoT. IEEE Trans. Industr. Inf. **16**(8), 5505–5516 (2019)
20. Zhang, K., Gui, X., Ren, D., Li, D.: Energy-latency tradeoff for computation offloading in UAV-assisted multiaccess edge computing system. IEEE Internet Things J. **8**(8), 6709–6719 (2020)
21. Lyu, L., Zeng, F., Xiao, Z., Zhang, C., Jiang, H., Havyarimana, V.: Computation bits maximization in UAV-enabled mobile-edge computing system. IEEE Internet Things J. **9**(13), 10640–10651 (2021)