Multi-UAV Mobile Edge Computing and Path Planning Platform Based on Reinforcement Learning

Huan Chang, Yicheng Chen, Baochang Zhang, *Senior Member, IEEE*, and David Doermann, *Fellow, IEEE*

***Abstract*—Unmanned Aerial vehicles (UAVs) are widely used as network processors in mobile networks, but more recently, UAVs have been used in Mobile Edge Computing as mobile servers. However, there are significant challenges to use UAVs in complex environments with obstacles and cooperation between UAVs. We introduce a new multi-UAV Mobile Edge Computing platform, which aims to provide better Quality-of-Service and path planning based on reinforcement learning to address these issues. The contributions of our work include: 1) optimizing the quality of service for mobile edge computing and path planning in the same reinforcement learning framework; 2) using a sigmoid-like function to depict the terminal users' demand to ensure a higher quality of service; 3) applying synthetic considerations of the terminal users' demand, risk and geometric distance in reinforcement learning reward matrix to ensure the quality of service, risk avoidance, and the cost-savings. Simulations have shown the effectiveness and feasibility of our platform, which can help advance related researches. The source code can be found at https://github.com/bczhangbczhang.**

***Index Terms*—Unmanned Aerial Vehicle, Mobile Edge Computing, Path Planning, Reinforcement Learning.**

I. Introduction

Mobile data processing technology is experiencing a growing demand in the communication market. New technologies like 5 G are emerging to accelerate its development. However, the demands of terminal users in uncertain environments and extreme situations have never been perfectly satisfied as the calculations and services are often hard to reach from base stations. As a result, mobile edge computing comes out as one of the fastest-growing topics in telecommunications in the past few decades [1].

Mobile edge computing is a concept that integrates the capabilities of the network, computing, storage, and intelligence services on the edge of the network physically close to the data source. In a typical mobile edge computing scenario, terminal users are served by edge servers with high computing power [2], [3]. The effectiveness of mobile edge computing is measured by the Quality of Service (QoS) of each terminal user. The higher the QoS, the more efficient a terminal user's demand is satisfied or served.

Unmanned Aerial Vehicles (UAVs) have become ideal servers for mobile edge computing that assures QoS, improving stability, reliability, and calculating efficiency through research and development investment [4]–[6]. They are also flexible and cost-effective due to their small size [7], [8]. Therefore, a UAV can move flexibly from terminal user to terminal user and conduct highly efficient calculating services to improve QoS.

UAV-mounted mobile edge computing remains challenging due to the complexity of the working environment, the uncertainty of the distribution of terminal users, and the limitations of the UAV's energy [1]. Therefore, path planning plays an indispensable role when using UAVs for mobile edge computing to handle these issues. For example, [9] proposed a multi-agent algorithm to determine the optimal path of UAVs based on Q-learning. An echo state network (ESN) based prediction algorithm is also proposed for predicting the future movement of terminal users. Liu *et al.* [10] built a model to evaluate QoS and proposed an algorithm to maximize the reward during the planning process.

However, most of the existing research focuses on finding the optimal path under a specified mission or treating the planning process as a simple greedy strategy without a continuously improving process. This prevents the UAVs adaption to the changing environment, and the planning can easily fall into a local optimum. Furthermore, previous works on UAV-mounted mobile edge computing seldom consider risk avoidance or collisions between UAVs as this is impractical in real environments.

To find an adaptive global optimum solution for different tasks, [11], [12] proves that Reinforcement Learning (RL) is effective. [13] dramatically improved QoS by introducing RL and regards the path planning problem as an optimization problem with constraints from the environment. Compared with traditional path planning methods such as the A* algorithm and RRT, RL is more flexible for the following reasons: 1) In the mobile edge computing scenario, the terminal user's residual demand dynamically changes, which requires real-time policy updating. However, traditional methods cannot work efficiently in such time-varying scenarios; 2) There are both obstacles and terminal users on the map, which requires not only obstacle avoidance but also task allocation, and thus it is difficult for

Manuscript received January 20, 2021; revised April 18, 2021; accepted May 9, 2021. Date of publication June 8, 2021; date of current version May 26, 2022. This study was supported by Grant NO.2019JZZY011101 from the Key Research and Development Program of Shandong Province to Dianmin Sun. This work was supported in part by the National Natural Science Foundation of China under Grant 62076016 and 61876015. Huan Chang and Yicheng Chen contributed equally to this paper.*(Corresponding author: Baochang Zhang.)*

Huan Chang is with Beihang University, Beijing 100191, China; The civil engineering department, Imperial College London, South Kensington Campus, London SW7 2AZ, UK (e-mail: changhuan1998@hotmail.com).

Yicheng Chen is with Beihang University, Beijing 100191, China (e-mail: yicheng@buaa.edu.cn).

Baochang Zhang is with Beihang University, Beijing 100191, China (e-mail: bczhang@buaa.edu.cn).

David Doermann is with University at Buffalo, Buffalo, NY 14260 USA (e-mail: doermann@buffalo.edu).

Digital Object Identifier 10.1109/TETCI.2021.3083410

those algorithms that only consider geometrical constraints to handle this issue; 3) In RL, elements in the environment can be depicted uniformly by the cost function, thus making it possible to flexibly adjust the policies by changing coefficients in the cost function according to different mission' needs. As a result, RL is more adaptive to various scenarios and is more suitable to be a base for building the mobile edge computing platform.

Motivated by the reasons mentioned above, we propose a platform to advance research in UAV-mounted mobile edge computing by building a unified framework with path planning algorithms based on RL. The main contributions of this paper are summarized as follows:

- First, we provide a novel framework in which UAV-mounted mobile edge computing and path planning are combined based on RL by considering the geometric distance, risk, and terminal users' demand in a single cost matrix.
- Second, we investigate multi-UAV collaboration in the mobile edge computing scenario. Geometric and terminal user's information are shared among UAVs, thus ensuring cost-saving and obstacle avoidance.
- Third, we introduce an efficient way of depicting the terminal users' demand for achieving a higher QoS. Compared with the traditional linear demand function, the sigmoid-like function enables better task allocation.
- Fourth, we perform extensive experiments to test the proposed platform and evaluate different coefficients in the cost function. Results have shown the effectiveness and feasibility of our approach.

The rest of this paper is organized as follows. First, related work is presented in Section II. Then in Section III, we provide a detailed description of the UAV-mounted mobile edge computing platform. Finally, the effectiveness and feasibility of the proposed platform are validated by simulations in Section IV, and conclusions are presented in Section V.

II. Related Work

We conduct our literature review to cover mobile edge computing, path planning, and their combination.

A. Mobile Edge Computing

Mobile edge computing has attracted increased interest and is becoming one of the hottest topics in edge computing. For example, [14] develops a taxonomy of mobile edge computing applications and use cases. In addition, there some representative reviews: [1] presents related concepts and technologies, architectures, advantages, and typical scenarios of mobile edge computing, [3] explains architecture and computation offloading, and [2] investigates mobile edge computing communications. These works demonstrate the potential of this field.

While the concept of mobile edge computing has been seen in the literature before, it remains an open problem. According to [3], the distribution and management of mobile edge computing resources is a key requirement for ensuring the QoS of terminal users. When servers are moving dynamically, the system benefits from flexibility but at the same time becomes more complicated, which deteriorates the incapability of most arrangement methods.

B. Path Planning

Kim *et al.* [15] introduce a path planning algorithm for dynamic environments where small UAVs are used as relay nodes in a network of naval vessels. The motion estimates of the vessels and the states of the UAVs are taken as input to generalize the strategy. Instead of optimizing a centralized system, the approach exploits a fully decentralized non-linear model predictive control concept. To emphasize the cooperation between UAVs, Zhang *et al.* [16] propose the Cooperative and Geometric Learning Algorithm (CGLA) designed for path planning based on the cooperation of multiple UAVs. CGLA introduces a weight matrix based on geometric distance and integral risk information to guide the movement of UAVs. The weight matrix can be efficiently calculated and updated, making the system lighter than systems based on methods such as neural networks and guarantees real-time path planning. We note that there is a relatively low requirement for computing power in CGLA reinforcement learning, and this makes it a suitable approach for UAV-mounted mobile edge computing.

C. Combination of Mobile Edge Computing and Path Planning

Previous works introduce successive convex approximation (SCA) to combine mobile edge computing and path planning. Jeong *et al.* [17] leverage SCA strategies to calculate the path of UAVs under latency and UAV's energy budget constraints. [18] investigates a scenario where the UAV offloads its computation tasks to multiple ground stations along its trajectory. The authors exploit alternating optimization and SCA techniques to design the UAVs trajectory to minimize the mission completion time. However, these works do not involve a cooperative mechanism between UAVs. In addition, these approaches have a limitation when the environment is unknown in advance. The learning-based algorithms also attract the attention of path planning in UAV-mounted mobile edge computing. For example, [19] investigates a joint task scheduling and resource allocation approach in a space-air-ground integrated network based on policy gradients and actor-critic methods. [20] applies a deterministic policy gradient algorithm to maximize the uplink sum rate in the UAV-aided cellular network with multiple ground users.

Inspired by these works, we provide an organic combination of mobile edge computing and path planning by building an open-source platform for the UAV-mounted mobile edge computing networks. Simulation results have shown the feasibility and flexibility of our platform.

III. Multi-UAV Mobile Edge Computing and Path Planning Based on Reinforcement Learning

In reinforcement learning, the agent's goal is to find an optimal strategy under each scenario to obtain the maximum expected reward. In our platform, the mobile network processor UAV is the agent who constantly learns from the environment. At each time slot, a UAV chooses a planning strategy to achieve the best possible reward according to its surroundings. After the UAV moves, the surrounding changes and positive or negative feedback is provided to the UAV in the form of a reward matrix A given the factors of risk, geometric distance, and terminal users' demand. The UAV then learns from the surroundings through a stochastic iterative cost matrix G generated from A and selects a strategy—a path toward the target. G can be considered as the memory of each agent, which is intensified and "trained" through each episode of the planning process.

A. Environment Modeling

This paper considers the UAV's collision avoidance and terminal users' demand on the same platform. The surroundings contain two basic elements of obstacles and terminal users. First, obstacles vary in shape, positions, and risk levels, which include buildings, cars, or mountains in a real environment. Second, we assume the obstacles comply with a Gaussian distribution but have different variances σ used to calculate their risk exposure probability.

For n independent obstacles in the map, giving the ith obstacle position $O_i = (X_i, Y_i)$, the risk $r_i(x, y)$ indicates the risk from O_i at the point (x, y) and can be defined as

$$r_i(x,y) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{d^2}{2\sigma}},\ d = \sqrt{(x - X_i)^2 + (y - Y_i)^2},\quad i \in \{1, 2, \ldots, n\}. \tag{1}$$

Considering all n obstacles in the map, the overall risk to a point (x, y) in the risk exposure probability matrix can be described as

$$R(x,y) = 1 - \prod_{i=1}^{n} [1 - r_i(x,y)]. \tag{2}$$

The exposed risk from any point p to any point q on the map is the integral risk of $R(x, y)$ for any (x, y) on C, where C is the linear path from p to q:

$$\int_{(x,y)\in C} R(x,y). \tag{3}$$

Second, for serving terminal users, we assume that each terminal user has an initial demand d_j^0 for the UAVs to process. We also assume the demand can only be served by UAVs within a constant service radius because the UAVs have limited capability for detecting demand signals beyond a certain distance. Therefore, the service area is denoted $s(p_j, \epsilon)$, where p_j is the position of TU_j, and ϵ is the service radius, as shown in Figure 1.

When a UAV enters the service range of TU_j, the service for TU_j begins. The remaining demand of TU_j will decrease at a constant speed τ per unit time per UAV. We can easily infer that a terminal user with a greater demand needs more time to be served, and the longer a UAV remains in the service range of TU_j, the more service can be provided for TU_j. d_j changes over time t_k time served by UAV_k as

$$d_j^{l+1} = d_j^l - \tau t_k. \tag{4}$$

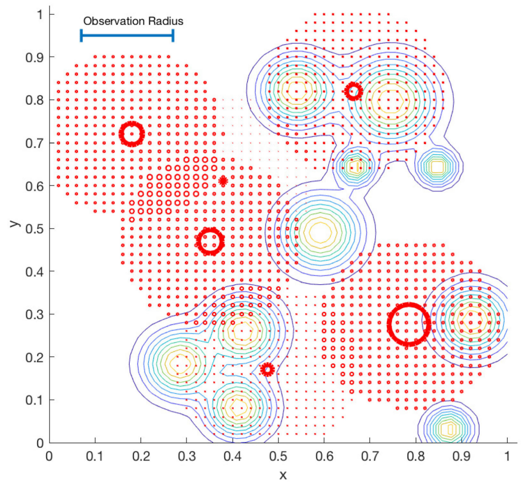


Fig. 1. Obstacles risk and terminal users demand map ($\epsilon = 0.2$).

The correlation between the terminal users' demand and UAVs' demand detection should have a non-linear relationship to improve system performance. With reference to [10] and [21], a sigmoid-like function can help enhance strong signals and abate weak signals. Thus, we adopt a sigmoid-like demand detection function $U(d_j) \in (0, 1]$ to describe the correlation between real demand and detected demand as

$$U(d_j) = 1 - \exp\left[-\frac{(d_j)^\eta}{d_j + \beta}\right], \tag{5}$$

where η and β are controlling variables.

From Figure 2(a), $U(d_j)$ first increases steeply as the demand rises and becomes steady when the demand is sufficiently great. Thus, (5) can encourage a UAV to focus on terminal users with greater unserved demand and prevent it from serving any terminal user over a long time, thus improving QoS.

Generally, a Sigmoid-like function is an increasing function with an inflection point x_0, which follows $\frac{d^2 f(x)}{dx^2} > 0$ when $x < x_0$ and $\frac{d^2 f(x)}{dx^2} < 0$ when $x > x_0$ [21]. Functions with such form satisfy the following properties:

Property 1: For any $x > 0$ in $U(x)$, the function is valid only when

$\eta \in (1, \infty)$,
$\beta \in (0, \infty)$

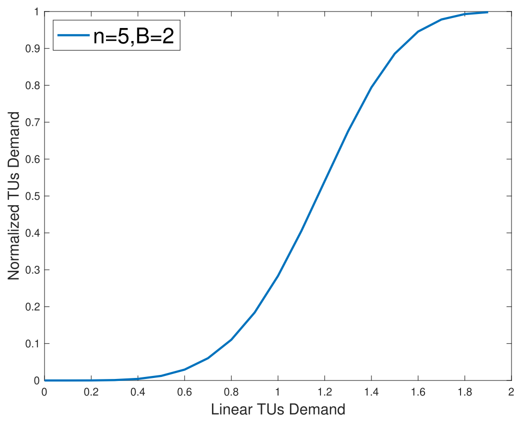
The proof is given in Proof 1 in the appendix.

Property 2: η controls the slope and centrality of the curve, pivots through an inflection point $(1, 1 - e^{-\frac{1}{1+\beta}})$. β controls the horizontal movement of the curve. As a result, the intersection point can be moved vertically by changing β.

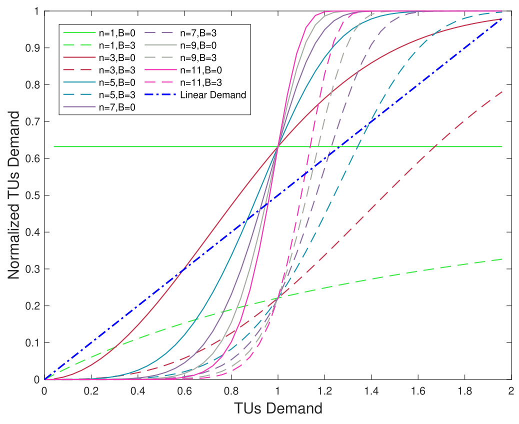
The proof is given in Proof 2 in the appendix.

As shown in Figure 2(b), $U(x)$ has an inflection point when $x = 1$. The curve becomes steeper when η increases and moves vertically downward when β increases. Therefore, η and β are constant variables that affect QoS. The evidence will be provided in part IV.

To simplify the model, we assume the demands are linearly accumulated. For a UAV at a point p in the map, with a detection range ϵ, the detected demand is the linear accumulation of

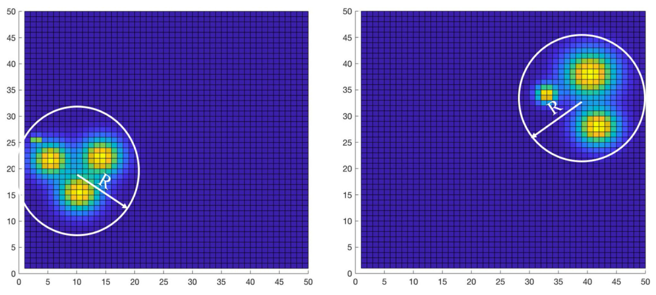


(a) $SingleSigmoidFunction$



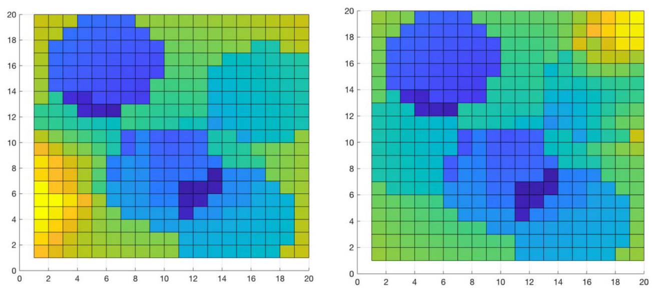
(b) $SigmoidFunctions$

Fig. 2. Examples of normalized sigmoid demand functions.



(a) $P = (0.15, 0.4)$ (b) $P = (0.75, 0.6)$

Fig. 3. Observed risk distribution ($R = 0.2$).



(a) $Pos = (0.15, 0.4)$ (b) $Pos = (0.75, 0.6)$

Fig. 4. Weight matrix on point (0.5,0.5).

terminal users' demand within circle area $s(p, \epsilon)$:

$$\sum_{j \in s(p,\epsilon)} U(d_j). \tag{6}$$

B. The Reward Matrix

A reward matrix is introduced for UAVs to learn and adapt to find the optimal path. The reward matrix is designed to measure the reward or punishment from any point towards any other points on the map given the factors of risk, geometric distance, and terminal users demand.

In our platform, the map is represented as a lattice of $N \times N$, and the reward A_{p_i,p_r} between any point p_i and p_r in the map is defined as

$$A_{p_i,p_r} = d_{p_i,p_r} + K \int_C R(x,y)ds + \frac{M}{1 + \sum_{j \in s(p_i,\epsilon)} U(d_j)}, \tag{7}$$

where d_{p_i,p_r} is the geometric distance between p_i and p_r. The second term in the equation denotes the risk detected from p_i to p_r or vice versa, which means the larger the risk detected, the larger the cost or punishment. The last term in the equation is the overall demand detected at p_i, as p_i is the current position of UAV. The larger the demand detected, the smaller the punishment, or the larger the reward.

For each point $p_r, r \in \{1, 2, \ldots, N^2\}$ in the map, a reward matrix A_{p_r} consists of points $A_{p_i,p_r}, i \in \{1, 2, \ldots, N^2\}$ is generated with regard to all points p_i on the map. K and M reflect the tolerance of risk and priority of service, which affects the strategy for path planning. When applied in real situations, K and M can be adjusted according to the mission's requirements. For instance, if K is set to a relatively high value, the UAVs will tend to stay away from obstacles even though this leads to a longer path length.

An obstacle observation radius is introduced for each UAV to fit real situations. When an obstacle enters the observed area of a UAV, the UAV detects the obstacle and obtains risk information. Only the observed obstacles will count as a risk when calculating the weight matrix. The highlighted color in Figure 3 shows the observed risk calculated by (1) and (2) for a UAV in position P with an observation radius R.

Figure 4 $(a), (b)$ shows the reward matrix on point [0.5,0.5] based on each UAV's observed risk distribution in Figure 3 and the overall terminal user's demand. According to (7), the terminal user's demand decreases the cost, and the observed risk adds the cost. In Figure 4, we can infer that places with more terminal user demand have lower values (dark color), and those with more obstacles have higher values (light color).

C. Cost Matrix

During the path planning process, a cost matrix G is introduced for a UAV to obtain a preliminary optimal path to the destination. The generation of the cost matrix is obtained using an iterative process. After several iterations, the cost matrix will converge [16] and remain steady. The update mechanism of the cost matrix in a map with $N \times N$ computing nodes is described as:

1) `Initialize G`: initialize the cost matrix G^0. Assign the value 0 to the target point and the value ∞ to all other points.
2) Update the cost matrix G. Randomly choose a position p_r from the map. For each point p_i in the map, update the point value in G by comparing the current value with the revised value considering the reward matrix:

$$G_{p_i}^{k+1} = min\left\{G_{p_i}^{k}, A_{p_i,p_r} + G_{p_r}^{k}\right\}, \\ i, r \in \left\{1, 2, \ldots, N^2\right\}. \tag{8}$$

3) Repeat step 2) until reaching the maximum number of iterations.

After G is produced, an ordinal sequence of points is generated as a preliminary $Path$ for the UAV to follow. Each UAV has its own $Path$. Points with the lowest cost in G are constantly added. The generation of $Path$ is described as follows:

1) Initialize $Path$ as an empty list.
2) Add p_i with the lowest value in G to $Path$, then assign G_{p_i} to ∞.
3) Repeat step 2) until reaching the target point or reaching a maximum length.

We note that the elements in $Path$ are in ascending order of cost. The process of calculating G and generating $Path$ together form the function `Planning` in Algorithm 1.

D. UAVs Movement

Each UAV is considered an agent in the RL process. They are therefore assigned a memory D_i and a cost matrix G^i. D_i stores map information and serves as the "eyes" and memory of the agent, and G^i serves as the "brain" of the agent. The agent produces a learned result $Path_i$ to complete each episode. All UAVs in the system move in sequence to realize information sharing. This can result in a slight time delay in real situations. When one UAV moves, the other UAVs would be treated as obstacles. Algorithm 1 describes the process. For the i^{th} UAV (UAV_i), after moving one step according to $Path_i$, `ScanEnv` is performed. In `ScanEnv`, UAV_i scans the circle area $s(pos_i, R)$ where R is the observation radius. This is used to decide whether further planning is needed. If new obstacles, including other UAVs, are observed, $ObstacleFound$ will be set to $True$, and memory D_i will be updated. Then, the weight matrix and G^i will be altered, and $Path_i$ will be recalculated. If the surroundings remain unchanged, UAV_i continues to `Move` according to $Path_i$. In `Move`, UAV_i moves a distance of $StepLength$ along the direction of the vector starting from pos_i and ending at $Path_i[1]$, if the distance between pos_i and $Path_i[1]$ is smaller than $StepLength$, the UAV will move directly to $Path_i[1]$. `Move` returns the new position of UAV_i. At the end of one loop, the remaining demands of all TUs within the service area of UAV_i are updated according to (4).

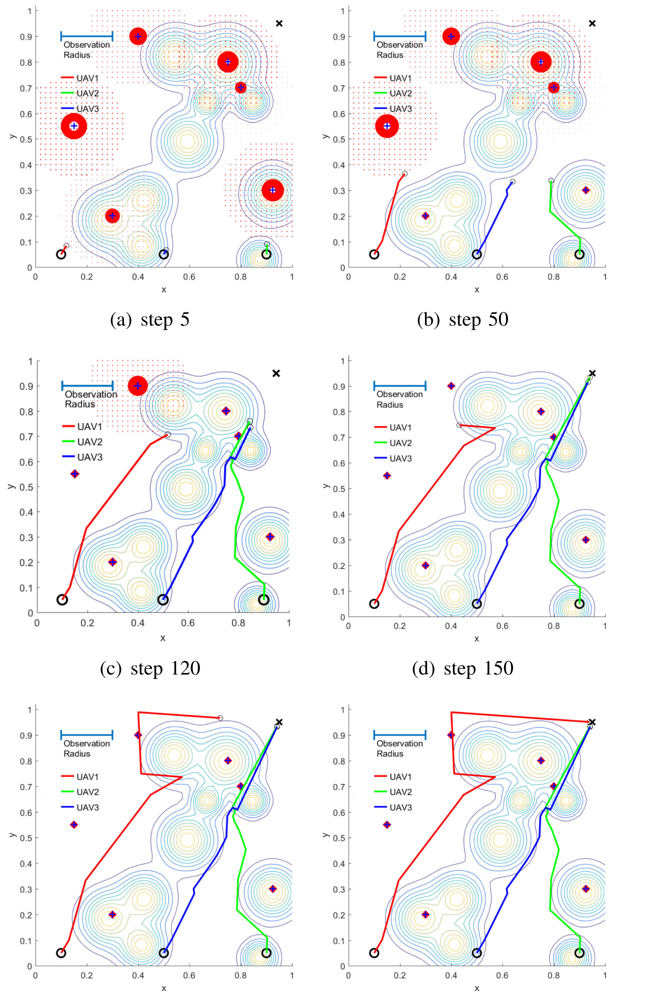
Algorithm 1: UAV Movement Algorithm.
```
1:  for $i$ in $UAVnum$ do
2:    Initialize G($i$)
3:    $Path_i \leftarrow$ Planning()
4:  end for
5:  for $i$ in $UAVnum$ do
6:    if $pos_i = TargetPoint$ then
7:      Stopmovement($i$)
8:    else
9:      // Remove outdated information from $D_i$ because
        $pos_j$ has changed in last loop
10:     for $j$ in $UAVnum$ and $j! = i$ do
11:       delete $pos_j$ from memory $D_i$
12:     end for
13:     $ObstacleFound \leftarrow$ ScanEnv($pos_i, R$)
14:     if $ObstacleFound$ then
15:       $Path_i \leftarrow$ Planning()
16:     end if
17:     if $pos_i = Path_i[1]$ then
18:       $Path_i \leftarrow Path_i[2...end]$
19:     end if
20:     $pos_i \leftarrow$ Move($StepLength, Path_i[1]$)
21:     for $TU_j$ within $s(pos_i, \epsilon)$ do
22:       $d_j \leftarrow d_j - \tau$
23:     end for
24:   end if
25: end for
```

IV. Simulation and Discussion

This section first illustrates the UAV dynamic planning process and discusses the influence of the parameters K and M in (7) on the planning results. Second, we demonstrate the efficiency of the sigmoid demand function (4) by comparing it with a linear demand function. Finally, we compare our algorithm with a commonly used algorithm A*, which shows that our planning algorithms achieve a much better result in terms of QoS.

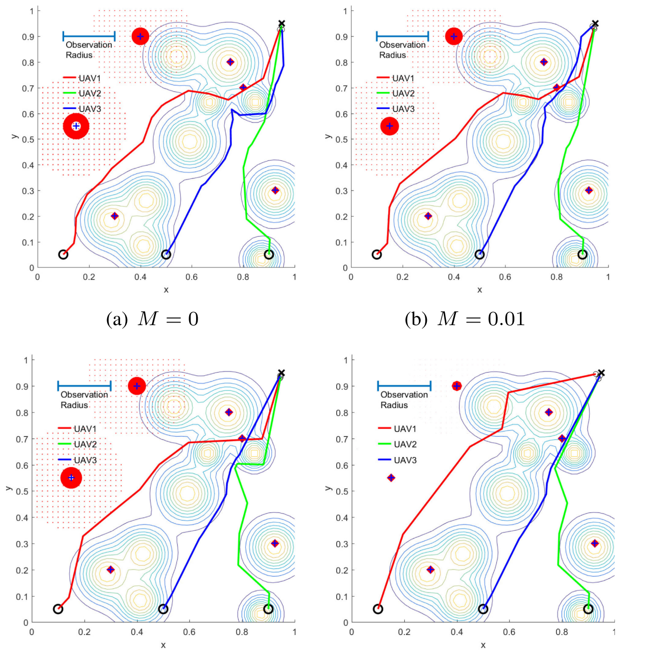
To demonstrate our algorithm, we have made several simplifying assumptions:

- The site is abstracted to grids so that objects, including obstacles, terminal users, and UAVs, are aligned with the grid.
- All object information is stored in databases for UAVs to 'scan,' while in real situations, the scanning process could be realized by perception algorithms based on sensors like cameras or radars.
- The variation in speed is realized through adjusting $StepLength$. In our proposed algorithm, each UAV takes 1 unit of time to move one $StepLength$. If the distance between a UAV's current position and its planned position is smaller than $StepLength$, the UAV will move to the planned position instead of moving one $StepLength$. In practical situations, the calculation distance in a map could be set to smaller than one $StepLength$, which would require speed variations. However, we are assuming a



(a) step 5 (b) step 50 (c) step 120 (d) step 150 (e) step 180 (f) step 191

Fig. 5. Simulation of UAV-mounted mobile edge computing based on proposed path planning platform.



(a) $M = 0$ (b) $M = 0.01$ (c) $M = 0.1$ (d) $M = 0.4$

Fig. 6. Planning results with different M ($K = 5$).

2D planning scenario. The altitude of the UAV in a 3D environment is not considered.

A. Path Planning for Multiple UAVs

In this simulation, we set $K = 20$, $M = 1$, $\eta = 2$, $\beta = 8$, $\epsilon, R = 0.2$. K, M is decided by users of the platform with different needs, while other parameters are determined by practical conditions and the UAVs' capabilities. Ten obstacles are given random positions with random variances $\sigma_i > 0$, $i = 0, 1, 2, \ldots, 10$. Six terminal users are assigned random demands $d_j \in [0, 10]$, $j = 0, 1, 2, \ldots, 6$. In a real situation, the terminal user's demand could be a real-time variable.

The planning process of the three UAVs is shown in Figure 5. Note that:

- The point with the black cross marks the target point for all UAVs. All UAVs are tasked to deliver service to terminal users on the map and fly to the target point for each mission.
- The red dots show the existence and the amount of the terminal users' demand with a service radius ϵ. The red dots shrink while terminal users are being served, representing a reduction in the remaining demand. As can be seen from the overlap area of the service radius, the demands are cumulative.

From Figure 5, we can infer that UAVs can choose a low-risk path to serve each terminal user in a complex environment based on our platform. Terminal users with higher demand are more attractive to UAVs. After the demand is reduced, UAVs will change direction for other demand-high areas. Meanwhile, information sharing is effective in avoiding collisions between UAVs during planning.

B. The Evaluation of M

The parameter M decides the priority of serving the terminal users and therefore controls the QoS. As shown in Figure 6, given a fixed K, in the scenarios with larger M, UAVs are prone to meet more terminal users demands but take more risk and sacrifice path length (i.e., energy) to do so. On the contrary, UAVs in the scenarios with smaller M fail to serve all terminal users but save more energy on a shorter path.

The same results are also obtained by numerical simulation. To compare the service rate, we define

$$QoS = 1 - \frac{\sum_{j=1}^{m} d_j}{\sum_{j=1}^{m} d_j^0}. \tag{9}$$

As shown in Figure 7, when M increases, QoS and average risk increase accordingly.

By changing M, our platform can meet the requirements of different missions with different service needs. QoS and risk can be flexibly balanced.

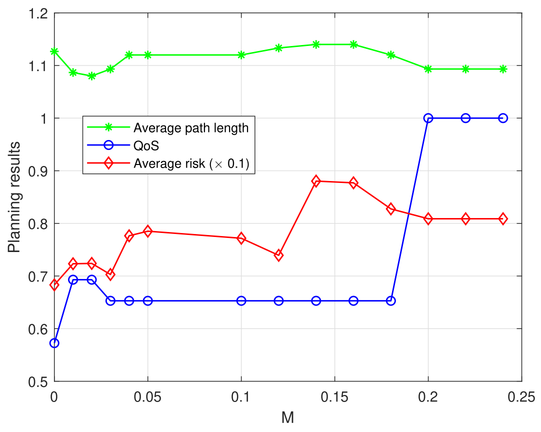


Fig. 7. Planning result measurement with different M ($K = 5$).

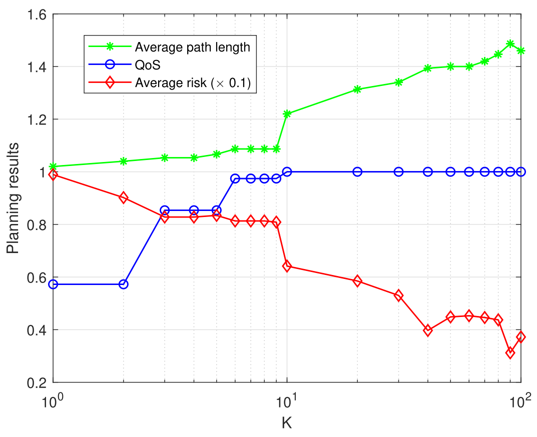


Fig. 9. Planning results measurements with changing K ($M = 0.5$).

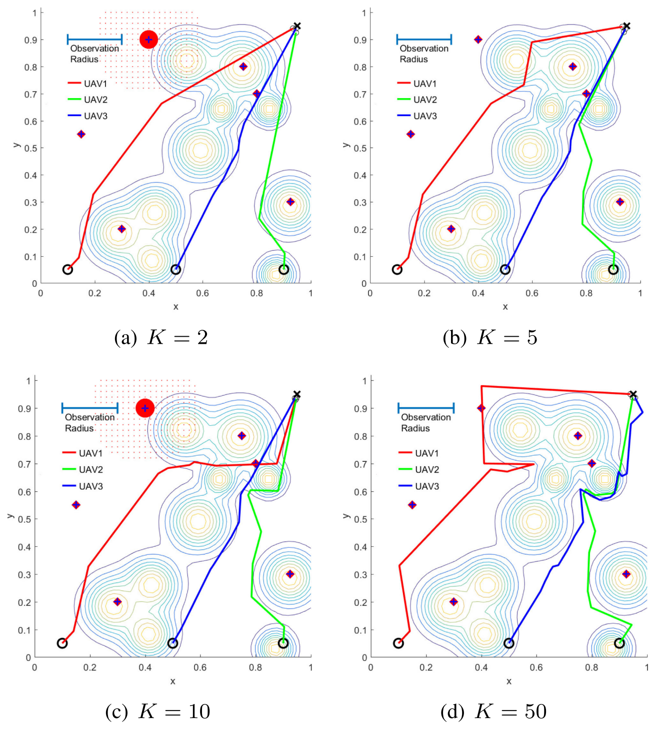


Fig. 8. Planning results with different K ($M = 0.2$).

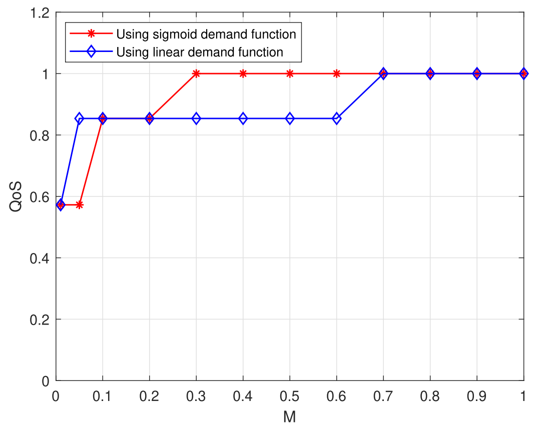


Fig. 10. QoS comparison using sigmoid or linear demand function ($K = 10$).

C. The Evaluation of K

Similar to parameter M, altering the parameter K can make the algorithm more flexible in different environments. K controls the tolerance of risk. UAVs set with higher K in the result tend to sacrifice energy cost to avoid risk and thereby influence QoS. Figure 8 compares the path planning results with different K. When K increases, a higher risk avoidance strategy is adopted, and the UAV follows a path to avoid obstacles at all costs instead of flying through narrow tunnels to serve the terminal users.

Numerical results are shown in Figure 9 to better illustrate the algorithm. When K increases, path length and path risk move in opposite directions.

D. Comparison of the Sigmoid Demand Function and Linear Demand Function

According to [10], a sigmoid processed demand can improve system performance. To verify the effectiveness of (5), we performed experiments separately with the sigmoid demand function $\sum_{j\in s(p,\epsilon)} U(d_j)$ and the linear demand function $\sum_{j\in s(p,\epsilon)} d_j$ where $d_j \in [0, 1]$. Results show that the former gives higher QoS with the same experimental conditions and leads to higher service speed with the same QoS. Figure 10 shows that the sigmoid demand function assures a higher QoS than the linear demand function.

Table I compares the service completion time of each terminal user with the two schemes. Accordingly, the sigmoid demand function leads to a higher terminal user service speed with the same QoS. This is because $U(d_j) > d_j$ in the early period, as shown in (7) and (8). The UAVs are attracted to the terminal users earlier and thus finish the service more quickly.

E. Comparison of Our Algorithm With the A* Algorithm

The A* algorithm is widely used as a baseline in various path planning scenarios. We compare our proposed algorithm with A* in the multiple UAV mobile edge computing environment.

TABLE I
SERVICE SPEED COMPARISON USING SIGMOID OR LINEAR DEMAND FUNCTION

QoS	Ratio of service completion time (sigmoid vs linear)					
	Terminal user 1	Terminal user 2	Terminal user 3	Terminal user 4	Terminal user 5	Terminal user 6
0.85 ($K = 10, M = 0.1$)	1	1	1	1.03	0.82	N
0.85 ($K = 10, M = 0.2$)	0.96	1	0.96	1	0.96	N
1.00 ($K = 10, M = 0.7$)	1	1	1	1	0.95	0.96

TABLE II
RESULTS COMPARISON WITH DIFFERENT M OF THE PROPOSED ALGORITHM AND THE A* ALGORITHM ($K = 2$)

M	Proposed algorithm			A* algorithm		
	QoS	Average path length	Average risk	QoS	Average path length	Average risk
0.01	0.87	1.06	1.68	0.36	1.13	0.30
0.05	0.85	1.07	2.00	0.39	1.13	0.30
0.10	0.88	1.09	2.00	0.56	1.17	0.32
0.15	0.88	1.09	2.62	0.56	1.17	0.32
0.50	1.00	1.09	4.42	0.61	1.19	0.31
1.00	1.00	1.09	4.33	0.61	1.19	0.31
2.00	1.00	1.17	3.83	0.61	1.19	0.31
5.00	1.00	1.17	4.23	0.61	1.19	0.31
10.0	1.00	1.17	4.23	0.61	1.19	0.31

TABLE III
RESULTS COMPARISON WITH DIFFERENT K OF THE PROPOSED ALGORITHM AND A* ALGORITHM ($M = 0.5$)

K	Proposed algorithm			A* algorithm		
	QoS	Average path length	Average risk	QoS	Average path length	Average risk
0.50	1.00	1.07	6.98	0.59	1.13	0.61
1.00	1.00	1.09	4.77	0.59	1.17	0.45
2.00	1.00	1.09	4.42	0.61	1.19	0.31
3.00	1.00	1.09	4.42	0.61	1.17	0.28
5.00	1.00	1.10	4.54	0.59	1.19	0.21
10.0	1.00	1.12	1.41	Deadlock		
20.0	1.00	1.12	1.16			
50.0	1.00	1.13	0.42			
100	1.00	1.15	0.36			

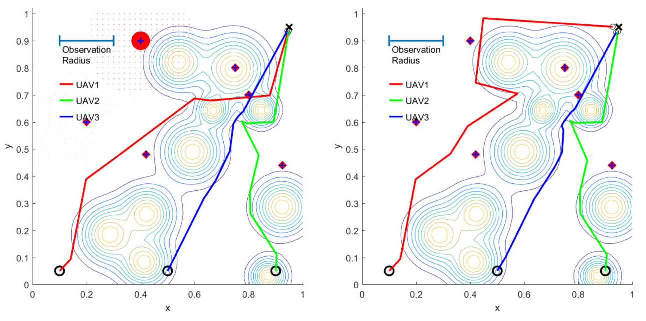
Based on the A* algorithm, at each step of path planning, a UAV can choose among one of a fixed number of equally distributed directions to move one unit step. In our experiments, we set up eight directions for UAVs and thus have eight candidate nodes p_i for a UAV to choose at each step. Considering the path length and risk as a cost and the demand of the terminal users as a reward, we formulate the weight function F_i at each point p_i in the A* algorithm as

$$F_i = d_{p_i,p_t} + KR_{p_i} + \frac{M}{1+\sum_{j\in s(p_i,\epsilon)} U(d_j)},$$
$$i = 1, 2, \ldots, 7, 8, \tag{10}$$

where d_{p_i,p_t} is the Euclidean distance between candidate point p_i and the target point p_t. R_{p_i} and $U(d_j)$ are defined in the previous equations.

In general, Figure 10 shows the A* algorithm fails to perform effective service when terminal users are surrounded by obstacles. However, with the proposed algorithm (Figure 12(b)), the UAVs manage to serve the terminal users while avoiding risk.

QoS is the most important index in a mobile edge computing mission. As shown in Table II and Table III, the proposed algorithm can achieve a higher QoS compared to the A* algorithm. Furthermore, our algorithm can react flexibly to the change of K and M, thus adjust priorities to path length and average risk while assuring QoS. For example, with the proposed algorithm



(a) Using linear demand function (b) Using sigmoid demand function

Fig. 11. Results comparison using linear or sigmoid demand function ($K = 10, M = 0.12$).

($K = 50, M = 0.5$), all three indices are better than using the A* algorithm ($K = 0.5, M = 0.5$).

A deadlock means that the mission can never come to an end because one or more UAVs are stuck in the current surrounding and loop infinitely. Our experiments show that the A* algorithm can easily fall into a deadlock with parameter K above a certain level. The results are illustrated in Table III and Figure 13.

Based on the above comparisons, our RL platform performs better not only on environment adaptation and high QoS assurance but also on algorithm reliability and high mission

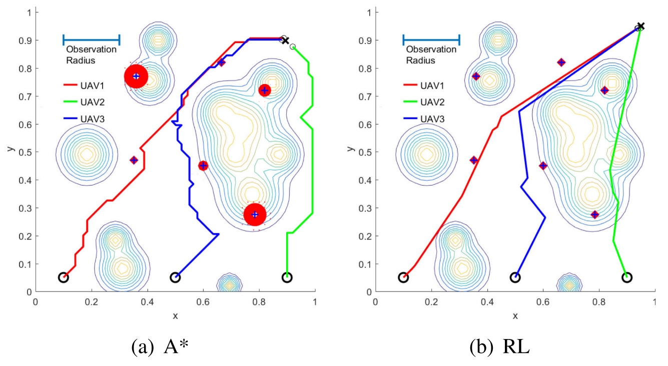


Fig. 12. Comparison of RL and A* algorithm ($K = 5, M = 0.5$).

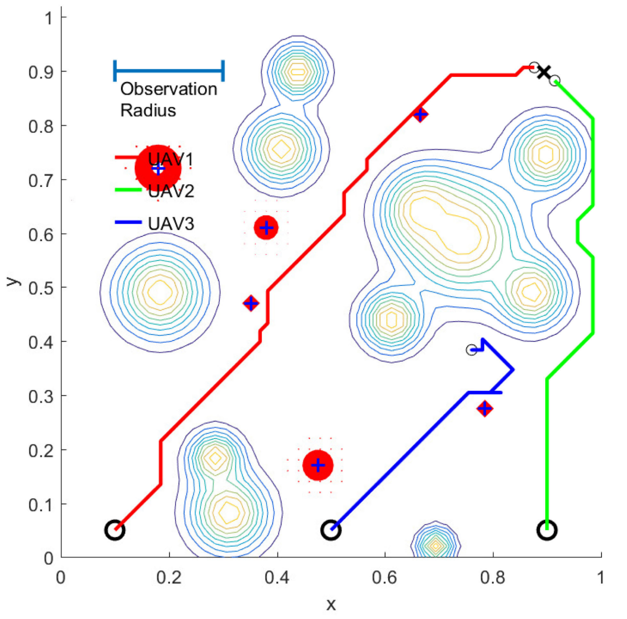


Fig. 13. The deadlock of the A* algorithm ($K = 1, M = 0.2$).

completeness. The effectiveness and feasibility of our method are clear.

V. Conclusion

This paper develops the first multi-UAV mobile edge computing and path planning platform where UAVs serve terminal users as mobile network processors based on reinforcement learning. We implemented our platform with 1) a Quality-of-Service (QoS) for each terminal user, 2) maximum collision avoidance with minimum risk, and 3) cooperation between UAVs. The simulations and experiments are provided to show the efficiency and usability of the platform, which can be a useful baseline for mobile edge computing.

Appendix

Proof: 1 According to Property 1, for any $x > 0$ in $U(x)$,

$$\frac{dU}{dx} = \frac{x^{\eta-1}e^{-\frac{x^\eta}{x+\beta}}}{(x+\beta)^2}[(\eta-1)x+\eta\beta] > 0, \tag{11}$$

$$1 - e^{-\frac{x^\eta}{x+\beta}} > 0. \tag{12}$$

then we have

$$(\eta-1)x+\eta\beta > 0, \tag{13}$$

$$\frac{x^\eta}{x+\beta} > 0. \tag{14}$$

According to (14), because $x^\eta > 0$, then for any $x > 0, x+\beta > 0$, so $\beta > 0$. By (13), we have $\eta > \frac{x}{x+\beta} > 0$, then $\eta > 0$.

If $\eta <= 1$, then $x <= \frac{\eta\beta}{1-\eta}$, which contradicts the domain of x, thus we have $\eta > 1$ ■

Proof: 2 We have

$$\frac{dU}{d\eta} = \frac{\ln x e^{-\frac{x^\eta}{x+\beta}}}{x+\beta}x^\eta, \tag{15}$$

thus we have

$$\frac{dU}{d\eta} < 0, x \in [0,1), \tag{16}$$

$$\frac{dU}{d\eta} > 0, x > 1. \tag{17}$$

It means as η increases, $U(x)$ decreases when $x < 1$ and increases when $x > 1$, and the slope of the curve increases accordingly. When $x = 1$, $\frac{dU}{d\eta} = 0$ for any η, it creates an inflection point that is not affected by changing η.

We also have

$$\frac{dU}{d\beta} = -\frac{x^\eta e^{-\frac{x^\eta}{x+\beta}}}{(x+\beta)^2} < 0, \tag{18}$$

which means when β increases, the sigmoid demand curve drops vertically to the inflection point. ■

References

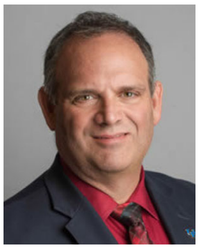
[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2017.
[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, 2017.
[3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1628–1656, 2017.
[4] Y. Du, K. Yang, K. Wang, G. Zhang, Y. Zhao, and D. Chen, "Joint resources and workflow scheduling in UAV-enabled wirelessly-powered MEC for IoT systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 10187–10200, Oct. 2019.
[5] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 36–42, May 2016.
[6] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in UAV communication networks," *IEEE Commun. Surv. Tut.*, vol. 18, no. 2, pp. 1123–1152, 2015.
[7] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multiuav-enabled load-balance mobile edge computing for iot networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898–6908, 2020.
[8] L. Yizhe, D. Wenrui, B. Zhang, W. Huang, and L. Chunhui, "Optimization of bits allocation and path planning with trajectory constraint in uav-enabled mobile edge computing system," *Chin. J. Aeronaut.*, vol. 33, no. 10, pp. 2716–2727, 2020.
[9] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory design and power control for multi-UAV assisted wireless networks: A machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7957–7969, Aug. 2019.

[10] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for uav-mounted mobile edge computing with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5723–5728, May 2020.
[11] G. Faraci, C. Grasso, and G. Schembra, "Reinforcement-learning for management of a 5G network slice extension with uavs," in *Proc. IEEE INFOCOM 2019-IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*. IEEE, Paris, France, 2019, pp. 732–737.
[12] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, 2019.
[13] Q. Wang, W. Zhang, Y. Liu, and Y. Liu, "Multi-UAV dynamic wireless networking with deep reinforcement learning," *IEEE Commun. Lett.*, vol. 23, no. 12, pp. 2243–2246, Dec. 2019.
[14] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. 6th Int. Conf. Adv. Future Internet. Citeseer*, 2014, pp. 48–55.
[15] S. Kim, H. Oh, J. Suk, and A. Tsourdos, "Coordinated trajectory planning for efficient communication relay using multiple UAVs," *Control Eng. Pract.*, vol. 29, pp. 42–49, 2014.
[16] B. Zhang, W. Liu, Z. Mao, J. Liu, and L. Shen, "Cooperative and geometric learning algorithm (CGLA) for path planning of uavs with limited information," *Automatica*, vol. 50, no. 3, pp. 809–820, 2014.
[17] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2017.
[18] X. Cao, J. Xu, and R. Zhang, "Mobile edge computing for cellular-connected UAV: Computation offloading and trajectory optimization," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*. Kalamata, Greece, 2018, pp. 1–5.
[19] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for iot applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.
[20] S. Yin, S. Zhao, Y. Zhao, and F. R. Yu, "Intelligent trajectory design in uav-aided communications with reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8227–8231, Aug. 2019.
[21] J.-W. Lee, R. R. Mazumdar, and N. B. Shroff, "Non-convex optimization and rate control for multi-class services in the Internet," *IEEE/ACM Trans. Netw.*, vol. 13, no. 4, pp. 827–840, Aug. 2005.

Huan Chang received the bachelor's degree in management science and engineering from Beihang University, Beijing, China, in 2020. She is currently working toward the Master's degree with Civil Engineering Department, Empire College London, Ilford, U.K. Her study interests include RL-based path planning and optimization, supply chain management, and operational research.

Yicheng Chen is currently working toward the B.E. degree in automation with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include motion planning, pattern recognition, and reinforcement learning.

Baochang Zhang (Senior Member (M'2010-SM'2021), IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from the Harbin Institute of the Technology, Harbin, China, in 1999, 2001, and 2006, respectively. From 2006 to 2008, he was a Research Fellow with the Chinese University of Hong Kong, Hong Kong, and with Griffith University, Brisbane, QLD, Australia. From 2014 to 2015, he was also a Senior PostDoc Fellow with the Italy Institute of Technology, Italy. He is currently a Professor with Beihang University, Beijing, China. His current research interests include deep learning, pattern recognition, object recognition and tracking, and wavelets.


David Doermann (Fellow, IEEE) is currently a Professor of empire innovation with the University at Buffalo (UB), Buffalo, NY, USA, and the Director of the University at Buffalo Artificial Intelligence Institute. Prior to coming to UB, he was a Program Manager with the Defense Advanced Research Projects Agency, where he developed, selected, and oversaw research and transition funding in computer vision, human language technologies, and voice analytics. From 1993 to 2018, he was a Member of the research faculty with the University of Maryland, College Park, MD, USA. He has more than 250 publications in conferences and journals, is a Fellow of the IAPR, has numerous awards, including an honorary doctorate from the University of Oulu, Finland, and is a founding Editor-in-Chief of the *International Journal on Document Analysis and Recognition*.